HAL
open science

La méthode multipôle rapide en électromagnétisme. Performances, parallélisation, applications

Guillaume Sylvand

▶ **To cite this version:**

Guillaume Sylvand. La méthode multipôle rapide en électromagnétisme. Performances, parallélisation, applications. Mathématiques [math]. Ecole des Ponts ParisTech, 2002. Français. NNT : . tel-00005683

HAL Id: tel-00005683

https://pastel.hal.science/tel-00005683

Submitted on 5 Apr 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

La Méthode Multipôle Rapide en Electromagnétisme : Performances, Parallélisation, Applications

Guillaume SYLVAND (CERMICS/INRIA)

18 juin 2002

Table des matières

Introduction **9**

I La Méthode Multipôle Rapide : Version séquentielle **13**

1 Présentation de la méthode **15**
Introduction . . . 15
1.1 Technique de résolution . . . 16
1.1.1 Modèle physique considéré . . . 16
1.1.2 Modélisation mathématique . . . 16
1.1.3 Discrétisation . . . 18
1.1.4 Résolution numérique . . . 19
1.2 Méthode mono-niveau . . . 21
1.2.1 Premier survol . . . 21
1.2.2 Approfondissement . . . 29
1.3 Méthode multi-niveau . . . 45
1.3.1 Premier survol : calcul à 2 niveaux . . . 45
1.3.2 Approfondissement . . . 61
Conclusion . . . 76

2 Implémentation et Optimisations **79**
Introduction . . . 79
2.1 Utilisation d'une liste de tâches . . . 80
2.1.1 Principe de la méthode . . . 80
2.1.2 Tâches élémentaires . . . 82
2.1.3 Création de la liste . . . 83
2.2 Traitement des opérations de transfert . . . 86
2.2.1 Expressions de la matrice de transfert . . . 86
2.2.2 Optimisation visant à diminuer le nombre de matrices calculées . . . 96
2.2.3 Synthèse . . . 99
2.3 Traitement des opérations de montée/descente . . . 100
2.3.1 Expressions continues des montées/descentes . . . 100
2.3.2 Translation . . . 101
2.3.3 Extrapolation et réduction . . . 102
2.3.4 Avant ou après ? . . . 105

2.3.5 Récapitulatif . . . 106
2.4 Traitement des autres opérations . . . 107
2.4.1 Initialisations . . . 107
2.4.2 Intégrations . . . 108
2.4.3 Interactions proches . . . 109
2.5 Optimisation de la mémoire . . . 111
2.5.1 Limitation . . . 111
2.5.2 Réordonnancement des sous-listes . . . 112
2.5.3 Réécriture out-of-core . . . 112
2.5.4 Représentation graphique . . . 117
Conclusion . . . 120

3 Performances et Applications 121
Introduction . . . 121
3.1 Présentation technique . . . 121
3.1.1 Langages . . . 121
3.1.2 Bibliothèques . . . 122
3.1.3 Plates-formes . . . 122
3.2 Présentation des cas tests . . . 122
3.2.1 Sphère . . . 122
3.2.2 Cetaf . . . 123
3.2.3 Avion . . . 124
3.2.4 Raffinement de maillages . . . 124
3.3 Comparaison avec les méthodes classiques . . . 125
3.3.1 Objectif . . . 125
3.3.2 Temps d'exécution . . . 126
3.3.3 Stockage . . . 126
3.3.4 Précision . . . 127
3.3.5 Synthèse . . . 128
3.4 Choix des paramètres . . . 128
3.4.1 Nombre de pôles . . . 129
3.4.2 Nombre de groupes . . . 132
3.4.3 Raffinement du maillage . . . 133
3.4.4 Formulation intégrale en multipôle . . . 135
3.4.5 Points de Gauss . . . 136
3.4.6 Taille des feuilles . . . 138
3.4.7 Niveau plafond . . . 141
3.4.8 Précision machine . . . 142
3.4.9 Nombre de composantes . . . 143
3.4.10 Stockage des matrices de transfert et de translation . . . 144
3.5 Comparaison par machine . . . 145
3.5.1 Comparaison matérielle . . . 145
3.5.2 Comparaison logicielle . . . 147
3.6 Scalabilité . . . 147
3.6.1 Introduction . . . 147
3.6.2 Temps . . . 148
3.6.3 Mémoire . . . 151

3.6.4 Interprétation . . . 153
3.7 Applications industrielles . . . 154
3.7.1 Cetaf . . . 154
3.7.2 Airbus A318 . . . 157
3.7.3 Chasseur furtif à 1 GHz . . . 159
Conclusion . . . 161

II La Méthode Multipôle Rapide : Version parallèle 163

4 Implémentation parallèle 165
Introduction . . . 165
4.1 Méthode de parallélisation . . . 165
4.1.1 Options techniques . . . 165
4.1.2 Principe de la méthode . . . 166
4.2 Répartition des cellules d'un octree . . . 166
4.2.1 Simplification du problème . . . 166
4.2.2 Numérotation d'un niveau . . . 168
4.2.3 Répartition . . . 176
4.3 Nouvelles tâches et nouvelles sous-listes . . . 177
4.3.1 Parallélisation par cellule . . . 177
4.3.2 Optimisation des échanges . . . 179
4.3.3 Parallélisation par direction . . . 185
4.4 Autres aspects de la parallélisation . . . 189
4.4.1 Fonctionnalités héritées du mode séquentiel . . . 190
4.4.2 Interactions proches . . . 190
Conclusion . . . 191

5 Performance parallèle 193
Introduction . . . 193
5.1 Présentation des architectures . . . 193
5.1.1 Station biprocesseur . . . 194
5.1.2 Ferme de PC . . . 194
5.1.3 SGI Origin3800 . . . 195
5.1.4 IBM SP3 . . . 195
5.2 Présentation des cas-tests utilisés . . . 197
5.3 Choix des paramètres . . . 197
5.3.1 Niveau plafond . . . 197
5.3.2 Niveau de séparation . . . 199
5.3.3 Niveau distribué . . . 200
5.4 Scalabilité . . . 201
5.4.1 Scalabilité numérique . . . 201
5.4.2 Scalabilité parallèle . . . 207
5.5 Calculs de grande taille . . . 213
5.5.1 Sphère à 25 millions d'inconnues . . . 213
5.5.2 Avions préconditionnés . . . 215

5.5.3 Cetaf préconditionné . . . 217
Conclusion . . . 220

III Applications de la méthode multipôle rapide **221**

6 Acoustique **223**
Introduction . . . 223
6.1 Cas d'une onde incidente . . . 223
6.1.1 Modèle physique considéré . . . 223
6.1.2 Modélisation mathématique . . . 224
6.1.3 Discrétisation . . . 226
6.1.4 Résolution numérique classique . . . 227
6.1.5 Résolution numérique multipôle . . . 228
6.1.6 Application . . . 234
6.2 Cas d'un guide d'ondes . . . 235
6.2.1 Modèle physique considéré . . . 236
6.2.2 Modélisation mathématique . . . 236
6.2.3 Discrétisation . . . 238
6.2.4 Ecriture matricielle . . . 240
6.2.5 Résolution numérique classique . . . 241
6.2.6 Résolution numérique multipôle . . . 242
6.2.7 Application . . . 245
Conclusion . . . 246

7 Autres utilisations de la FMM **247**
Introduction . . . 247
7.1 Champs proches . . . 248
7.1.1 Objectif . . . 248
7.1.2 Ecriture classique . . . 248
7.1.3 Algorithme multipôle . . . 249
7.1.4 Approfondissement . . . 253
7.2 Champs lointains . . . 255
7.2.1 Objectif . . . 255
7.2.2 Ecriture classique . . . 255
7.2.3 Algorithme multipôle . . . 256
7.3 Matériaux diélectriques . . . 257
7.3.1 Modèle physique considéré . . . 257
7.3.2 Modélisation mathématique . . . 258
7.3.3 Résolution numérique . . . 262
7.4 Objets hétérogènes . . . 265
7.4.1 Cas traité . . . 266
7.4.2 Mise en équations . . . 266
7.4.3 Aspects numériques . . . 269
7.4.4 Matériaux absorbants . . . 270
7.5 Structures filaires minces . . . 271

7.5.1 Approximation des fils minces 271
7.5.2 Discrétisation 271
7.5.3 Formulation multipôle 272
Conclusion 274

Conclusion **275**
Bibliographie 277

Introduction

Le problème auquel nous nous intéressons est la détermination du comportement électromagnétique d'une structure métallique tridimensionnelle soumise à l'action d'une onde incidente. Ce modèle s'applique à une grande variété de cas concrets, comme par exemple les calculs de furtivité radar, la conception et le placement des antennes, l'interaction entre appareils électriques, et bien d'autres encore. Dans tous les cas, le phénomène physique est le même : le champ électromagnétique incident crée des courants (magnétiques et électriques) à la surface et à l'intérieur des matériaux conducteurs rencontrés, et ces courants génèrent à leur tour un champ électromagnétique dans tout l'espace. Il y a donc un phénomène d'interaction, assez délicat à modéliser. Une première simplification consiste à se placer dans le domaine fréquentiel : on suppose que les excitations et les réponses sont toutes des ondes sinusoïdales de même fréquence, ce qui permet dans la suite du calcul d'ôter la dépendance en temps des variables. A partir de là, deux grandes familles de méthodes de résolution existent. D'une part, les méthodes volumiques localisent leur calcul dans tout le volume des objets, intérieur et extérieur, permettant une bonne prise en compte des caractéristiques des objets, mais nécessitant un grand nombre d'inconnues et une gestion explicite des conditions aux limites. D'autre part les méthodes surfaciques, qui placent leurs inconnues sur le bord des objets considérés et prennent en compte de manière implicite les conditions aux limites, mais ne s'appliquent qu'aux corps homogènes. Les équations de Maxwell et de Helmholtz peuvent s'écrire sous forme intégrale et être résolues par ce type de méthodes. C'est donc à ces dernières que nous nous intéressons.

Les équations de Maxwell harmoniques mises sous forme intégrale et discrétisées par des éléments finis de frontières conduisent à l'écriture d'un système linéaire complexe plein dont la résolution peut se faire de deux manières. Les méthodes directes factorisent la matrice – ce qui revient à l'inverser. Une fois cette opération terminée, le problème physique modélisé n'a pour ainsi dire plus de secret. Malheureusement, la factorisation en question est une opération très coûteuse, mettant en œuvre un nombre d'opérations en $\mathcal{O}(N^3)$ où N est le nombre d'inconnues. Les factorisations $L.U$ ou $L.D.L^t$ sont les méthodes directes les plus connues. Les méthodes itératives constituent l'autre famille de solveurs. Elles ont toutes en commun de chercher à s'approcher petit à petit de la solution du problème en réalisant à chaque étape un produit matrice-vecteur. Chaque itération a alors un coût en $\mathcal{O}(N^2)$, le problème étant que l'on ne connaît pas a priori le nombre d'itérations. Les méthodes itératives les plus populaires s'appellent gradient conjugué, GMRES, QMR. Les solveurs itératifs n'utilisant pas la méthode multipôle et les solveurs directs partagent la caractéristique de requérir le calcul et le stockage de la matrice du problème (qui est pleine, donc de taille N^2). En pratique, sur les machines disponibles en 2002, N ne peut guère dépasser 50,000 sur une station de travail, et 1,000,000 sur un calculateur massivement parallèle.

Or les appareils modernes exploitent des ondes électromagnétiques de fréquence de plus en

plus élevée (1,8 GHz pour les téléphones portables, autour de 30 GHz pour les applications plus pointues comme le futur satellite militaire Syracuse3). Les longueurs d'onde correspondantes décroissent, et le nombre d'inconnues N croît comme le carré de la fréquence. A 30 GHz, un calcul sur un avion de 20 mètres d'envergure nécessiterait environ $N = 10^9$ inconnues. On est bien loin des valeurs accessibles mentionnées auparavant. La méthode multipôle rapide permet de repousser ces limites. Initiée par Rokhlin et Greengard pour le problème à N corps dans les années 80 [23], adaptée par Rokhlin [34] et Chew [42] à l'électromagnétisme dans les années 90, une école française est apparue à la fin du siècle passé dans le sillage des travaux de Darve [15]. Ce mémoire s'inscrit dans ce prolongement.

Décrire la méthode multipôle rapide en quelques mots est une gageure, tant elle est dense, complexe, multiforme. Selon le point de vue adopté, nous pourrions en donner la description suivante :

Du point de vue du solveur, la FMM (pour *Fast Multipole Method*) remplace le produit matrice-vecteur standard par un calcul que nous appellerons « produit multipôle » qui réalise le même calcul de manière rapide et approchée. Ce calcul est rapide car il s'exécute en un temps en $\mathcal{O}(N \log N)$ (contre $\mathcal{O}(N^2)$ par la méthode usuelle) et s'avère compétitif dès que N dépasse quelques milliers. Il est approché car dans un usage courant, l'écart relatif entre les produits matrice-vecteur classiques et multipôle sera de l'ordre de 10^{-3}.

Du point de vue de l'électromagnétisme, la FMM découpe l'objet rayonnant en domaines, et calcule pour chacun d'eux une fonction de radiation en champ lointain. Ces fonctions servent ensuite à calculer toutes les interactions entre domaines « suffisamment » éloignés, tandis que les interactions restantes sont traitées classiquement.

Du point de vue matriciel, la FMM décompose – de manière simplifiée – chaque terme $A_{i,j}$ de la matrice A du système en une somme de termes séparant les indices i et j, permettant la réalisation rapide du produit matrice-vecteur par A en factorisant d'abord sur j, puis sur i.

Du point de vue algorithmique, enfin, la FMM parcourt un arbre basé sur un découpage récursif de l'objet diffractant. Cet aspect récursif, qui n'est pas sans rappeler celui de la transformée de Fourier rapide ou de l'algorithme de tri rapide *quicksort*, justifie le qualificatif « rapide » de la FMM car il est à l'origine de l'évaluation asymptotique en $\mathcal{O}(N \log N)$.

Nous avons implémenté la méthode multipôle rapide dans un code industriel d'électromagnétisme et d'acoustique par éléments finis de frontière de la société EADS. Nous nous sommes spécialement intéressés à la performance en cherchant à diminuer au maximum les temps d'exécution et les besoins en mémoire vive. Nous avons ainsi introduit la première FMM out-of-core, et également la première FMM à précision variable. Nous avons parallélisé cette méthode, et réalisé sur machines parallèles les calculs de ce type les plus importants jamais présentés (jusqu'à 25 millions d'inconnues). Enfin, nous avons jeté les bases d'une utilisation industrielle de la méthode multipôle en adaptant cette dernière aux besoins des utilisateurs (formulation acoustique, diélectrique, etc.).

Plan de la thèse

Ce mémoire de thèse comporte 3 parties (FMM séquentielle, FMM parallèle, applications de la FMM) et 7 chapitres. Regardons maintenant plus en détail le contenu de chacun de ces chapitres, en mettant l'accent sur les aspects originaux ou nouveaux de notre travail.

La méthode multipôle rapide : version séquentielle (p. 15)

Le chapitre 1 s'intitule **Présentation de la méthode** (p. 15). On y introduit la méthode multipôle rapide appliquée aux équations de Maxwell. Le contenu de ce chapitre se veut nouveau sur la forme plus que sur le fond. Sur la forme, nous avons tenté de faire un exposé didactique mais néanmoins rigoureux de la FMM, en introduisant petit à petit les difficultés, pour la méthode mono-niveau d'abord, puis pour la formulation multi-niveau. Sur le fond, l'écriture retenue pour la méthode multipôle est maintenant relativement standard (voir par exemple [39], [15], [8]). Soulignons toutefois que l'exposé réalisé est à la fois très complet et tout à fait au goût du jour (il inclut notamment la formulation à deux composantes de [14]). Au chapitre des nouveautés, on peut souligner une étude sur les différentes manières de découper le maillage en sous-domaines, ainsi que les comptages exhaustifs des nombres d'opérations des FMM mono-niveau et multi-niveau grâce auxquels on montre par exemple que la taille optimale des feuilles de l'octree ne dépend pas du nombre d'inconnues.

Le chapitre 2 s'intitule **Implémentation et Optimisations** (p. 79). On y présente les méthodes utilisées pour mettre en œuvre informatiquement la méthode multipôle rapide. L'idée au cœur de notre implémentation est l'utilisation d'une liste de tâches pour gérer les différentes opérations élémentaires (initialisation, montée, transfert, descente, intégration) de la FMM. L'intérêt principal de ce choix est de pouvoir réaliser de nombreuses transformations ou optimisations de la FMM de manière relativement simple, par réordonnancement ou enrichissement de cette liste. Ce chapitre s'intéresse ensuite au traitement des phases de transferts, de montées/descentes, puis aux autres. On présente des optimisations le plus souvent inédites, en analysant à chaque fois en détail les avantages et les inconvénients des différentes options possibles, là où la littérature existante se contente généralement d'un rapide survol. On présente pour finir la gestion de la mémoire dans notre code multipôle, et en particulier le mode out-of-core permettant de stocker sur disque une partie des données. On montre – graphique à l'appui – l'efficacité de cette technique. Cela fait de notre logiciel le seul code FMM out-of-core connu.

Le chapitre 3 s'intitule **Performances et Applications** (p. 121). On y présente notre code multipôle en action à travers une large variété de tests. L'ensemble des résultats de ce chapitre sont originaux. On compare tout d'abord les temps d'exécution et les résultats des codes multipôles et standards afin de vérifier la pertinence de nos travaux. On détermine ensuite tous les paramètres de la FMM (nombre de pôles, taille des feuilles, nombre de niveaux, ...). Certains des résultats obtenus ne sont valables que pour *notre* implémentation mais la méthodologie des tests reste générique. On étudie ensuite la scalabilité numérique de la méthode, qui conduit à établir (de manière inattendue et inédite) une complexité en $\mathcal{O}(N)$ pour $N \leq 1{,}2.10^6$ (où N est le nombre d'inconnues). On termine par la présentation de quelques cas de calcul industriels comportant jusqu'à un million d'inconnues mais néanmoins traités sur station de travail.

La méthode multipôle rapide, version parallèle (p. 165)

Le chapitre 4 s'intitule **Implémentation parallèle** (p. 165). On y expose les méthodes et techniques mises en œuvre pour paralléliser la FMM. Les choix que nous avons faits semblent se rapprocher de ceux faits dans les logiciels FISC [43], [44] et ScalaME [48], mais nous ne pouvons en être certains puisque les aspects techniques ne font pas l'objet de communications. La parallélisation se fait en trois temps : distribution des données, écriture « brute » des com-

munications, optimisation des communications. On introduit également une variante appelée parallélisation par direction qui permet de diminuer les communications et le coût en mémoire vive. En outre, on montre que cette parallélisation permet de conserver les améliorations introduites au chapitre 2.

Le chapitre 5 s'intitule **Performance parallèle** (p. 193). Comme dans le cas séquentiel, l'analyse des performances se fait en trois temps. Tout d'abord, on étudie les différents paramètres spécifiques à la FMM parallèle. Puis on regarde la scalabilité numérique et parallèle du code, c'est-à-dire son comportement lorsque respectivement le nombre d'inconnues et de processeurs augmente. En particulier, on retrouve une complexité en $\mathcal{O}(N \log N)$ conforme à la théorie pour $N \leq 2{,}5.10^7$. On termine par la présentation de cas tests de très grande taille, qui sont à ce jour les calculs de ce type mettant en œuvre les plus grands nombres de degrés de liberté.

Applications de la méthode multipôle rapide (p. 223)

Le chapitre 6 s'intitule **Acoustique** (p. 223). On s'intéresse à la résolution de l'équation de Helmholtz par formulation intégrale pour les problèmes d'acoustique dans des fluides parfaits. La similitude du noyau de Green permet d'utiliser la méthode multipôle rapide pour accélérer l'utilisation de solveurs itératifs. Dans un premier temps, on traite le cas simple d'un objet rigide et/ou traité soumis à une onde acoustique plane incidente. On définit les termes adéquats de la FMM, puis on vérifie la validité de nos développements grâce à quelques tests. Dans un deuxième temps, on étudie une formulation plus élaborée basée sur une nacelle de moteur d'avion, formulation dans laquelle l'intérieur de l'objet est considéré comme un guide d'ondes interagissant avec l'extérieur. Le but est ici de prouver que la FMM peut parfaitement s'adapter à des résolutions complexes, allant au-delà de la simple résolution d'un système linéaire. Tous les développements de ce chapitre sont nouveaux.

Le chapitre 7 s'intitule **Autres utilisations de la FMM** (p. 247). On revient ici au cas électromagnétique, et l'on s'intéresse à tous les autres modèles physiques et post-traitements gérés par les formulations intégrales classiques : matériaux diélectriques et absorbants, objets hétérogènes, fils, calculs des champs proches et lointains. Les nouvelles écritures multipôles associées sont toutes basées sur le même noyau de Green, et donc sur le même cœur FMM. Cette partie est exclusivement théorique, les formules introduites (nouvelles pour la plupart) n'ayant pas encore été implémentées.

Première partie

La Méthode Multipôle Rapide : Version séquentielle

Chapitre 1

Présentation de la méthode

Sommaire

Introduction **15**
1.1 Technique de résolution **16**
1.2 Méthode mono-niveau **21**
1.3 Méthode multi-niveau **45**
Conclusion **76**

Introduction

On se propose de calculer l'interaction d'une onde plane électromagnétique monochromatique et d'un objet métallique parfaitement conducteur en 3D. Ce type de problème surgit, par exemple, lors de la conception d'antennes ou d'engins furtifs. La diffraction d'une onde électromagnétique par un obstacle est régie par les équations de Maxwell [45], [25]. Il existe numériquement de nombreuses formulations mathématiques permettant de traiter cette expérience physique [12]. Les méthodes dites exactes qui résolvent ces équations directement se partagent entre *les méthodes volumiques* (qui travaillent dans le domaine de propagation) et *les méthodes intégrales* [29] qui ramènent le problème à une équation sur chaque interface du domaine (*i.e.* la surface de l'objet lorsque celui-ci est homogène). On s'intéresse ici à une formulation intégrale des équations de Maxwell harmoniques résolue par éléments finis de frontière.

Cette méthode date du début des années 80 [3]. Par rapport à une formulation classique volumique de type différences finies ou éléments finis, elle présente un certain nombre d'avantages. D'une part, seule la surface des objets considérés est maillée. Cela diminue le nombre d'inconnues, et permet de travailler sur un maillage bidimensionnel. D'autre part, le traitement des conditions aux limites est intégré à la formulation, il n'est donc pas nécessaire de mailler le vide autour de l'objet, ni même une frontière fictive simulant l'infini.

Cependant, les équations intégrales conduisent à résoudre un système linéaire complexe plein. Ce type de résolution peut se faire de deux manières. La première approche consiste à utiliser un solveur direct, ce qui revient à assembler et inverser la matrice du problème. C'est un processus robuste, précis (lorsque les questions de conditionnement sont traitées correctement), mais très coûteux, tant en mémoire qu'en temps de calcul : le temps de résolution est multiplié

par mille chaque fois que le nombre d'inconnues est multiplié par dix. Dès lors, ce type de solveur est inutilisable au-delà de 10^4 inconnues.

La seconde approche utilise un solveur itératif : les plus courants s'appellent GMRES [37], gradient conjugué [26], QMR [20]. Ils ont tous en commun le fait d'utiliser des produits matrice-vecteur pour se rapprocher progressivement de la solution recherchée. Cette deuxième approche est certes moins robuste (car le nombre de produits matrice-vecteur à réaliser pour converger n'est pas connu à l'avance), mais elle est déjà nettement plus économique en temps et en mémoire. Néanmoins, lorsque l'on cherche à traiter des cas avec de grands nombres d'inconnues (de l'ordre de 10^5 et plus), les méthodes itératives classiques atteignent leurs limites. Les produits matrice-vecteur et l'assemblage de la matrice deviennent longs, et rendent indispensable l'utilisation de calculateurs parallèles pour résoudre ce type d'équations.

La méthode multipôle rapide (*Fast Multipole Method* ou *FMM*) permet de repousser ces limitations (voir par exemple [41], [13]). Elle réalise les produits matrice-vecteur de manière beaucoup plus rapide, tout en diminuant considérablement le besoin en mémoire. En particulier, la matrice complète n'est plus assemblée. En contrepartie, les produits matrice-vecteur ainsi réalisés ne sont plus exacts mais approchés. Par ailleurs, la complexité des algorithmes pour ce nouveau type de produit est assez élevée.

Nous nous proposons dans la suite de présenter de manière claire et synthétique les principes de la méthode multipôle rapide. Dans un premier temps, nous verrons le problème physique considéré, les équations intégrales associées à ce cas, et les différentes méthodes de résolution existantes. Nous présenterons ensuite la FMM dans sa version la plus simple dite « mono-niveau », puis dans sa version complète dite « multi-niveau ».

1.1 Technique de résolution

1.1.1 Modèle physique considéré

On se donne un objet Ω parfaitement conducteur de frontière Γ se trouvant dans le vide. Une onde électromagnétique incidente $\vec{E}_{inc}$ de fréquence f et de pulsation $\omega = 2\pi f$ illumine l'objet. On prend pour inconnues les champs diffractés $\vec{E}_{diff}$ et $\vec{H}_{diff}$ à l'extérieur de Ω , prolongés à l'intérieur par l'opposé des champs incidents $-\vec{E}_{inc}$ et $-\vec{H}_{inc}$ (de sorte que le champ total y est nul). On note $\vec{\nu}$ la normale unitaire *sortante* en tout point de Γ.

1.1.2 Modélisation mathématique

1.1.2.1 Equations de Maxwell harmoniques

On travaille dans le domaine fréquentiel avec une dépendance implicite en temps en $e^{-i\omega t}$. Le problème harmonique de Maxwell extérieur s'écrit :

$$\begin{cases} \vec{rot}\vec{E} - i\omega\mu_0\vec{H} = 0 & dans\ \mathbb{R}^3 - \Omega, \\ \vec{rot}\vec{H} + i\omega\epsilon_0\vec{E} = 0 & dans\ \mathbb{R}^3 - \Omega, \\ \vec{E} \wedge \vec{\nu}|_\Gamma = -\vec{E}_{inc} \wedge \vec{\nu}|_\Gamma & sur\ \Gamma, \\ \lim\limits_{r\to+\infty} r \left| \sqrt{\epsilon_0}\vec{E} - \sqrt{\mu_0}\vec{H} \wedge \dfrac{\vec{r}}{r} \right| = 0 \end{cases}$$

où ϵ_0 est la permittivité du vide et μ_0 la perméabilité du vide. On note en outre $Z_0 = \sqrt{\mu_0/\epsilon_0}$ l'impédance du vide et $c = 1/\sqrt{\mu_0\epsilon_0}$ la vitesse de propagation des ondes électromagnétiques

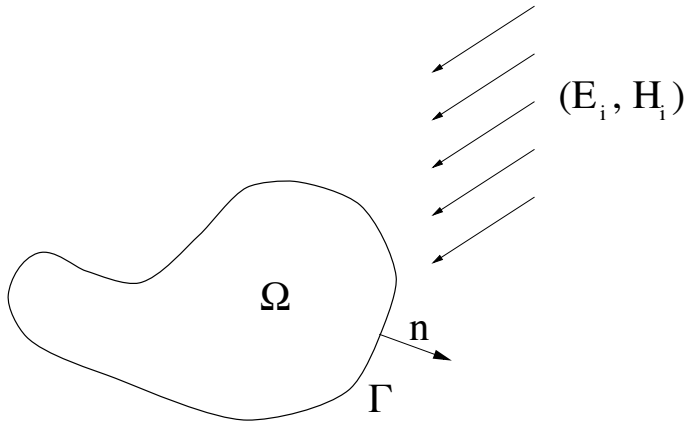


FIG. 1.1 – *Problème traité*

dans le vide.

1.1.2.2 Représentation intégrale

Les résultats de cette section sont issus de [29], auquel on renvoie le lecteur pour plus de détails, ainsi qu'à la section 7.3. On note $\vec{j}$ et $\vec{m}$ les traces tangentielles du champ total sur Γ :

$$\begin{cases} \vec{j} = \vec{\nu} \wedge \vec{H}_{tot}, \\ \vec{m} = \vec{\nu} \wedge \vec{E}_{tot} \end{cases}$$

Dans le cas d'un matériau parfaitement conducteur, le champ $\vec{m}$ est nul. Le courant électrique surfacique $\vec{j}$ représente le saut de la composante tangentielle de $\vec{H}$ au franchissement de l'interface Γ. Physiquement, c'est le courant de conduction circulant dans une épaisseur de peau tendant vers zéro. $\vec{j}$ s'exprime en A/m. $\vec{j}$ est la véritable inconnue du problème. En effet, connaissant $\vec{j}$, on peut grâce au théorème de représentation des équations de Maxwell en déduire la valeur des champs diffractés $\vec{E}$ et $\vec{H}$ en tout point de l'espace hors Γ :

$$\begin{cases} \vec{E}(y) = i\omega\mu_0 \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx + \frac{i}{\omega\epsilon_0} \vec{grad}_y \int_\Gamma G(|y-x|) div_\Gamma \vec{j}(x)dx & y \in \mathbb{R}^3 - \Omega, \\ \vec{H}(y) = -\vec{rot}_y \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx & y \in \mathbb{R}^3 - \Omega \end{cases}$$

où

$$G(R) = \frac{e^{ikR}}{4\pi R}$$

est la fonction de Green solution élémentaire de l'équation de Helmholtz en 3D

$$\Delta u + k^2 u = -\delta_0$$

associée à la condition de radiation sortante

$$\lim_{r \to +\infty} r \left(\frac{\partial u}{\partial r} - iku \right) = 0$$

Il découle de cette représentation une formulation variationnelle du problème traité : on cherche le courant surfacique $\vec{j}$ tel que pour tout courant surfacique $\vec{j^t}$ tangent à Γ, on ait :

$$\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{j}(x).\vec{j^t}(y) - \frac{1}{k^2} div_\Gamma \vec{j}(x).div_\Gamma \vec{j^t}(y) \right) dxdy = \frac{i}{kZ_0} \int_\Gamma \vec{E}_{inc}(x).\vec{j^t}(x)dx \tag{1.1}$$

Cette écriture découle du *principe de réaction de Rumsey* [35]. Dans la mesure où elle est issue du théorème de représentation du champ électrique $\vec{E}$, le plus souvent on la désigne sous le nom d'*EFIE* pour *Electric Field Integral Equation*.

De la même manière, on a la *MFIE* pour *Magnetic Field Integral Equation* qui découle de la représentation du champ magnétique $\vec{H}$:

$$\frac{1}{2} \int_\Gamma \vec{j}(y).\vec{j^t}(y)dy + \int_\Gamma \vec{j^t}(y).\vec{\nu}(y) \wedge \left(\int_\Gamma \vec{grad}_x G(|y-x|) \wedge \vec{j}(x)dx \right) dy = \int_\Gamma \vec{j^t}(y).\vec{\nu}(y) \wedge \vec{H}_{inc}(u)dy$$

Enfin, afin d'éliminer les problèmes de résonance liés à l'existence de valeurs propres pour le problème intérieur, il est usuel de considérer la formulation *CFIE* pour *Combined Field Integral Equation*, qui est en fait une combinaison linéaire d'*EFIE* et de *MFIE* :

$$CFIE = \alpha EFIE + (1-\alpha)\frac{i}{k} MFIE$$

Le choix $\alpha = 0{,}2$ donne généralement de bons résultats. Notons que l'*EFIE* est symétrique en $\vec{j}$ et $\vec{j^t}$ contrairement aux deux autres formulations. Pour la suite de cette partie, nous nous restreindrons à l'équation *EFIE*, sauf mention spécifique.

1.1.3 Discrétisation

On utilise la discrétisation standard pour ce type de problème, à savoir celle de Raviart-Thomas ([33]), redécouverte dans le contexte qui nous intéresse ici quelques années plus tard par Rao-Wilton-Glisson ([32]). On remplace tout d'abord la surface réelle par une surface polyédrique composée de triangles formant une triangulation. A chaque arête A_i sont associés un degré de liberté λ_i et une fonction de base $\vec{\varphi}_i$. On note n_{dl} le nombre d'arêtes et de degrés de liberté. Le support de $\vec{\varphi}_i$ se restreint aux deux triangles $\mathcal{T}$ et $\mathcal{T}'$ partageant l'arête A_i. Chaque arête est (arbitrairement) orientée par un « sens » de passage, noté par exemple $\mathcal{T} \to \mathcal{T}'$, qui détermine la normale unitaire à l'arête $\vec{n}_i$. Le degré de liberté λ_i est égal au flux de la fonction de courant surfacique (tangentiel) $\vec{j}$ à travers l'arête A_i ainsi orientée :

$$\lambda_i = \int_{A_i} \vec{j}(M).\vec{n}_i(M)dM$$

Sur le triangle $\mathcal{T}$, on note S le sommet opposé à A_i. On a :

$$\vec{\varphi}_i(M) = \frac{\overrightarrow{SM}}{2.|\mathcal{T}|}$$

La fonction $\vec{\varphi}_i$ est donc vectorielle, dans le plan du triangle, affine, de flux nul à travers les quatre arêtes issues des sommets S et S' ($\vec{\varphi}_i$ y est tangente) et de flux 1 à travers l'arête A_i (grâce à la normalisation par $2.|\mathcal{T}|$). La figure 1.2 illustre ces propriétés.

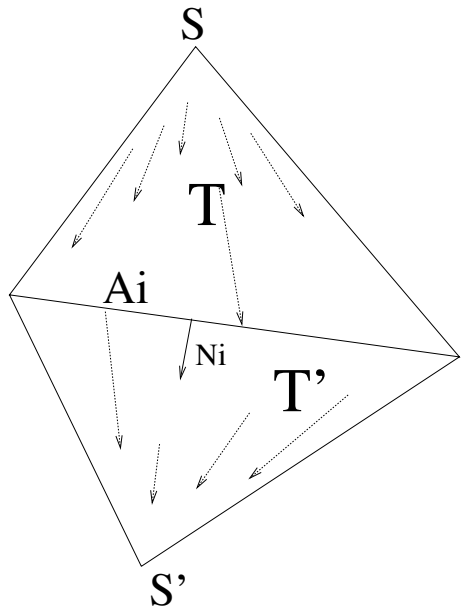


FIG. 1.2 – *La fonction de base $\vec{\varphi}_i$ sur le triangle $\mathcal{T}$*

Symétriquement sur $\mathcal{T}'$, on note S' le sommet opposé à A_i. On a :

$$\vec{\varphi}_i(M) = -\frac{\overrightarrow{S'M}}{2.|\mathcal{T}'|}$$

Le flux de $\vec{\varphi}_i$ à travers l'arête A_i vaut toujours $+1$ (la traversée de l'arête reste orientée par $\vec{n}_i$). La fonction $\vec{j}$ sera recherchée dans l'espace d'élément fini sous la forme :

$$\vec{j} = \sum_{1 \leq i \leq n_{dl}} \lambda_i \vec{\varphi}_i$$

1.1.4 Résolution numérique

1.1.4.1 Ecriture matricielle

On reprend la formulation intégrale (1.1). On cherche donc $\vec{j}$ telle que pour toute fonction-test $\vec{j}^t$ on ait :

$$\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{j}(x).\vec{j}^t(y) - \frac{1}{k^2} div_\Gamma \vec{j}(x).div_\Gamma \vec{j}^t(y) \right) dxdy$$
$$= \frac{i}{kZ_0} \int_\Gamma \vec{E}_{inc}(x).\vec{j}^t(x)dx$$

Dans cette équation, on cherche $\vec{j}$ sous la forme $\sum_{1 \leq i \leq n_{dl}} \lambda_i \vec{\varphi}_i$, et on prend pour fonction test une des fonctions de base $\vec{j}^t = \vec{\varphi}_j$. Notons que ce choix est le plus courant mais ce n'est

pas le seul possible, on pourrait également prendre $\vec{j}^t = \vec{\varphi}_j \wedge \vec{\nu}$ où $\vec{\nu}$ est la normale sortante (voir [39]). On cherche le vecteur $(\lambda_i)_{1\leq i\leq n_{dl}}$ tel que pour tout $1 \leq j \leq n_{dl}$ on ait :

$$\sum_{1\leq i\leq n_{dl}} \lambda_i \left[\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{\varphi}_i(x).\vec{\varphi}_j(y) - \frac{1}{k^2} div_\Gamma \vec{\varphi}_i(x).div_\Gamma \vec{\varphi}_j(y) \right) dxdy \right] = \frac{i}{kZ_0} \int_\Gamma \vec{E}_{inc}(x).\vec{\varphi}_j(x)dx$$

On est finalement amené à résoudre un système linéaire de n_{dl} équations à n_{dl} inconnues de la forme

$$A.\lambda = b$$

avec les termes matriciels et vectoriels suivants :

$$\begin{cases} A_{i,j} = \displaystyle\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{\varphi}_i(x).\vec{\varphi}_j(y) - \frac{1}{k^2} div_\Gamma \vec{\varphi}_i(x).div_\Gamma \vec{\varphi}_j(y) \right) dxdy, \\ b_j = \dfrac{i}{kZ_0} \displaystyle\int_\Gamma \vec{E}_{inc}(x).\vec{\varphi}_j(x)dx \end{cases}$$

On retrouve le fait que la formulation *EFIE* est symétrique (non hermitienne) : $A_{i,j} = A_{j,i}$. Notons également que les intégrales doubles dans la définition de $A_{i,j}$ se restreignent au support de $\vec{\varphi}_i$ pour la variable x, et à celui de $\vec{\varphi}_j$ pour la variable y, soit deux triangles à chaque fois. Nous ne rentrerons pas dans le détail du calcul de ces coefficients. Signalons simplement que la principale difficulté émerge de la singularité du noyau de Green $G(R) = \frac{e^{ikR}}{4\pi R}$ en $R = 0$ c'est-à-dire lorsque $x = y$. Si les fonctions $\vec{\varphi}_i$ et $\vec{\varphi}_j$ ont des supports « éloignés » (en un sens qu'il convient à l'utilisateur de définir), l'intégrale dans $A_{i,j}$ est calculée de manière numérique à l'aide de points d'intégration de Gauss. Si les supports sont proches, on procède de manière semi-analytique en utilisant des points de Gauss pour l'une des intégrations, et des formules exactes pour l'autre. Les calculs complets peuvent être consultés dans [3] et [15].

1.1.4.2 Solveurs directs

Il existe deux grandes classes de solveurs permettant de résoudre ce type de système linéaire : les solveurs directs et les solveurs itératifs. Les solveurs directs suivent un processus qui aboutit de manière certaine à la solution au bout d'un nombre d'opérations connu à l'avance. Souvent, ce type de solveur est composé d'une phase de factorisation de la matrice ($L.U$, $L.D.^tL$ par exemple) suivie d'une phase de résolution effective du problème.

Les avantages d'un tel solveur sont :

- Sa robustesse : les besoins en puissance de calcul et en mémoire sont connus à l'avance avec précision ;
- Sa précision : si les problèmes de conditionnement ont été bien traité, le système linéaire est résolu avec la précision machine (proche de 10^{-16} le plus souvent en arithmétique double précision).
- Sa capacité multi-second membre : une fois la matrice factorisée, un grand nombre de seconds membres peut être traité pour un surcoût en temps relativement faible.

Malgré ses avantages, ce type de solveur est rapidement inutilisable dans un contexte industriel. En effet, les inconvénients d'un tel solveur sont :

- Son coût mémoire : la factorisation nécessite le calcul et le stockage de la matrice A soit $n_{dl}^2/2$ scalaires dans le cas symétrique. Sur une station de travail normale (avec 30 Go de stockage disque), on est limité à 60.000 inconnues en double précision (ce qui est peu compte tenu des besoins des industriels).
- Son temps d'exécution : le nombre d'opérations pour une factorisation $L.D.^tL$ est $2/3.n_{dl}^3$. En 24 heures sur une station de travail normale (puissance effective de 100 Mflops), on est limité à 20.000 inconnues environ.

De ces deux limitations, on voit que la seconde est la plus contraignante, car elle est proportionnelle à n_{dl}^3. Les solveurs itératifs permettent de faire sauter ce verrou.

1.1.4.3 Solveurs itératifs

Les solveurs itératifs suivent un processus répétitif s'approchant peu à peu de la solution du système linéaire. C'est l'utilisateur qui fixe le seuil à partir duquel la solution sera considérée comme satisfaisante. A chaque étape de cette progression, le solveur réalise un produit matrice-vecteur. La matrice du problème n'intervient plus que par le biais de ces produits. Si on note N_{iter} le nombre d'itérations nécessaire à la convergence du solveur, le nombre total d'opérations sera de l'ordre de $N_{iter}.n_{dl}^2$. Les solveurs itératifs que nous avons le plus utilisés sont GMRES ([37], [10]) et TF-QMR [20] (une variante de QMR [19] qui a l'avantage de ne pas réclamer de transposition).

Par rapport à un solveur direct, la performance d'un solveur itératif est beaucoup plus incertaine. Elle va dépendre essentiellement du nombre d'itérations N_{iter}, lequel va dépendre du conditionnement de la matrice A. La plupart des solveurs itératifs garantissent la convergence de leurs calculs en n_{dl} itérations au plus. Bien sûr, si $N_{iter} = n_{dl}$, on retrouve un nombre total d'opérations proportionnel à n_{dl}^3, ce qui ne présente aucun intérêt. Heureusement, le plus souvent N_{iter} reste très inférieur à n_{dl}, voire même ne dépend pas de n_{dl}. Si tel est le cas, on obtient un coût mémoire et un temps d'exécution proportionnels à n_{dl}^2.

Il y a donc déjà une amélioration par rapport à un solveur direct. Néanmoins, il est possible de faire beaucoup mieux. Dans la mesure où la matrice A n'intervient que via des produits matrice-vecteur, il serait intéressant de pouvoir réaliser ces produits sans recourir à l'assemblage complet de A. C'est précisément ce que propose la méthode multipôle rapide.

1.2 Méthode mono-niveau

1.2.1 Premier survol

Le but de cette partie est de présenter rapidement, sans détailler, et de manière si possible didactique la méthode multipôle rapide dans sa version simplifiée à un niveau. Les éléments présentés ici doivent suffire à implémenter une première méthode multipôle mono-niveau.

1.2.1.1 Principe de base

La méthode multipôle rapide permet de réaliser de manière économique des produits matrice-vecteur. On se donne donc un vecteur $(t_i)_{1\leq i\leq n}$ représentant la fonction $\vec{t}(x) = \sum_{1\leq i\leq n} t_i.\vec{\varphi}_i(x)$, et on cherche à calculer le produit $A.t$ dont la j-ième coordonnée s'écrit :

$$(A.t)_j = \int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{t}(x).\vec{\varphi}_j(y) - \frac{1}{k^2} div_\Gamma \vec{t}(x).div_\Gamma \vec{\varphi}_j(y) \right) dxdy \tag{1.2}$$

L'idée de base de la méthode multipôle est de tenter de séparer les variables x et y afin de pouvoir séparer les deux intégrales. Pour cela, il nous faut réécrire le noyau de Green différemment. Cette réécriture déterminera la forme que prendront les fonctions manipulées par la FMM.

1.2.1.2 Simplification des termes matriciels

Avant toute chose, nous allons tâcher de simplifier la forme des produits matrice-vecteur à calculer. Dans la formule (1.2), le terme central peut se réécrire sous la forme :

$$\begin{aligned} \vec{t}(x).\vec{\varphi}_j(y) - 1/k^2 div_\Gamma \vec{t}(x).div_\Gamma \vec{\varphi}_j(y) = & (\vec{t})_x(x).(\vec{\varphi}_j)_x(y) \\ & + (\vec{t})_y(x).(\vec{\varphi}_j)_y(y) \\ & + (\vec{t})_z(x).(\vec{\varphi}_j)_z(y) \\ & - 1/k^2 div_\Gamma \vec{t}(x).div_\Gamma \vec{\varphi}_j(y) \end{aligned}$$

L'expression (1.2) apparaît donc comme étant la somme de quatre termes de la forme :

$$\int_\Gamma \int_\Gamma G(|y-x|) f(x) g(y) dxdy \tag{1.3}$$

où les fonctions scalaires f et g sont respectivement les composantes x, y, z et divergence des fonctions $\vec{t}$ et $\vec{\varphi}_j$. Il est donc équivalent de manipuler des termes de la forme (1.2) ou (1.3). Dans la suite de cette partie, on se contentera donc de manipuler des fonctions scalaires comme dans (1.3).

Le produit matrice-vecteur d'origine se ramène donc à quatre produits matrice-vecteur scalaires. Nous verrons plus loin qu'il est possible de ramener ce nombre à trois, voire à deux.

1.2.1.3 Décomposition du noyau

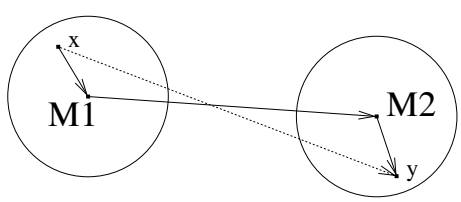


FIG. 1.3 – *Configuration type*

On se donne quatre points x, M_1, M_2 et y. Pour fixer les idées, on suppose que l'on est dans la configuration représentée figure 1.3 : x est proche de M_1, y est proche de M_2, M_1 et M_2 sont éloignés. On précisera ultérieurement le sens précis de ces assertions. Le vecteur $\overrightarrow{xy}$ se décompose bien sûr sous la forme :

$$\overrightarrow{xy} = \overrightarrow{xM_1} + \overrightarrow{M_1M_2} + \overrightarrow{M_2y}$$

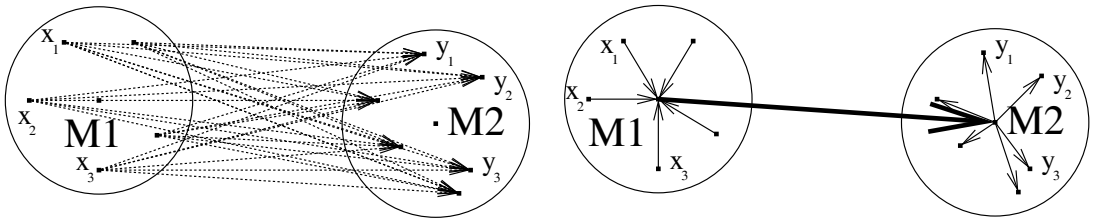
On souhaiterait donc décomposer le noyau de Green $G(|y-x|)$ de la même manière. Le théorème d'addition de Gegenbauer [1] permet de faire cela. Précisons tout d'abord quelques notations : on désigne par $\mathcal{S}$ la sphère unité de $\mathbb{R}^3$, par $\vec{s}$ un point générique de $\mathcal{S}$, par P_l le polynôme de Legendre de rang l, et par $h_l^{(1)}$ la fonction de Hankel sphérique du premier type de rang l. On a alors la décomposition suivante pour le noyau de Green :

$$G(|y-x|) = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}_1} T^L_{M_1\vec{M}_2}(\vec{s}) e^{ik\vec{s}.\vec{M_2}y} d\vec{s} \tag{1.4}$$

où

$$T^L_{M_1\vec{M}_2}(\vec{s}) = \sum_{0\leq l\leq L} (2l+1) i^l h_l^{(1)}(k.|M_1\vec{M}_2|) P_l(cos(\vec{s},M_1\vec{M}_2)) \tag{1.5}$$

Tâchons d'interpréter la formule (1.4). Elle comporte trois termes : le terme $e^{ik\vec{s}.x\vec{M}_1}$ transporte l'information du point source x au point M_1. Le terme $T_{M_1\vec{M}_2}(\vec{s})$ assure le transfert de l'information entre M_1 et M_2. Enfin, le terme $e^{ik\vec{s}.\vec{M_2}y}$ transporte l'information jusqu'au point destination y. On voit que dans cette formule, les variables x et y sont bien séparées. La fonction (1.5) s'appelle *fonction de transfert*.



(a) Sans la FMM (b) Avec la FMM

FIG. 1.4 – *Traitement des interactions*

L'intérêt de cette décomposition du noyau de Green est illustré sur la figure 1.4. Dans le cas où un ensemble de points x_i proches de M_1 agit sur un autre ensemble de points y_j proches de M_2, la méthode classique (figure 1.4a) génère un grand nombre d'interactions, tandis que la méthode multipôle (figure 1.4b) centralise les informations en M_1 et M_2 et génère ainsi beaucoup moins de calculs.

On voit apparaître ici deux difficultés : d'une part l'intégrale sur $\mathcal{S}$ dans (1.4) va devoir être discrétisée, d'autre part le nombre de termes de la somme (1.5) va devoir être fixé, et ces deux approximations devront être réalisées conjointement.

1.2.1.4 Découpage en domaine

Afin de retrouver les points x et y de l'équation (1.3) dans une configuration proche de celle de la figure 1.3, on va procéder au découpage de la surface de l'objet traité Γ en sous-domaines de tailles homogènes. Il existe pour cela une infinité de méthodes possibles, on en a choisi une qui est à la fois simple et systématique : on conçoit une grille 3D cubique de pas a englobant Ω (figure 1.5), chaque intersection non-vide d'un cube de la grille et de la surface Γ constitue un sous-domaine de notre découpage.

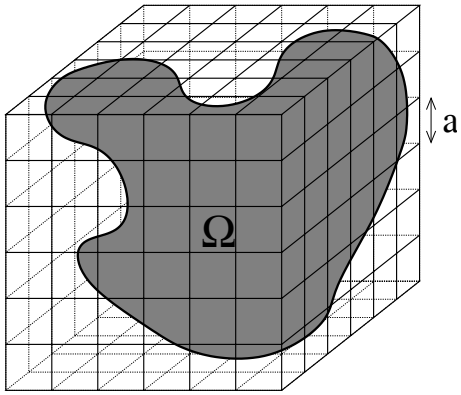


FIG. 1.5 – *Découpage de* Γ *avec une grille 3D d'arête a*

Le découpage obtenu sur un objet plus réaliste est représenté sur la figure 1.6. Le maillage utilisé est celui d'un airbus A318 d'envergure 15 longueurs d'onde, comportant 23676 inconnues. Le découpage est constitué de 584 boîtes d'arête égale à une demi longueur d'onde.

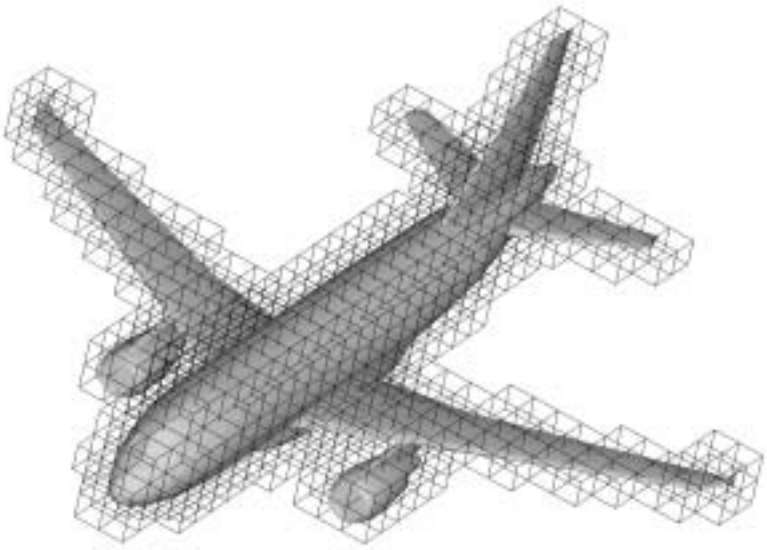


FIG. 1.6 – *Découpage d'un airbus A318 avec une grille 3D*

L'équivalent 2D de ce partitionnement est représenté sur la figure 1.7. Les cellules ayant une intersection non vide avec Γ sont grisées. On note $\mathcal{C}$ les cellules ainsi découpées, et M le centre de $\mathcal{C}$.

1.2.1.5 Interaction entre deux sous-domaines

On se donne deux cellules $\mathcal{C}$ et $\mathcal{C}'$ de notre grille, de centres respectifs M et M', et on cherche à calculer le terme d'interaction entre deux sous-domaines $\Gamma \cap \mathcal{C}$ et $\Gamma \cap \mathcal{C}'$, à savoir :

$$\int_{x \in \Gamma \cap \mathcal{C}} \int_{y \in \Gamma \cap \mathcal{C}'} G(|y-x|) f(x) g(y) dx dy \tag{1.6}$$

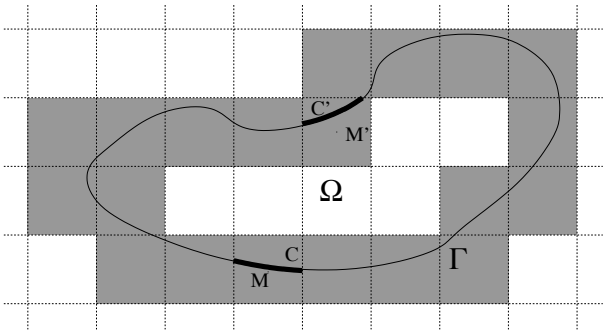


FIG. 1.7 – *Découpage de* Γ *en sous-domaine (version 2D)*

En utilisant la décomposition du noyau (1.4), en mettant de côté la constante $ik/16\pi^2$, et en supposant L fixé, ce terme peut s'écrire :

$$\int_{x\in\Gamma\cap\mathcal{C}}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}}T^L_{M\vec{M}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}f(x)g(y)dxdy$$

Réordonnons les intégrales :

$$\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\left[T^L_{M\vec{M}'}(\vec{s})\left(\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}f(x)dx\right)\right]e^{ik\vec{s}.\vec{M'}y}g(y)d\vec{s}dy \tag{1.7}$$

On voit apparaître trois phases dans le calcul de cette formule :

Initialisation : on calcule la fonction $\mathcal{F}_\mathcal{C}$ définie sur la sphère unité $\mathcal{S}$ par :

$$\mathcal{F}_\mathcal{C}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}f(x)dx \tag{1.8}$$

$\mathcal{F}_\mathcal{C}$ ne dépend que du courant f, de la cellule $\mathcal{C}$ et de son centre M. Elle représente l'influence du domaine $\Gamma\cap\mathcal{C}$ sur l'extérieur. $\mathcal{F}_\mathcal{C}$ sera parfois appelée « fonction d'émission » de $\mathcal{C}$. D'une manière générale, les fonctions définies sur $\mathcal{S}$ comme $\mathcal{F}_\mathcal{C}$ seront appelées « fonctions de radiation ».

Transfert : on multiplie la fonction $\mathcal{F}_\mathcal{C}$ par la fonction de transfert $T^L_{M\vec{M}'}$. Le produit résultant est toujours une fonction définie sur $\mathcal{S}$, elle représente l'action des courants f portés par $\Gamma\cap\mathcal{C}$ au point M' de l'espace.

Intégration : on termine le calcul en intégrant le résultat du transfert à la fois sur $\mathcal{S}$ et sur $\Gamma\cap\mathcal{C}'$:

$$\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\left[T^L_{M\vec{M}'}(\vec{s})\mathcal{F}_\mathcal{C}(\vec{s})\right]e^{ik\vec{s}.\vec{M'}y}g(y)d\vec{s}dy$$

1.2.1.6 Troncature et discrétisation

1.2.1.6.1 Troncature de la fonction de transfert On va tenter d'utiliser la formule de décomposition du noyau (1.4) dans la configuration de la figure 1.7. Pour cela, il est tout d'abord nécessaire de choisir L dans la somme définissant la fonction de transfert (1.5).

On a donc deux cellules $\mathcal{C}$ de centre M, $\mathcal{C}'$ de centre M' et deux points $x\in\mathcal{C}$ et $y\in\mathcal{C}'$. On note :

$$\begin{cases} \vec{r_0} &= \vec{MM'}, \\ \vec{r} &= \vec{xy} - \vec{MM'} = \vec{xM} + \vec{M'y} \end{cases} \tag{1.9}$$

On a donc $\vec{xy} = \vec{r} + \vec{r_0}$. Le vecteur $\vec{xy}$ se décompose comme la somme du vecteur $\vec{r_0}$ reliant les centres des cellules et du vecteur $\vec{r}$ constituant le « reliquat ». On cherche à calculer le noyau de Green $G(|\vec{r} + \vec{r_0}|)$ à partir de la fonction de transfert $T^L_{\vec{r_0}}$. Nous donnons ici un résultat simplifié (tiré de [15]) mais correct en première approximation, que nous détaillerons ultérieurement. Sous la condition :

$$\frac{|\vec{r}|}{|\vec{r_0}|} \leq \frac{2}{\sqrt{5}} \tag{1.10}$$

on peut se contenter de prendre $L = k|\vec{r}|$ termes dans la somme de la série (1.5) pour obtenir la convergence (La notation $L = k|\vec{r}|$ est un raccourci pour $L = \lfloor k|\vec{r}| \rfloor$ où $\lfloor x \rfloor$ désigne la partie entière de x). Dans la pratique, cette valeur de L s'avère convenable pour des valeurs de a supérieure à 2λ, mais conduit à une *FMM* peu précise en dessous. Pour une méthode précise à 10^{-3} (i.e. un écart relatif de 10^{-3} entre les produits matrice-vecteur classiques et multipôles), on peut prendre les valeurs présentées dans la table 1.1.

a	L
$\lambda/4$	8
$\lambda/2$	12
λ	20
2λ	32

TAB. 1.1 – *Suggestions de valeurs pour a et L*

Dans le cas d'un découpage de Γ par une grille cubique 3D d'arête a, on a :

$$|\vec{r}| \leq \sqrt{3}a \tag{1.11}$$

donc (1.10) sera vérifiée dès que :

$$|\vec{r_0}| \geq \frac{\sqrt{5}}{2}.\sqrt{3}a \approx 1{,}9a$$

Cette condition exclut toutes les cellules $\mathcal{C}'$ ayant une face, une arête ou un sommet en commun avec $\mathcal{C}$, puisqu'on a alors $|\vec{r_0}|/a$ qui vaut 1 dans le premier cas, $\sqrt{2}$ dans le second et $\sqrt{3}$ dans le troisième. Bien sûr $\mathcal{C}' = \mathcal{C}$ est également exclu.

On qualifiera désormais de *voisines* deux cellules ayant au moins un sommet commun. On vient donc de voir que :

- Si $\mathcal{C}$ et $\mathcal{C}'$ ne sont pas voisines, la série (1.5) peut être tronquée au rang $L = k|\vec{r}|$. On peut alors calculer l'interaction entre $\Gamma \cap \mathcal{C}$ et $\Gamma \cap \mathcal{C}'$ à partir de (1.7).
- Si $\mathcal{C}$ et $\mathcal{C}'$ sont voisines, on ne peut pas tronquer la fonction de transfert, on est donc obligé de calculer le terme (1.6) classiquement (cf. section 1.1.4.1).

On notera par la suite $v(\mathcal{C})$ l'ensemble des cellules voisines de $\mathcal{C}$.

1.2.1.6.2 Discrétisation de la sphère unité L étant désormais fixé, on a besoin de calculer l'intégrale de surface sur $\mathcal{S}$ définie par :

$$\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}(\vec{s}) d\vec{s}$$

Les fonctions $\mathcal{L}^2$ de $\mathcal{S}$ ont une base naturelle qui est celle des harmoniques sphériques, notée :

$$(Y_{l,m})_{l\geq 0,\ -l\leq m\leq l}$$

Pour une présentation complète des harmoniques sphériques et de leur nombreuses propriétés, on pourra se référer à [29]. La fonction $T^L_{\vec{r_0}}$ appartient à l'espace engendré par les harmoniques de rang $l \leq L$: on dit qu'elle est de largeur de bande L.

Or le terme $e^{ik\vec{s}.\vec{r}}$ se développe en série :

$$e^{ik\vec{s}.\vec{r}} = \sum_{l\geq 0} (2l+1) i^l j_l^{(1)}(k.|\vec{r}|) P_l(cos(\vec{s},\vec{r}))$$

De la même manière que l'on arrête la somme de la fonction de transfert $T^L_{\vec{r_0}}$ au rang L, on démontre que la série $e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}$ peut être tronquée au rang L avec une erreur du même ordre. On déduit du résultat précédent que la fonction intégrée $T^L_{\vec{r_0}} e^{ik\vec{s}.\vec{r}}$ est de largeur de bande $2L$, c'est à dire qu'elle peut s'écrire :

$$e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}(\vec{s}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq 2L}} A_{l,m} Y_{l,m}(\vec{s})$$

Il nous faut trouver des points de quadrature $\vec{s}_p$ et des poids ω_p qui intègrent exactement les harmoniques sphériques $Y_{l,m}(\theta,\phi)$ avec $0 \leq l \leq 2L$ et $-l \leq m \leq l$.

Le choix le plus simple est de prendre pour points d'intégration une distribution uniforme sur θ et sur ϕ :

$$\begin{cases} \theta_i = \pi \dfrac{i+1/2}{2L+1} & 0 \leq i \leq 2L, \\ \phi_j = 2\pi \dfrac{j}{2L+1} & 0 \leq j \leq 2L \end{cases}$$

associés aux poids d'intégration adéquats (donnés par (1.41)).

Nous reviendrons plus loin sur cette question cruciale. Le choix fait ici n'est pas optimal, mais il est le plus simple à implémenter, et est suffisant en première approche.

1.2.1.7 Récapitulatif

On a désormais tous les éléments pour réaliser un produit matrice-vecteur multipôle à un niveau. On se donne un courant surfacique $\vec{t}$ en entrée, et un découpage de Γ à travers une grille. Le produit $A.t$ se réalise en deux parties :

Interactions proches : pour tout cellule $\mathcal{C}'$ fixée, et pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on passe en revue les cellules $\mathcal{C}$ voisines de $\mathcal{C}'$ pour calculer de manière classique le terme d'interaction :

$$\sum_{\mathcal{C}\in v(\mathcal{C}')} \int_{x\in\Gamma\cap\mathcal{C}} \int_{y\in\Gamma\cap\mathcal{C}'} G(|y-x|) f(x) \varphi_j(y) dx dy \tag{1.12}$$

f est l'une des composantes de $\vec{t}$, ou bien $div_\Gamma(\vec{t})$. φ_j représente l'expression correspondante pour $\vec{\varphi_j}$. Le résultat de cette intégration constitue la partie « interaction proche » de la j-ième composante du vecteur $A.t$.

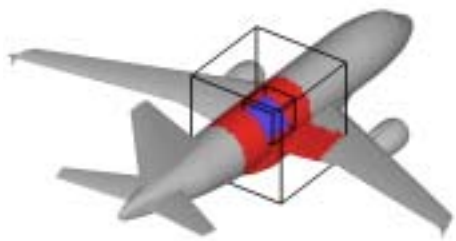


FIG. 1.8 – *Interactions proches sur un Airbus A318*

Cette partie du calcul est illustrée sur la figure 1.8 dans le cas d'un avion. Si $\mathcal{C}'$ est la boîte centrale (dont l'intersection avec le maillage est en bleu), la grande boîte, et la portion rouge du maillage, constituent la zone voisine qui interagira avec $\mathcal{C}'$ via l'équation (1.12).

Interactions Lointaines : le calcul se fait en trois étapes.

1. **Initialisation :** pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\mathcal{F}_\mathcal{C}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}} f(x)dx \tag{1.13}$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$.

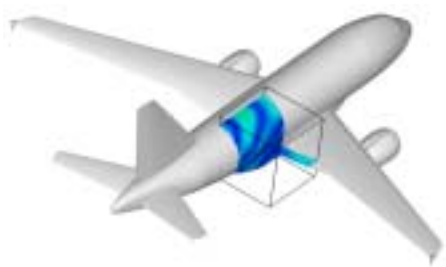


FIG. 1.9 – *Initialisation de la fonction de radiation $\mathcal{F}_\mathcal{C}$ à partir des courants surfaciques*

La figure 1.9 illustre le fait que le calcul d'une fonction de radiation $\mathcal{F}_\mathcal{C}$ se réalise à partir des seuls courants surfaciques contenus dans $\Gamma\cap\mathcal{C}$. Notons que ces courants, issus du vecteur $\vec{t}$ donné en entrée du produit matrice-vecteur, ont un sens mathématique mais n'auront pas de sens physique tant que le solveur itératif n'aura pas convergé.

2. **Transfert :** pour toute cellule $\mathcal{C}'$ fixée, on passe en revue les cellules $\mathcal{C}$ non-voisine de $\mathcal{C}'$ pour calculer :

$$\mathcal{G}_{\mathcal{C}'}(\vec{s}) = \sum_{\mathcal{C}\notin v(\mathcal{C}')} T^L_{M\vec{M}'}(\vec{s}).\mathcal{F}_\mathcal{C}(\vec{s}) \tag{1.14}$$

De la même manière que $\mathcal{F}_\mathcal{C}$ représente l'influence du domaine $\Gamma\cap\mathcal{C}$ sur l'extérieur, la fonction $\mathcal{G}_{\mathcal{C}'}$ représente l'influence de la partie de Γ loin de $\mathcal{C}'$ sur $\Gamma\cap\mathcal{C}'$.

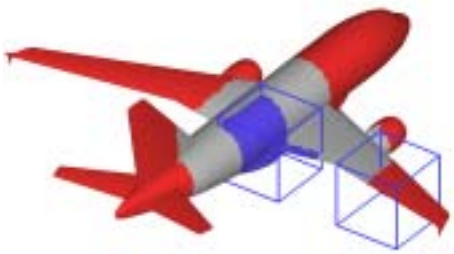


FIG. 1.10 – *Transferts des fonctions de radiation entre cellules non-voisines*

Sur la figure 1.10, $\Gamma \cap \mathcal{C}'$ est tracé en bleu tandis que la portion non-voisine du maillage est tracée en rouge.

3. **Intégration :** pour toute cellule $\mathcal{C}'$, et pour toute fonction de base φ_j localisée dans $\mathcal{C}'$, on calcule l'intégrale :

$$\frac{ik}{16\pi^2} \int_{y \in \Gamma \cap \mathcal{C}'} \int_{\vec{s} \in \mathcal{S}} \mathcal{G}_{\mathcal{C}'}(\vec{s}) e^{ik\vec{s}.\vec{M'}y} \varphi_j(y) d\vec{s} dy \tag{1.15}$$

Le résultat de cette intégration constitue la partie « interaction lointaine » de la j-ième composante du vecteur $A.t$, et se rajoute tout naturellement à la partie « interaction proche ».

1.2.2 Approfondissement

Dans la section 1.2.1, nous avons donné les principaux éléments permettant de réaliser un produit matrice-vecteur multipôle à un niveau. Nous allons maintenant détailler certains des points précédemment abordés.

1.2.2.1 Ecritures multipôles améliorées

Le but de cette section est d'améliorer la formulation multipôle présentée ci-dessus pour :

- diminuer sa complexité,
- permettre de traiter également les formulations *MFIE* et *CFIE*.

1.2.2.1.1 Produit multipôle EFIE à 3 termes On se donne un courant $\vec{t}$. Rappelons qu'un produit matrice-vecteur *EFIE* consiste à calculer pour toute fonction de base $\vec{\varphi_j}$ la quantité :

$$\int_\Gamma \int_\Gamma G(|y-x|) \left(\vec{t}(x).\vec{\varphi_j}(y) - \frac{1}{k^2} div_\Gamma \vec{t}(x).div_\Gamma \vec{\varphi_j}(y) \right) dxdy$$

La première écriture *EFIE* a consisté à transformer ceci en quatre produits multipôles correspondant aux composantes x, y, z et à la divergence des fonctions $\vec{t}$ et $\vec{\varphi_j}$.

Si on se donne deux cellules $\mathcal{C}$ et $\mathcal{C}'$ non-voisines, la méthode multipôle conduit à calculer le terme d'interaction correspondant sous la forme :

$$\int_{x\in\Gamma\cap\mathcal{C}}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}}T^L_{M\vec{M}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}\left(\vec{t}(x).\vec{\varphi_j}(y)-\frac{1}{k^2}div_\Gamma\vec{t}(x).div_\Gamma\vec{\varphi_j}(y)\right)dxdy \tag{1.16}$$

On s'intéresse spécifiquement au terme en divergence, et on réordonne les intégrales :

$$\int_{\vec{s}\in\mathcal{S}}\left(\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}div_\Gamma\vec{t}(x)dx\right)\left(\int_{y\in\Gamma\cap\mathcal{C}'} e^{ik\vec{s}.\vec{M'}y}div_\Gamma\vec{\varphi_j}(y)dy\right)T^L_{M\vec{M}'}(\vec{s})d\vec{s}$$

On réalise une intégration par partie en x dans la première parenthèse, en y dans la seconde. Les courants considérés sont de flux nul sur le bord de Γ, donc si Γ est régulière (ce que l'on suppose désormais), il n'y a pas de terme de bord dans ces deux intégrations par partie (sur le bord de $\Gamma\cap\mathcal{C}$, les courants seront aussi de flux nul : on précisera ce point au paragraphe 1.2.2.3.1). On obtient :

$$\begin{cases}\displaystyle\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}div_\Gamma\vec{t}(x)dx=-\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(-ik\vec{s}).\vec{t}(x)dx,\\ \displaystyle\int_{y\in\Gamma\cap\mathcal{C}'} e^{ik\vec{s}.\vec{M'}y}div_\Gamma\vec{\varphi_j}(y)dy=-\int_{y\in\Gamma\cap\mathcal{C}'} e^{ik\vec{s}.\vec{M'}y}(ik\vec{s}).\vec{\varphi_j}(y)dy\end{cases} \tag{1.17}$$

Si on replace cela dans (1.16), il reste :

$$\int_{x\in\Gamma\cap\mathcal{C}}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}}T^L_{M\vec{M}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}\left(\vec{t}(x).\vec{\varphi_j}(y)-\vec{s}.\vec{t}(x)\vec{s}.\vec{\varphi_j}(y)\right)dxdy$$

A partir de là, deux méthodes sont possibles.

Première méthode : pour aboutir à une formulation multipôle *EFIE* à 3 termes, il suffit de remarquer que :

$$\vec{t}.\vec{\varphi_j}-(\vec{s}.\vec{t})(\vec{s}.\vec{\varphi_j})=\left(\vec{t}-(\vec{s}.\vec{t})\vec{s}\right).\vec{\varphi_j} \tag{1.18}$$

Autrement dit, on se ramène à trois termes en ne considérant que les composantes x, y, et z des fonctions $\vec{\varphi_j}$ et $\vec{t}-(\vec{s}.\vec{t})\vec{s}$ (qui est la composante de $\vec{t}$ normale à $\vec{s}$). Dans la pratique, l'étape d'initialisation (1.13) devient :

Initialisation : pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s})=\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{t}(x)-(\vec{s}.\vec{t}(x))\vec{s})dx \tag{1.19}$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$. Dans (1.19), la fonction de radiation calculée a désormais 3 composantes ($\vec{\mathcal{F}_\mathcal{C}}_x$,$\vec{\mathcal{F}_\mathcal{C}}_y$,$\vec{\mathcal{F}_\mathcal{C}}_z$).

Deuxième méthode : on peut aussi écrire que :

$$\vec{t}.\vec{\varphi_j}-(\vec{s}.\vec{t})(\vec{s}.\vec{\varphi_j})=\vec{t}.\left(\vec{\varphi_j}-(\vec{s}.\vec{\varphi_j})\vec{s}\right) \tag{1.20}$$

Dans ce cas, c'est la phase d'intégration (1.15) qui devient :

Intégration : pour toute cellule $\mathcal{C}'$, et pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on calcule l'intégrale :

$$\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\left(\vec{\varphi_j}(y)-(\vec{s}.\vec{\varphi_j}(y))\vec{s}\right).\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s})d\vec{s}dy$$

De même que ci-dessus, il est sous entendu que la fonction de radiation utilisée a en fait 3 composantes (x,y,z).

Notons enfin que ces deux méthodes ne sont pas mutuellement exclusives, puisque l'on peut écrire :

$$\vec{t}.\vec{\varphi_j} - (\vec{s}.\vec{t})(\vec{s}.\vec{\varphi_j}) = \left(\vec{t} - (\vec{s}.\vec{t})\vec{s}\right).(\vec{\varphi_j} - (\vec{s}.\vec{\varphi_j})\vec{s}) \tag{1.21}$$

Dans ce cas, ce sont à la fois l'initialisation et l'intégration qui sont modifiées. Bien sûr, en l'état, cela n'apporte rien.

1.2.2.1.2 Produit multipôle EFIE à 2 termes La modification présentée ici est issue de [14].

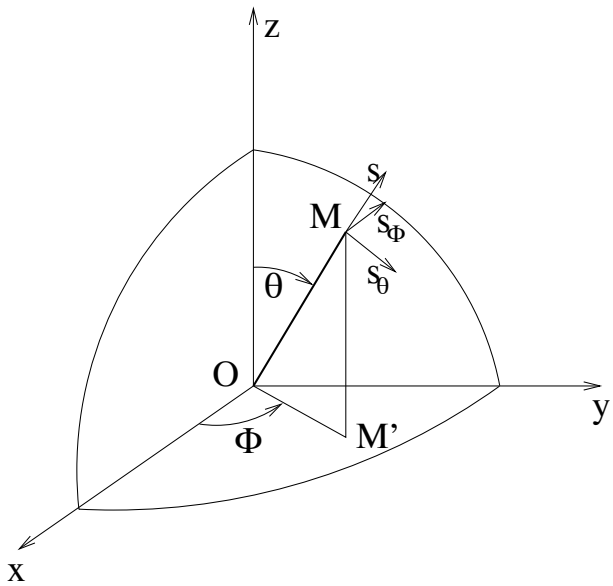


FIG. 1.11 – *Coordonnées sphériques sur la sphère unité $\mathcal{S}$*

Les différentes écritures (1.18), (1.20) et (1.21) nous montrent que seules les parties tangentielles de $\vec{t}$ et de $\vec{\varphi_j}$ sur la sphère unité $\mathcal{S}$ servent pour le calcul multipôle. Si on note ($\vec{s}$,$\vec{s_\theta}$,$\vec{s_\phi}$) la base orthonormée locale en coordonnées sphériques sur la sphère unité (cf. figure 1.11) , la composante tangentielle de $\vec{t}$ s'écrit :

$$\vec{t} - (\vec{s}.\vec{t})\vec{s} = (\vec{s_\theta}.\vec{t})\vec{s_\theta} + (\vec{s_\phi}.\vec{t})\vec{s_\phi}$$

Avec des notations identiques pour $\vec{\varphi_j}$, on a :

$$\begin{aligned}
\vec{t}.\vec{\varphi_j} - (\vec{s}.\vec{t})(\vec{s}.\vec{\varphi_j}) &= \left(\vec{t} - (\vec{s}.\vec{t})\vec{s}\right).(\vec{\varphi_j} - (\vec{s}.\vec{\varphi_j})\vec{s}) \\
&= \left((\vec{s_\theta}.\vec{t})\vec{s_\theta} + (\vec{s_\phi}.\vec{t})\vec{s_\phi}\right).((\vec{s_\theta}.\vec{\varphi_j})\vec{s_\theta} + (\vec{s_\phi}.\vec{\varphi_j})\vec{s_\phi}) \\
&= (\vec{s_\theta}.\vec{t})(\vec{s_\theta}.\vec{\varphi_j}) + (\vec{s_\phi}.\vec{t})(\vec{s_\phi}.\vec{\varphi_j})
\end{aligned}$$

On se ramène à une formulation multipôle à deux termes en ne considérant que les composantes selon $\vec{s_\theta}$ et $\vec{s_\phi}$ des fonctions de transfert $\vec{\mathcal{F}}$ et $\vec{\mathcal{G}}$. Dans la pratique, les étapes d'initialisation (1.13) et d'intégration (1.15) deviennent :

1. Initialisation : pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\vec{t}(x)dx$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$. Puis on passe aux composantes sphériques tangentielles :

$$\begin{cases} (\mathcal{F}_\mathcal{C})_\theta(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\theta}(\vec{s}), \\ (\mathcal{F}_\mathcal{C})_\phi(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\phi}(\vec{s}) \end{cases}$$

2. Transfert : inchangé. On effectue le transfert (1.14) séparément sur les deux fonctions $(\mathcal{F}_\mathcal{C})_\theta$ et $(\mathcal{F}_\mathcal{C})_\phi$.

3. Intégration : pour toute cellule $\mathcal{C}'$, on repasse en notation cartésienne (x,y,z) pour la fonction de radiation $\mathcal{G}_{\mathcal{C}'}$:

$$\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}) = (\mathcal{G}_{\mathcal{C}'})_\theta(\vec{s}).\vec{s_\theta}(\vec{s}) + (\mathcal{G}_{\mathcal{C}'})_\phi(\vec{s}).\vec{s_\phi}(\vec{s})$$

Puis, pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on intègre cette fonction de manière usuelle comme dans (1.15) :

$$\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}).\vec{\varphi_j}(y)d\vec{s}dy$$

1.2.2.1.3 Produit multipôle MFIE On se donne un courant $\vec{t}$. Rappelons qu'un produit matrice-vecteur *MFIE* consiste à calculer pour toute fonction de base $\vec{\varphi_j}$ la quantité :

$$\frac{1}{2}\int_\Gamma \vec{t}(y).\vec{\varphi_j}(y)dy + \int_\Gamma \vec{\varphi_j}(y).\vec{\nu}(y)\wedge\left(\int_\Gamma \vec{grad}_x G(|y-x|)\wedge\vec{t}(x)dx\right)dy$$

On va d'emblée ôter le premier terme $\int_\Gamma \vec{t}(y).\vec{\varphi_j}(y)dy$ qui sera forcément traité dans le cadre des interactions proches. L'interaction entre deux cellules $\mathcal{C}$ et $\mathcal{C}'$ non-voisines de notre découpage, centrées en M et M', s'écrit :

$$\int_{y\in\Gamma\cap\mathcal{C}'} \vec{\varphi_j}(y).\vec{\nu}(y)\wedge\left(\int_{x\in\Gamma\cap\mathcal{C}} \vec{grad}_x G(|y-x|)\wedge\vec{t}(x)dx\right)dy$$

En utilisant le théorème d'addition (1.4), cela donne :

$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'} \vec{\varphi_j}(y).\vec{\nu}(y)\wedge\left[\int_{x\in\Gamma\cap\mathcal{C}} \vec{grad}_x\left(\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}}T^L_{\vec{MM'}}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}\right)\wedge\vec{t}(x)dx\right]dy \tag{1.22}$$

Le gradient ne s'applique qu'à $e^{ik\vec{s}.x\vec{M}}$, seul terme dépendant de x, et il vaut :

$$\vec{grad}_x(e^{ik\vec{s}.x\vec{M}}) = -ik\vec{s}e^{ik\vec{s}.x\vec{M}}$$

En réordonnant les intégrations, l'expression (1.22) devient :

$$\frac{ik}{16\pi^2}(-ik)\int_{\vec{s}\in\mathcal{S}}\int_{y\in\Gamma\cap\mathcal{C}'}\vec{\varphi_j}(y).\vec{\nu}(y)\wedge\left[e^{ik\vec{s}.\vec{M'}y}\vec{s}\wedge\left(T^L_{M\vec{M}'}(\vec{s})\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}\vec{t}(x)dx\right)\right]dyd\vec{s} \tag{1.23}$$

Pour simplifier le calcul, on va utiliser la propriété vectorielle suivante :

$$\vec{a}.(\vec{b}\wedge\vec{c}) = \vec{c}.(\vec{a}\wedge\vec{b}) \tag{1.24}$$

Dans (1.23), on prend $\vec{a} = \vec{\varphi_j}(y)$, $\vec{b} = \vec{\nu}(y)$ et $\vec{c}$ le terme entre crochets. On obtient :

$$\frac{ik}{16\pi^2}(-ik)\int_{\vec{s}\in\mathcal{S}}\int_{y\in\Gamma\cap\mathcal{C}'}\left[e^{ik\vec{s}.\vec{M'}y}\vec{s}\wedge\left(T^L_{M\vec{M}'}(\vec{s})\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}\vec{t}(x)dx\right)\right].[\vec{\varphi_j}(y)\wedge\vec{\nu}(y)]dyd\vec{s} \tag{1.25}$$

On met de côté la constante initiale, on retrouve un algorithme multipôle à un niveau avec trois étapes qui s'écrivent :

1. Initialisation : pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\vec{\mathcal{F}_{\mathcal{C}}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}\vec{t}(x)dx$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$.

2. Transfert : comme pour l'*EFIE* : pour toute cellule $\mathcal{C}'$ fixée, on passe en revue les cellules $\mathcal{C}$ non-voisine de $\mathcal{C}'$ pour calculer :

$$\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}) = \sum_{\mathcal{C}\notin v(\mathcal{C}')}T^L_{M\vec{M}'}(\vec{s}).\vec{\mathcal{F}_{\mathcal{C}}}(\vec{s})$$

3. Intégration : pour toute cellule $\mathcal{C}'$, et pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on calcule l'intégrale :

$$\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}e^{ik\vec{s}.\vec{M'}y}\left[\vec{s}\wedge\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s})\right].[\vec{\varphi_j}(y)\wedge\vec{\nu}(y)]\,d\vec{s}dy \tag{1.26}$$

L'application de la formule (1.24) permet de séparer la dépendance en y et en $\vec{s}$ dans l'étape d'intégration (1.26). Par ailleurs, il faut noter que la composante de $\vec{t}(x)$ dirigée selon $\vec{s}$ n'intervient pas, grâce à la présence du $\vec{s}\wedge\ldots$ dans l'équation (1.23). On peut donc calculer l'étape d'initialisation avec $\vec{t}(x)-(\vec{t}(x).\vec{s})\vec{s}$ au lieu de $\vec{t}(x)$. Cela permet d'utiliser la même étape d'initialisation (1.19) pour l'*EFIE* multipôle et la *MFIE* multipôle. Cela permet également d'écrire une formulation multipôle à deux termes pour la *MFIE*, l'idée étant de manipuler les fonctions de radiation $\vec{\mathcal{F}}$ et $\vec{\mathcal{G}}$ non pas via leurs trois composantes ($\vec{\mathcal{F}}.\vec{u_x}$,$\vec{\mathcal{F}}.\vec{u_y}$,$\vec{\mathcal{F}}.\vec{u_z}$) mais via leurs composantes ($\vec{\mathcal{F}}.\vec{s_\theta}$,$\vec{\mathcal{F}}.\vec{s_\phi}$).

1.2.2.1.4 Produit multipôle CFIE Rappelons que la formulation *CFIE* (*Combined Field Integral Equation*) est une combinaison linéaire d'*EFIE* et de *MFIE* :

$$CFIE = \alpha EFIE + (1-\alpha)\frac{i}{k}MFIE \tag{1.27}$$

Compte tenu de ce que l'on vient de voir, on peut écrire une formulation multipôle à deux termes pour la *CFIE*. Voici le récapitulatif complet de l'algorithme, tel qu'il est actuellement implémenté.

Interactions proches : pour tout cellule $\mathcal{C}'$ fixée, et pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on passe en revue les cellules $\mathcal{C}$ voisines de $\mathcal{C}'$ pour calculer de manière classique le terme d'interaction :

$$\begin{aligned}
&\sum_{\mathcal{C}\in v(\mathcal{C}')} \int_{x\in\Gamma\cap\mathcal{C}} \int_{y\in\Gamma\cap\mathcal{C}'} \alpha \left[G(|y-x|) \left(\vec{t}(x).\vec{\varphi_j}(y) - \frac{1}{k^2} div_\Gamma \vec{t}(x).div_\Gamma \vec{\varphi_j}(y) \right) \right] \\
&+(1-\alpha)\frac{i}{k} \left[\vec{\varphi_j}(y).\vec{\nu}(y) \wedge \left(\vec{grad}_x G(|y-x|) \wedge \vec{t}(x) \right) \right] dxdy \\
&+(1-\alpha)\frac{i}{k}\frac{1}{2} \int_{y\in\Gamma\cap\mathcal{C}'} \vec{t}(y).\vec{\varphi_j}(y)dy
\end{aligned} \tag{1.28}$$

Le résultat de cette intégration constitue la partie « interaction proche » de la j-ième composante du vecteur $A.t$.

Interactions Lointaines : le calcul se fait en trois étapes.

1. **Initialisation :** pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}} \vec{t}(x)dx \tag{1.29}$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$. Puis on passe aux composantes sphériques tangentielles :

$$\begin{cases} (\mathcal{F}_\mathcal{C})_\theta(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\theta}(\vec{s}), \\ (\mathcal{F}_\mathcal{C})_\phi(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\phi}(\vec{s}) \end{cases} \tag{1.30}$$

2. **Transfert :** pour toute cellule $\mathcal{C}'$ fixée, on passe en revue les cellules $\mathcal{C}$ non-voisines de $\mathcal{C}'$ pour calculer :

$$(\mathcal{G}_{\mathcal{C}'})_\theta(\vec{s}) = \sum_{\mathcal{C}\notin v(\mathcal{C}')} T^L_{M\vec{M}'}(\vec{s}).(\mathcal{F}_\mathcal{C})_\theta(\vec{s}) \tag{1.31}$$

On calcule de même $(\mathcal{G}_{\mathcal{C}'})_\phi$.

3. **Intégration :** pour toute cellule $\mathcal{C}'$, on repasse aux coordonnées cartésiennes (x,y,z) pour les fonctions de radiation, en calculant :

$$\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}) = (\mathcal{G}_{\mathcal{C}'})_\theta(\vec{s}).\vec{s_\theta}(\vec{s}) + (\mathcal{G}_{\mathcal{C}'})_\phi(\vec{s}).\vec{s_\phi}(\vec{s}) \tag{1.32}$$

Puis, pour toute fonction de base $\vec{\varphi_j}$ localisée dans $\mathcal{C}'$, on calcule l'intégrale :

$$\begin{aligned}
&\frac{ik}{16\pi^2} \int_{y\in\Gamma\cap\mathcal{C}'} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y} \left[\alpha \left(\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}).\vec{\varphi_j}(y) \right) \right. \\
&\qquad \left. +(1-\alpha) \left(\vec{s} \wedge \vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}).\vec{\varphi_j}(y) \wedge \vec{\nu}(y) \right) \right] d\vec{s}dy
\end{aligned} \tag{1.33}$$

Le résultat de cette intégration constitue la partie « interaction lointaine » de la j-ième composante du vecteur $A.t$, et se rajoute à la partie « interaction proche » calculée précédemment. Dans (1.27), le coefficient i/k disparaît avec le $(-ik)$ de l'équation (1.25).

1.2.2.2 Troncatures et discrétisation

Les problèmes abordés ici sont cruciaux pour la méthode multipôle. Nous les avons survolés dans la section 1.2.1.6, nous y revenons plus en détail maintenant.

1.2.2.2.1 Troncature de la fonction de transfert On reprend les notations (1.9) définissant $\vec{r}$ et $\vec{r_0}$. On souhaite appliquer à nos équations intégrales la décomposition suivante pour le noyau de Green :

$$G(|y-x|) = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}(\vec{s}) d\vec{s} \tag{1.34}$$

Rappelons que l'on a choisi de prendre pour noyau de Green :

$$G(R) = \frac{e^{ikR}}{4\pi R}$$

Les formules écrites ici se retrouvent parfois pour la fonction e^{ikR}/R (avec un coefficient $1/4\pi$ en moins), ou pour $h_0^{(1)}(kR) = e^{ikR}/ikR$ (avec un coefficient $ik/4\pi$ en moins). Il semblerait que chaque auteur choisisse ses propres conventions de notation, je n'ai donc pas failli à cette tradition.

Rappelons que la fonction de transfert selon $\vec{r_0}$ s'écrit :

$$T^L_{\vec{r_0}}(\vec{s}) = \sum_{0\leq l\leq L} (2l+1) i^l h_l^{(1)}(k.|\vec{r_0}|) P_l(cos(\vec{s},\vec{r_0})) \tag{1.35}$$

La fonction $h_l^{(1)}(x)$ (fonction de Hankel sphérique du premier type de rang l) diverge lorsque l tend vers $+\infty$, il est donc faux d'écrire :

$$G(|y-x|) = \frac{ik}{16\pi^2} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{r}} \lim_{L\to+\infty} T^L_{\vec{r_0}}(\vec{s}) d\vec{s}$$

car la série $T^{+\infty}_{\vec{r_0}}$ diverge. De là survient toute la difficulté : il faut prendre suffisamment de termes dans $T^L_{\vec{r_0}}$ pour que la convergence ait lieu dans (1.34), mais pas trop car sinon la série (1.35) diverge.

En substituant la fonction de transfert (1.35) dans le théorème d'addition (1.34), et en utilisant l'égalité :

$$\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{r}} P_l(cos(\vec{s},\vec{r_0})) = 4\pi i^l j_l(k.|\vec{r}|) P_l(cos(\vec{r},\vec{r_0}))$$

on peut démontrer le résultat suivant :

$$G(|y-x|) = \frac{ik}{4\pi} \sum_{l\geq 0} (2l+1)(-1)^l h_l^{(1)}(k.|\vec{r_0}|) j_l(k.|\vec{r}|) P_l(cos(\vec{r},\vec{r_0})) \tag{1.36}$$

La série qui apparaît s'appelle série de Gegenbauer. C'est sa convergence qui détermine le choix de L. Le sujet fait encore l'objet de recherches à ce jour. Dans le cadre de notre implémentation, nous avons utilisé un résultat extrait de [15] et repris dans [16]. Cette proposition affirme que sous réserve que $\vec{r}$ et $\vec{r_0}$ vérifient :

$$\frac{r_0}{r} \geq \frac{\sqrt{5}}{2}$$

alors $\forall \epsilon < 1$, on peut trouver 4 constantes C_1, C_2, C_3 et C_4 telles que $L = C_1 + C_2 kr + C_3 \log(kr) + C_4 \log(\epsilon^{-1})$ soit un nombre suffisant de termes dans (1.36) pour assurer une erreur inférieure à ϵ :

$$\left| \sum_{l \geq L} (2l+1)(-1)^l h_l^{(1)}(k.|\vec{r_0}|) j_l(k.|\vec{r}|) P_l(cos(\vec{r},\vec{r_0})) \right| < \epsilon$$

On a déjà vu que la condition $\frac{r}{r_0} \leq \frac{2}{\sqrt{5}}$ était vérifiée automatiquement pour des cellules non-voisines de notre grille. Ce résultat nous assure donc l'existence d'une troncature convenable. Dans la pratique, c'est un point qui demeure très empirique. La formule suivante donne de bons résultats, avec une erreur ε en moyenne de l'ordre de 10^{-3} :

$$L = \sqrt{3}ka + 7{,}5 \log_{10}(\sqrt{3}ka + \pi)$$

On note a l'arête des cubes de la grille. On reviendra dans la section 3.4 sur le choix du nombre de pôles effectivement implémenté.

1.2.2.2.2 Discrétisation de la sphère unité

1.2.2.2.2.1 Nature des fonctions intégrées Le paramètre L est désormais fixé. On souhaite donc calculer le noyau de Green via la formule :

$$G(|y-x|) = \frac{ik}{16\pi^2} \int_{\vec{s} \in \mathcal{S}} e^{ik\vec{s}.\vec{r}} T_{\vec{r_0}}^L(\vec{s}) d\vec{s}$$

D'après [29], on a :

$$P_l(cos(\vec{s},\vec{r_0})) = \frac{4\pi}{2l+1} \sum_{-l \leq m \leq l} Y_{l,m}(\vec{s}) Y_{l,m}^*(\frac{\vec{r_0}}{r_0}) \tag{1.37}$$

où $Y_{l,m}^*$ désigne le complexe conjugué de $Y_{l,m}$.

La fonction de transfert (1.35) peut donc s'écrire :

$$T_{\vec{r_0}}^L(\vec{s}) = \sum_{0 \leq l \leq L} \sum_{-l \leq m \leq l} \left[4\pi i^l h_l^{(1)}(k.|\vec{r_0}|) Y_{l,m}^*(\frac{\vec{r_0}}{r_0}) \right] Y_{l,m}(\vec{s})$$

$T_{\vec{r_0}}^L$ est donc de largeur de bande finie. Dans la mesure où l'on sait intégrer ce type de fonction sur $\mathcal{S}$, on voudrait obtenir la même propriété pour $e^{ik\vec{s}.\vec{r}} T_{\vec{r_0}}^L(\vec{s})$.

On a un développement pour $e^{ik\vec{s}.\vec{r_0}}$ assez proche de l'expression de $T_{\vec{r_0}}^L$:

$$e^{ik\vec{s}.\vec{r}} = \sum_{l \geq 0} (2l+1) i^l j_l(k.|\vec{r}|) P_l(cos(\vec{s},\vec{r})) \tag{1.38}$$

où j_l est la fonction de Bessel sphérique de rang l.

Là encore, les développements observés autour de ce problème sont fortement empiriques. On peut néanmoins démontrer la proposition suivante (issue de [16] sous une forme légèrement différente) :

$\forall \epsilon < 1$, on peut trouver 4 constantes D_1, D_2, D_3 et D_4 telles que $K = D_1 + D_2 kr + D_3 \log(kr) + D_4 \log(\epsilon^{-1})$ soit un nombre suffisant de termes dans (1.38) pour assurer une erreur inférieure à ϵ :

$$\left| T^L_{\vec{r_0}}(\vec{s}) \left[\sum_{l \geq K} (2l+1) i^l j_l(k.|\vec{r}|) P_l(cos(\vec{s},\vec{r})) \right] \right| < \epsilon$$

Pour calculer $e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}(\vec{s})$, on peut se contenter de prendre K termes dans (1.38). En utilisant (1.37), on en déduit que $e^{ik\vec{s}.\vec{r}} T^L_{\vec{r_0}}(\vec{s})$ est une fonction de largeur de bande $L + K$ (à une erreur ϵ près). Dans la pratique, on prend $K = L$.

1.2.2.2.2.2 Cahier des charges On s'est donc ramené au problème suivant : trouver une quadrature $(\vec{s_k},\omega_k)_{k \in I}$ de $\mathcal{S}$ intégrant exactement les fonctions de largeur de bande $2L$ de la forme :

$$\sum_{\substack{-l \leq m \leq l \\ 0 \leq l \leq 2L}} A_{l,m} Y_{l,m}(\vec{s})$$

Les fonctions de radiation $\mathcal{F}_{\mathcal{C}}$ et $\mathcal{G}_{\mathcal{C}}$ seront connues uniquement par leurs valeurs sur cette grille $(\vec{s_k})_{k \in I}$. En minimisant le nombre de points de cette quadrature, on gagne donc à la fois en mémoire consommée et en temps de calcul. C'est un point crucial pour obtenir un code FMM performant. On cherche donc une quadrature avec le moins de points possible, capable d'intégrer exactement chacune des fonctions $Y_{l,m}$ avec $l \leq 2L$, *i.e.* de vérifier :

$$\sum_{k \in I} \omega_k Y_{l,m}(\vec{s_k}) = \int_{\vec{s} \in \mathcal{S}} Y_{l,m}(\vec{s}) d\vec{s} = \delta_{m,0} \delta_{l,0} \sqrt{4\pi} \tag{1.39}$$

En effet $\int_{\vec{s} \in \mathcal{S}} Y_{l,m}(\vec{s}) d\vec{s}$ peut être vu comme le produit scalaire de $Y_{l,m}$ et de $\sqrt{4\pi} Y_{0,0}$ (qui vaut 1). La base des harmoniques sphériques étant orthonormales, ce produit scalaire sera toujours nul, sauf si $(m,l) = (0,0)$ pour lesquels il vaudra $\sqrt{4\pi}$.

De plus, la condition (1.39) pour $(m,l) = (0,0)$ s'écrit juste :

$$\sum_{k \in I} \omega_k = 4\pi \tag{1.40}$$

Ce 4π est la surface de $\mathcal{S}$. On supposera cela vérifié pour ne s'intéresser qu'au cas $(m,l) \neq (0,0)$.

1.2.2.2.2.3 Discrétisation basique Rappelons que $Y_{l,m}$ peut s'écrire :

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

avec la constante

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

Les variables θ et ϕ étant séparées dans $Y_{l,m}$, on va chercher les points $\vec{s_k}$ sous la forme d'une grille (θ_i,ϕ_j), et les poids ω_k sous la forme de produits $\omega_i^\theta.\omega_j^\phi$. Pour $(m,l) \neq (0,0)$, la condition (1.39) devient :

$$\begin{aligned}\sum_{k\in I}\omega_k Y_{l,m}(\vec{s_k}) &= \sum_{i,j}\omega_i^\theta.\omega_j^\phi P_l^m(cos\theta_i)e^{im\phi_j}\\ &= \left(\sum_i \omega_i^\theta P_l^m(cos\theta_i)\right)\left(\sum_j \omega_j^\phi e^{im\phi_j}\right)\\ &= 0\end{aligned}$$

Selon ϕ, il faut avoir $\sum_j \omega_j^\phi e^{im\phi_j} = 0$ dès que $m \neq 0$, avec $-2L \leq m \leq 2L$. On choisit une distribution uniforme à $2L+1$ points sur [0,2π] :

$$\begin{cases}\phi_j = \dfrac{2\pi}{2L+1}j & 0 \leq j \leq 2L,\\ \omega_j^\phi = \dfrac{2\pi}{2L+1}\end{cases}$$

On obtient en effet la somme d'une suite géométrique :

$$\sum_{0\leq j\leq 2L}\omega_j^\phi e^{im\phi_j} = \frac{2\pi}{2L+1}\sum_{0\leq j\leq 2L}(e^{im\frac{2\pi}{2L+1}})^j = \frac{2\pi}{2L+1}\frac{1-(e^{im\frac{2\pi}{2L+1}})^{2L+1}}{1-(e^{im\frac{2\pi}{2L+1}})}$$

qui vaut 0 si $m \neq 0$. Si $m = 0$, cette somme vaut 2π

Selon θ, dans le cas où $m = 0$, il faut avoir $\sum_i \omega_i^\theta P_l(cos\theta_i) = 0$ dès que $l \neq 0$. Si $l = 0$, il faut que cette somme vaille 2 (afin de vérifier (1.40)). Pour intégrer exactement tous les polynômes de degré inférieur ou égal à $2L$, le plus simple est de prendre une distribution uniforme à $2L+1$ points sur [0,π] :

$$\theta_i = \frac{\pi(i+1/2)}{2L+1} \qquad 0 \leq i \leq 2L$$

(On rajoute 1/2 pour éviter les deux pôles $\theta = 0$ et $\theta = \pi$). Les poids sont obtenus en résolvant le système linéaire d'ordre $2L+1$:

$$\sum_{0\leq i\leq 2L} P_l(cos\theta_i)\omega_i^\theta = \delta_{l,0}2 \qquad \forall l = 0,\ \ldots,\ 2L \tag{1.41}$$

En résumé, on a choisit $2L+1$ angles ϕ et $2L+1$ angles θ pour intégrer exactement toutes les harmoniques sphériques de degré inférieur ou égal à $2L$.

1.2.2.2.2.4 Discrétisation usuelle La discrétisation précédente est certes la plus simple, mais elle n'est pas la plus utilisée.

En effet, une formule de Gauss-Legendre à L points permet d'intégrer exactement tous les polynômes de degré inférieur ou égal à $2L-1$. La raison de ce gain est que l'on ne choisit plus les points θ_i, la méthode nous impose à la fois les points et les poids. On trouvera une description de la méthode utilisée ici dans [47].

En deux mots, voici comment on procède :

1. On assemble la matrice tridiagonale T de taille $L \times L$ définie par :

$$\begin{cases}T_{i,i} = 0 & i = 1,\ \ldots,\ L,\\ T_{i,i+1} = T_{i+1,i} = \dfrac{i}{\sqrt{4i^2-1}} & i = 1,\ \ldots,\ L-1\end{cases}$$

2. On calcule les valeurs propres λ_i et vecteurs propres V_i de cette matrice (par exemple avec la fonction `SSTEV` de Lapack [27]).
3. On extrait les points et poids de quadrature :

$$\begin{cases} cos(\theta_i) = \lambda_i, \\ \omega_i^\theta = 2{(V_i)_0}^2 \end{cases}$$

où $(V_i)_0$ est la première coordonnée du i-ème vecteur propre V_i.

Dans notre cas, il nous faut prendre $L+1$ points d'intégration car nos polynômes sont de degré inférieur ou égal à $2L$. On obtient donc une quadrature de $\mathcal{S}$ à $(L+1).(2L+1)$ points. Il sera parfois nécessaire pour réaliser certaines optimisations d'arrondir le nombre d'angles ϕ au multiple de 2 ou de 4 supérieur.

1.2.2.2.2.5 Discrétisation optimale La recherche d'une quadrature $(\vec{s_k},\omega_k)$ intégrant exactement les harmoniques sphériques de rang au plus $2L$ a donc toujours une solution à $(L+1).(2L+1)$ points. Elle n'est en général pas optimale. Par exemple, McLaren [28] propose une quadrature à 72 points dans le cas $L = 7$, au lieu de $8 \times 15 = 120$ points. Le gain ainsi réalisé sur le nombre de points est important, d'autant qu'il va se reporter sur la consommation en mémoire pour les fonctions de radiation, et sur les temps de calcul pour les opérations d'initialisation, de transfert et d'intégration.

Néanmoins il y a aussi quelques inconvénients. D'une part ces quadratures ne sont plus des grilles uniformes en ϕ, ce qui nous interdira certaines améliorations (en particulier l'usage de transformée de Fourier dans le cas de *FMM* multi-niveau). D'autre part, ce type de quadrature est difficile à trouver et à calculer, et toutes ne sont pas aussi efficaces que celle évoquée ci-dessus.

Hardin et Sloane [24] effectuent une recherche exhaustive sur le problème pour des quadratures ayant jusqu'à 100 points, puis proposent une construction assez élaborée pour des quadratures au final à peine meilleures que la nôtre (environ 5 à 10% de points en moins). Il n'est donc pas certain que la recherche de la quadrature optimale pour un L donné soit rentable. En revanche, l'utilisation d'un schéma connu exceptionnellement bon, comme celui de McLaren, dans un code *FMM* peut s'avérer bénéfique.

1.2.2.3 Découpage du maillage

On s'intéresse au découpage de la surface Γ en sous-domaines évoqué au paragraphe 1.2.1.4. Dans la pratique, on ne découpe pas la surface Γ elle-même, mais plutôt le maillage triangulaire défini précédemment. Il faut donc préciser le sens à donner à l'expression $x \in \Gamma \cap \mathcal{C}$ apparaissant dans de nombreuses intégrales.

Pour réaliser une partition de notre maillage, trois approches sont possibles : la distribution des arêtes (et donc des degrés de liberté) entre les différents cubes de notre grille, la distribution des triangles, ou la distribution des points d'intégration.

1.2.2.3.1 Distribution des arêtes Si on distribue les arêtes en fonction de leur milieu(cf. figure 1.12), on cours le risque de voir le support de la fonction de base $\vec{\varphi_i}$ dépasser largement hors de la cellule $\mathcal{C}$ contenant l'arête A_i. Cela peut nuire à la précision du calcul. Par exemple, sur la figure 1.12, les arêtes A_i et A_j sont localisées dans les cellules non-voisines $\mathcal{C}$ et $\mathcal{C}'$. Par conséquent, les points M et M' vont interagir via la méthode multipôle, alors qu'ils sont dans la même cellule.

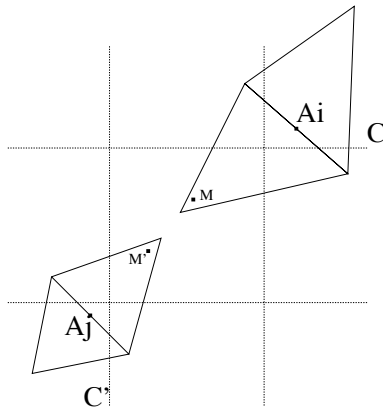


FIG. 1.12 – *Distribution des arêtes : cas critique*

Bien sûr, on a représenté ici un cas extrême, où la taille des cellules est très proche de celle des arêtes. Dans la pratique, avec des arêtes de taille $\lambda/10$ en moyenne, et des cellules de taille $\lambda/4$ pour les plus petites, on est plus proche du cas représenté figure 1.13.

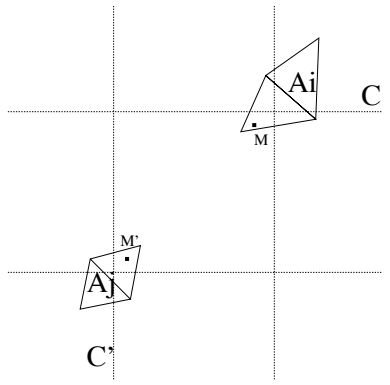


FIG. 1.13 – *Distribution des arêtes : cas réel*

Il y a un net avantage à ce type de distribution. Reprenons l'équation (1.17) :

$$\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}} div_\Gamma \vec{t}(x)dx = -\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(-ik\vec{s}).\vec{t}(x)dx \tag{1.42}$$

Cette intégration par partie permet d'écrire une formulation multipôle à 3 puis 2 termes pour l'EFIE, elle est donc d'une importance cruciale. Elle est valide si on remplace $x \in \Gamma \cap \mathcal{C}$ par $x \in \Gamma$, et si Γ est régulière, car $\vec{t}$ est de flux nul sur le bord éventuel de Γ. Il n'y a donc pas de terme de bord.

Lorsqu'on distribue les arêtes, si on écrit cette formule sur $\Gamma \cap \mathcal{C}$, cela veut dire que l'on ne travaille plus avec la fonction complète :

$$\vec{t}(x) = \sum_{1\leq i\leq n_{dl}} \lambda_i \vec{\varphi}_i(x)$$

mais avec sa restriction aux degrés de liberté situés dans $\Gamma \cap \mathcal{C}$:

$$\vec{t}_{\mathcal{C}}(x) = \sum_{A_i \in \mathcal{C}} \lambda_i \vec{\varphi}_i(x)$$

Aussi, il faut voir (1.42) comme étant :

$$\int_{x\in\Gamma} e^{ik\vec{s}.x\vec{M}} div_{\Gamma} \vec{t}_{\mathcal{C}}(x)dx = -\int_{x\in\Gamma} e^{ik\vec{s}.x\vec{M}}(-ik\vec{s}).\vec{t}_{\mathcal{C}}(x)dx$$

L'intégration par partie (1.42) est bien valide sans terme de bord. C'est le principal avantage de ce type de distribution par rapport aux deux autres types. Même si elle est potentiellement moins précise, elle est numériquement plus simple et plus « naturelle ». C'est celle que nous avons choisi.

1.2.2.3.2 Distribution des points d'intégration Les intégrales de surface sur Γ apparaissant au moment des étapes d'initialisation ou d'intégration sont calculées numériquement à l'aide de point d'intégration, généralement appelés points de Gauss. On note (G_k,ω_k) les points et poids de Gauss utilisés.

Le découpage du maillage peut se faire en répartissant ces points entre les cellules de la grille. Dans ce cas, une intégrale sur $\Gamma \cap \mathcal{C}$ se calculera :

$$\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}} \vec{t}(x)dx = \sum_{G_k\in\Gamma\cap\mathcal{C}} \omega_k.e^{ik\vec{s}.\vec{G_k M}} \vec{G_k}(x)$$

En théorie, cette méthode génère le moins de problèmes de précision. Dans le cas de la figure 1.12, l'interaction des points M et M' serait calculée de manière « proche » (si du moins c'étaient des points de Gauss).

En revanche, l'intégration par partie (1.42) ne pourrait pas se faire sans terme de bord, la fonction $\vec{t}$ ayant a priori un flux non-nul à travers le bord de $\Gamma \cap \mathcal{C}$. Cela rend plus délicate l'implémentation de cette amélioration.

1.2.2.3.3 Distribution des triangles Si on distribue les triangles en fonction de leur centre (cf. figure 1.14), certaines fonctions de base $\vec{\varphi}_i$ vont avoir leur support partagé entre deux cellules $\mathcal{C}$ et $\mathcal{C}'$. Ce type de distribution cumule les inconvénients des deux autres (potentiellement moins précise, terme de bord dans (1.42)) sans en avoir les avantages. Cela rend plus délicate l'implémentation du calcul multipôle.

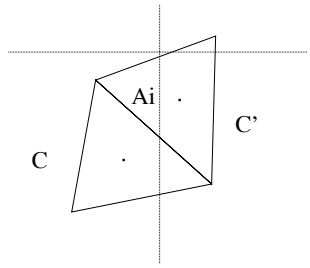


FIG. 1.14 – *Distribution des triangles : cas critique*

1.2.2.4 Nombre d'opérations

On va tenter de chiffrer le nombre moyen d'opérations flottantes exécutées lors d'un calcul multipôle, afin de quantifier le gain par rapport à un produit matrice-vecteur usuel. Pour simplifier, on ne va pas prendre en compte les temps d'initialisation de chacune des deux méthodes (calcul de la matrice complète dans le cas classique, calcul de la matrice des interactions proches, des matrices de translation dans le cas multipôle). On se place dans le cas d'un calcul nécessitant beaucoup de produits matrice-vecteur, et pour lequel la phase d'initialisation peut être négligée.

1.2.2.4.1 Données Les données du problème sont :

- Le nombre de degrés de liberté n_{dl}.
- La longueur d'onde λ.
- Le nombre de points par longueur d'onde d (*i.e.* la taille moyenne des arêtes est λ/d).
- L'arête des cellules cubiques de la grille en nombre de longueurs d'onde a (*i.e.* les cubes ont pour arête λa).
- Le nombre de points de Gauss : 3 par triangle.

1.2.2.4.2 Paramètres déduits On en déduit les grandeurs suivantes (qui sont à chaque fois des valeurs moyennes ou approchées) :

- La surface moyenne des triangles du maillage : avec des triangles équilatéraux d'arête λ/d, on a $|\mathcal{T}| = \frac{\sqrt{3}}{4}\frac{\lambda^2}{d^2}$.
- Le nombre de triangles du maillage : avec trois degrés de liberté par triangle et deux triangles par degrés de liberté, on a $n_{\mathcal{T}} = \frac{2}{3}n_{dl}$.
- La surface du maillage $|\Gamma| = n_{\mathcal{T}}.|\mathcal{T}| = \frac{\lambda^2 n_{dl}}{2\sqrt{3}d^2}$.
- La surface moyenne d'un domaine : sur des cas réels, l'intersection d'un cube d'arête $a\lambda$ avec Γ vaut en moyenne $|\Gamma \cap \mathcal{C}| = \lambda^2 a^2/2$.
- Le nombre de cellules $n_{\mathcal{C}} = |\Gamma|/|\Gamma \cap \mathcal{C}| = \frac{n_{dl}}{\sqrt{3}d^2a^2}$.
- Le nombre de triangles par cellule $n^c_{\mathcal{T}} = n_{\mathcal{T}}/n_{\mathcal{C}} = \frac{2d^2a^2}{\sqrt{3}}$.
- Le nombre de degrés de liberté par cellule $n^c_{dl} = n_{dl}/n_{\mathcal{C}} = \sqrt{3}d^2a^2$.
- Le nombre de pôles pour le calcul des fonctions de transfert $L = \sqrt{3}ka\lambda = 2\pi\sqrt{3}a$.
- Le nombre de points de quadrature sur la sphère unité $n_{\vec{s}} = 2L^2 = 24\pi^2a^2$.

On suppose enfin que chaque cellule a 9 voisins (y compris elle-même) et $n^c - 9$ cellules non-voisines.

1.2.2.4.3 Coût d'une *FMM* à un niveau On calcule maintenant le coût de chaque étape de la *FMM*. On se place dans le cas d'une *CFIE* à 2 termes décrit à la section 1.2.2.1.4. La notation $(a \times , b+)$ désigne un nombre de multiplications et d'additions complexes.

Initialisation : le nombre d'initialisations est égal au nombre de cellules $n_{\mathcal{C}}$. Le nombre d'opérations d'une initialisation (1.29) est $((1\times , 1+)$ par point de Gauss, par dimension et par direction soit $18n_{\vec{s}}n^c_{\mathcal{T}}$. Le passage aux composantes tangentielles (1.30) représente $2n_{\vec{s}}$ produits scalaires $((3 \times , 2+)$ chacun) soit $10n_{\vec{s}}$ opérations.

Total :

$$\begin{aligned} f_{init} &= n_{\mathcal{C}} n_{\vec{s}}[18 n_{\mathcal{T}}^{c} + 10] \\ &= n_{dl}\left[288\pi^2 a^2 + \frac{240\pi^2}{\sqrt{3}d^2}\right] \end{aligned} \tag{1.43}$$

Transfert : chaque cellule est transférée sur toutes les cellules non-voisines soit $n_{\mathcal{C}}(n_{\mathcal{C}} - 9)$ transferts. Chaque transfert (1.31) opère sur deux fonctions de radiation ($(\mathcal{F}_{\mathcal{C}})_\theta$ et $(\mathcal{F}_{\mathcal{C}})_\phi$) à $n_{\vec{s}}$ points chacune, avec $(1\times\ ,\ 1+)$ à chaque fois, soit un total de $4n_{\vec{s}}$ opérations par transfert.
Total :

$$\begin{aligned} f_{trans} &= 4 n_{\mathcal{C}}(n_{\mathcal{C}} - 9) n_{\vec{s}} \\ &= n_{dl}\frac{96\pi^2}{\sqrt{3}d^2}\left[\frac{n_{dl}}{\sqrt{3}d^2a^2} - 9\right] \end{aligned} \tag{1.44}$$

Intégration : le nombre d'intégrations est égal au nombre de cellules $n_{\mathcal{C}}$. Le retour aux composantes cartésiennes (1.32) représente $(2\times\ ,\ 1+)$ par direction et par dimension soit $9n_{\vec{s}}$ flops. Le calcul de $\vec{s}\wedge\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s})$ représente $9n_{\vec{s}}$ flops. L'intégration (1.33) comporte $(6\times\ ,\ 5+)$ par degré de liberté, par point de Gauss (6 sont concernés pour chaque degré de liberté) et par direction $\vec{s}$, soit $66 n_{dl}^{c} n_{\vec{s}}$ opérations flottantes.
Total :

$$\begin{aligned} f_{integ} &= n_{\mathcal{C}} n_{\vec{s}}(18 + 66 n_{dl}^{c}) \\ &= n_{dl} 24\pi^2\left[\frac{6\sqrt{3}}{d^2} + 66a^2\right] \end{aligned} \tag{1.45}$$

Interactions Proches : chaque cellule interagit avec elle-même et (en moyenne) ses huit voisins soit $9n_{\mathcal{C}}$ calculs d'interactions proches. A chaque fois, (1.28) réalise un produit matrice-vecteur de taille $n_{dl}^{c}\times n_{dl}^{c}$ soit $2(n_{dl}^{c})^2$ opérations.
Total :

$$\begin{aligned} f_{proche} &= 18 n_{\mathcal{C}}(n_{dl}^{c})^2 \\ &= n_{dl} 18\sqrt{3}d^2a^2 \end{aligned} \tag{1.46}$$

1.2.2.4.4 Interprétation

1.2.2.4.4.1 Etude en fonction de la taille des boîtes Le paramètre le plus intéressant à faire varier est la taille des boîtes a. Pour un maillage donné (d et n_{dl} fixés), on étudie la variation des fonctions obtenues ci-dessus en fonction du paramètre a^2.

Les fonctions f_{init}, f_{integ} et f_{proche} sont croissantes en a : plus le découpage utilise une grille large, plus ces calculs sont longs. La diminution du nombre de cellules ne suffit pas à compenser l'augmentation du nombre de points définissant les fonctions de radiation. Au contraire, la fonction f_{trans} décroît en a : la diminution du nombre de transferts suffit à compenser l'augmentation de la taille des fonctions de radiation. Si a est trop petit, les transferts sont coûteux, si a est trop grand, ce sont les autres étapes qui coûtent. Il y a donc un paramètre optimal a_{opt} pour lequel la dérivée de $f_{init} + f_{trans} + f_{integ} + f_{proche}$ s'annule. Appelons f_{total} cette fonction. On note A la quantité a^2. On a :

$$f_{total}(A) = n_{dl}\left[(1872\pi^2 + 18\sqrt{3}d^2)A - 64\sqrt{3}\frac{\pi^2}{d^2} + \frac{32\pi^2 n_{dl}}{d^4}\frac{1}{A}\right] \tag{1.47}$$

Pour un A fixé, cette fonction comporte une partie en n_{dl} et une en n_{dl}^2. La dérivée s'annule au point optimal A_{opt} suivant :

$$A_{opt} = \sqrt{\frac{32\pi^2 n_{dl}}{d^4(1872\pi^2 + 18\sqrt{3}d^2)}}$$

Le nombre d'opérations optimal $f_{total}(A_{opt})$ est alors proportionnel à $n_{dl}^{3/2}$. Le nombre de cellules $n_{\mathcal{C}}$, le nombre de directions $n_{\vec{s}}$ et le nombre de degrés de liberté par cellule n_{dl}^c sont tous proportionnels à $\sqrt{n_{dl}}$. La taille de la matrice des interactions proches est proportionnelle à $n_{dl}^{3/2}$, de même que la taille de l'ensemble des matrices de transfert. Avec un produit usuel, on aurait un nombre d'opérations et un stockage proportionnels à n_{dl}^2. Il y a donc déjà un gain conséquent. On retrouve ici des résultats vus dans [40] et [15].

Dans la pratique, d est généralement de l'ordre de 10. Le paramètre a_{opt} vaut alors :

$$a_{opt} = \left(\frac{32\pi^2 n_{dl}}{d^4(1872\pi^2 + 18\sqrt{3}d^2)}\right)^{1/4} = 3{,}48.10^{-2} n_{dl}^{1/4}$$

Ce paramètre est un peu plus plus petit que l'optimum constaté en pratique. Cela provient du choix fait sur le nombre de pôles : on a pris $L = 2\pi\sqrt{3}a$. Cette formule donne de mauvais résultats pour a petit, on utilise plutôt une formule de la forme $L = 2\pi\sqrt{3}a + 7\log_{10}(2\pi\sqrt{3}a + \pi)$ qui améliore la précision en augmentant le nombre de pôles des petites boîtes. De ce fait, l'optimum a_{opt} se déplace vers une valeur supérieure. Si on appliquait ce choix dans les calculs qui précèdent, il ne serait plus possible de les mener jusqu'au bout explicitement. La formule (1.47) deviendrait :

$$f_{total}(a) = n_{dl}\left[1872\pi^2 L(a)^2 + 36\sqrt{3}d^2a^2 - 32\sqrt{3}\frac{\pi^2 L(a)^2}{d^2a^2} + \frac{8\pi^2 n_{dl}}{d^4}\frac{L(a)^2}{a^4}\right]$$

avec $L(a) = 2\pi\sqrt{3}a + 7\log_{10}(2\pi\sqrt{3}a + \pi)$.

L'approche la plus efficace serait d'implémenter dans le code multipôle un module utilisant une approche comme celle développée ici pour calculer les paramètres optimaux à mettre en œuvre. Dans la pratique, le lecteur en quête de simplicité pourra prendre par exemple $a = 0{,}5$ et $L = 12$ (cf. tableau 1.1).

On peut donc retenir de cette étude que moyennant quelques simplifications, on obtient une complexité théorique en $n_{dl}^{3/2}$ pour la méthode multipôle mono-niveau, mais une petite étude au cas par cas est nécessaire pour obtenir les paramètres optimaux.

1.2.2.4.4.2 Etude en fonction de la finesse du maillage On s'intéresse maintenant au paramètre d qui fixe la taille moyenne des arêtes à la valeur λ/d. Pour un maillage donné, si d varie, le nombre de degrés de liberté n_{dl} varie conjointement de telle sorte que n_{dl}/d^2 reste constant. En effet, la surface du maillage est donné par $|\Gamma| = \frac{\lambda^2 n_{dl}}{2\sqrt{3}d^2}$ et cette quantité reste constante.

En utilisant l'équation :

$$\frac{n_{dl}}{d^2} = \frac{2\sqrt{3}|\Gamma|}{\lambda^2}$$

le nombre total d'opérations (1.47) en fonction de d s'écrit :

$$f_{total}(d) = \frac{2\sqrt{3}|\Gamma|}{\lambda^2}\left[18\sqrt{3}a^2d^4 + 1872\pi^2a^2d^2 - 64\sqrt{3}\pi^2 + \frac{64\pi^2\sqrt{3}|\Gamma|}{\lambda^2a^2}\right] \tag{1.48}$$

Il ne s'agit pas ici, contrairement au paragraphe précédent, de choisir d pour minimiser le nombre d'opérations, puisqu'on trouverait automatiquement $d = 0$ (i.e. pas d'inconnues!). Dans le cas classique (sans méthode multipôle), le choix de d fait l'objet d'un compromis entre la précision (qui croît avec d) et le temps de calcul (qui croît comme d^4). La valeur choisie est en général comprise entre $d = 6$ et $d = 10$. Il arrive également qu'en fonction du résultat escompté certaines parties du maillage soient plus finement maillées que d'autres ($d = 10$ par endroit, $d = 2$ à d'autres endroits) .

Ici, le surcoût lié à l'augmentation de d est moindre. Dans la formule (1.48), on voit trois types de dépendance en d :

- En d^4 : ce sont les interactions proches.
- En d^2 : ce sont les initialisations et les intégrations.
- Constant : ce sont les étapes de transfert.

Or les étapes de transfert sont les plus chères. Par conséquent, on peut se permettre de faire varier d jusqu'à des valeurs de l'ordre de 15 ou 20 sans surcoût excessif. On peut aussi (et c'est même recommandé) mailler uniformément toute la surface de l'objet avec la même valeur de d. Cela peut éviter de se trouver dans la situation décrite sur la figure 1.12. Nous reviendrons sur ce point au paragraphe 3.4.3.

1.2.2.4.4.3 Etude en fonction de la longueur d'onde Après avoir généré un maillage complexe, on souhaite l'utiliser pour faire des calculs à plusieurs fréquences. Par exemple, si ce maillage comporte 10 points par longueur d'onde pour une certaine fréquence f, on pourra l'utiliser sans problème de précision pour faire des calculs aux fréquences $f' \in [0{,}8.f; 1{,}2f]$, puisqu'on aura toujours entre 8 et 12 points par longueur d'onde. On note $\alpha \in [0{,}8; 1{,}2]$ le coefficient de variation de la longueur d'onde. Dans (1.48), λ et d sont remplacés par $\alpha\lambda$ et αd (de telle sorte que la taille moyenne des arêtes λ/d – qui est une donnée du maillage – reste inchangé). On obtient le nombre total d'opérations en fonction de α :

$$f_{total}(\alpha) = \frac{2\sqrt{3}|\Gamma|}{\lambda^2}\left[18\sqrt{3}a^2d^4\alpha^2 + 1872\pi^2a^2d^2 - \frac{64\sqrt{3}\pi^2}{\alpha^2} + \frac{64\pi^2\sqrt{3}|\Gamma|}{\lambda^2a^2\alpha^4}\right]$$

Si en partant de $\alpha = 1$, on augmente α (c'est-à-dire si la fréquence diminue et le nombre de points par longueur d'onde augmente), le coût de la partie interactions proches croît comme α^2 tandis que la partie purement FMM voit son coût diminuer en α^{-2} et α^{-4}. Globalement le calcul sera légèrement plus rapide. Si α diminue en-dessous de 1, c'est bien sûr le contraire. Cela ne prend pas en compte la nécessité de recalculer la matrice des interactions proches.

1.3 Méthode multi-niveau

1.3.1 Premier survol : calcul à 2 niveaux

Comme on l'a fait dans la section consacrée à la *FMM* mono-niveau, on va présenter la méthode multi-niveau dans une version basique, ce qui nous permettra d'évoquer les concepts fondamentaux dans une relative simplicité.

1.3.1.1 Principe de base

L'idée de base est d'appliquer une approche *divide-and-conquer* à la méthode multipôle, à la manière de l'algorithme de tri *quickSort* ou de la transformée de Fourier rapide. Dans l'algorithme *quickSort*, on divise le tableau à trier en deux moitiés que l'on trie séparément avant de les fusionner. Dans l'algorithme FFT, on procède de la même manière. A chaque fois, l'opération (tri dans le premier cas, Fourier dans le second) effectuée sur chaque demi-ensemble rend triviale la même opération effectuée sur l'ensemble tout entier.

Dans notre cas, on a aussi une propriété de ce type : si on divise une cellule $\mathcal{C}$ centrée en M en deux cellules $\mathcal{C}_1$ centrée en M_1 et $\mathcal{C}_2$ centrée en M_2, on a la relation suivante entre fonctions de radiation définies par l'équation (1.8) :

$$\mathcal{F}_{\mathcal{C}}(\vec{s}) = e^{ik\vec{s}.\overrightarrow{M_1M}}\mathcal{F}_{\mathcal{C}_1}(\vec{s}) + e^{ik\vec{s}.\overrightarrow{M_2M}}\mathcal{F}_{\mathcal{C}_2}(\vec{s})$$

On va donc tenter de tirer partie de ce type de propriété pour écrire une méthode multipôle à deux niveaux.

1.3.1.2 Découpage hiérarchique

On définit un découpage à deux niveaux de Γ (cf. figure 1.15). La grille « large » constitue le niveau 0. La grille fine constitue le niveau 1, elle est une subdivision de la précédente avec un pas deux fois plus petit.

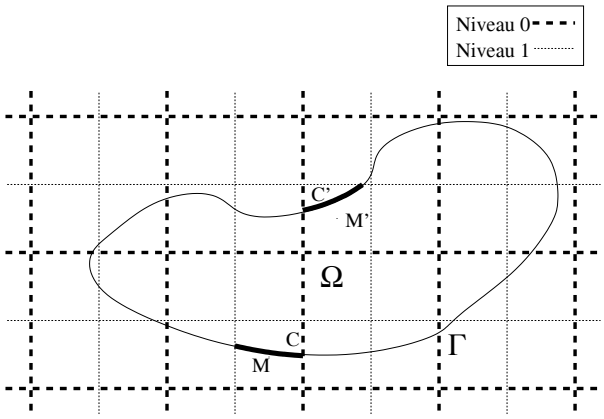


FIG. 1.15 – *Découpage à deux niveaux de* Γ

On définit une structure hiérarchique s'apparentant à un arbre entre ces deux grilles. Le niveau 0 est le haut de l'arbre, le niveau 1 le bas de l'arbre. Les cellules du niveau 1 issues de la division d'une cellule du niveau 0 sont appelés « enfants » de cette cellule. La relation inverse définit le « parent » d'une cellule. Ce type de lien est illustré par des flèches sur la figure 1.16. En trois dimensions, chaque cellule du niveau 0 a au plus huit enfants (on ne garde que les cellules ayant une intersection non-vide avec Γ), et chaque cellule du niveau 1 a exactement un parent.

Pour circuler au sein de cet arbre, définissons quelques notations. On indique le niveau d'une cellule par un exposant entre parenthèse à côté du nom de cette cellule : $\mathcal{C}^{(0)}$ ou $\mathcal{C}^{(1)}$. On note $p(\mathcal{C})$ le parent d'une cellule, et $e(\mathcal{C})$ l'ensemble des enfants d'une cellule. Enfin, on

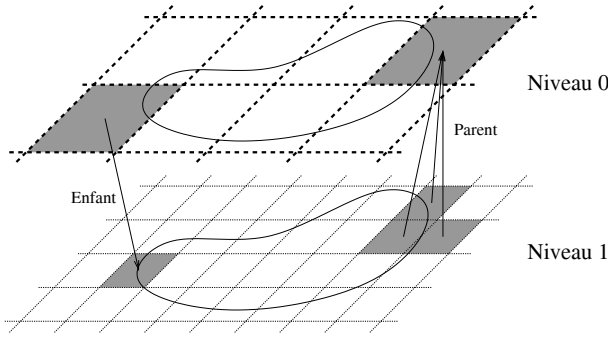


FIG. 1.16 – *Structure hiérarchique*

qualifie de voisines deux cellules ayant au moins un sommet en commun *et se trouvant au même niveau de l'arbre*. On conserve la notation $v(\mathcal{C})$ pour désigner l'ensemble des cellules voisines de $\mathcal{C}$.

1.3.1.3 Quelles interactions à quel niveau?

Avec deux grilles, on a potentiellement deux méthodes multipôle, une au niveau 0, l'autre au niveau 1. Essayons de voir quand utiliser chacune des deux.

On se donne une cellule $\mathcal{C}'^{(1)}$ ainsi qu'une fonction de base φ_j localisée dans $\mathcal{C}'^{(1)}$. Sur la figure 1.17, on a représenté en gris foncé la cellule $\mathcal{C}'^{(1)}$ ainsi que son parent $p(\mathcal{C}'^{(1)})$. En gris clair, on a leurs voisins respectifs. On a également représenté par des points épais sur chacun des deux niveaux quatre degrés de liberté sur Γ, dont le numéro j localisé dans $\mathcal{C}'^{(1)}$.

De manière schématique, pour un niveau donné, cette zone gris clair représente la portion du maillage qui ne peut pas interagir en mode « multipôle » avec la fonction de base φ_j localisée dans la cellule gris foncé (cf. section 1.2.1.6). Evidemment, la zone blanche contient la portion du maillage qui peut interagir en mode « multipôle » avec φ_j.

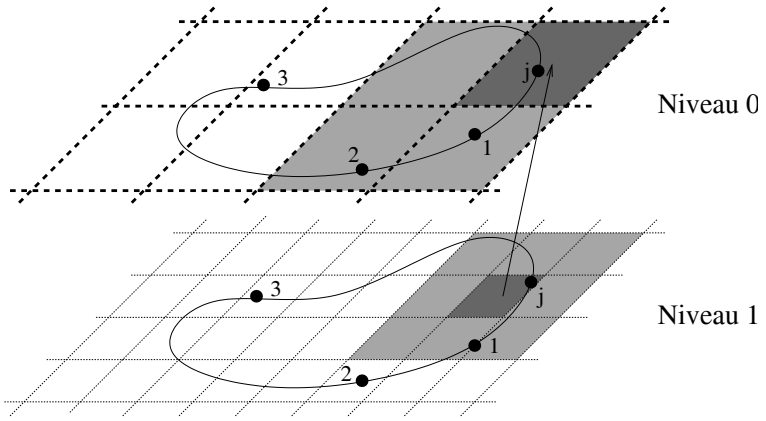


FIG. 1.17 – *Voisins de $\mathcal{C}'^{(1)}$ et de son parent $p(\mathcal{C}'^{(1)})$.*

Pour traiter l'interaction d'un degré de liberté donné avec le degré de liberté j, trois cas

de figure se présentent. Ils sont représentés sur la figure 1.17.

- L'interaction entre le dl 1 et le dl j ne peut se faire en mode multipôle ni au niveau 0, ni au niveau 1. On est obligé de la traiter en mode proche.
- L'interaction entre le dl 2 et le dl j peut se faire en mode multipôle au niveau 1 ou en mode proche au niveau 0.
- L'interaction entre le dl 3 et le dl j peut se faire en mode multipôle aux niveaux 0 et 1.

D'après la formule (1.44), plus les cellules sont grandes, et plus les transferts de la FMM sont rapides. Par conséquent, on va chercher à traiter les interactions en mode multipôle chaque fois que c'est permis, et ce au plus haut niveau possible (ici, au niveau 1 pour le dl 2, au niveau 0 pour le dl 3). On reviendra plus en détail sur le comptage du nombre d'opérations à la section 1.3.2.3.

1.3.1.4 Notion de banlieue

On va définir la notion de banlieue : on dit que deux cellules d'un même niveau sont « banlieues » l'une de l'autre si elles ne sont pas voisines, mais que leur parents respectifs le sont. On note $b(\mathcal{C})$ l'ensemble des banlieues de $\mathcal{C}$. Compte tenu de ce qui a été vu précédemment, la notion de banlieue apparaît naturellement : au niveau 1, $b(\mathcal{C})$ est l'ensemble des cellules interagissant avec $\mathcal{C}$ en mode multipôle au niveau 1.

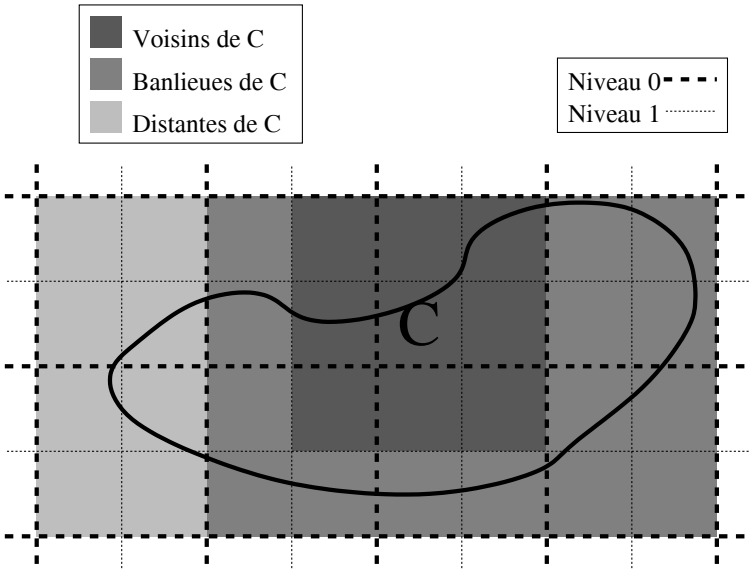


FIG. 1.18 – *Voisins et Banlieues au niveau 1*

Comme on le voit sur la figure 1.18, les cellules du niveau 1 sont partagées en trois sous-ensembles qui sont (des plus proches aux plus éloignées de $\mathcal{C}$) :

- Les voisines de $\mathcal{C}$, notés $v(\mathcal{C})$, contenant $\mathcal{C}$ elle-même ;
- Les banlieues de $\mathcal{C}$, notées $b(\mathcal{C})$;
- Les cellules distantes de $\mathcal{C}$, notées $d(\mathcal{C})$, qui sont les cellules restantes..

Compte tenu de la définition de $v(\mathcal{C})$ et de $b(\mathcal{C})$, on peut écrire :

$$\begin{cases} v(\mathcal{C}) \cup b(\mathcal{C}) = e\left[v(p(\mathcal{C}))\right], \\ d(\mathcal{C}) = \complement\left[v(\mathcal{C}) \cup b(\mathcal{C})\right] = \complement e\left[v(p(\mathcal{C}))\right] \end{cases} \tag{1.49}$$

où l'on note $\complement\mathcal{A}$ le complémentaire d'un ensemble $\mathcal{A}$.

La figure 1.19 illustre ce découpage sur un cas réel. Les images 1.19a et 1.19b montrent les découpages des niveaux 0 et 1. Sur les figures suivantes, on a repéré avec une flèche un degré de liberté sur le dos de l'appareil. Les figures 1.19c et 1.19d montrent en rouge les portions de maillage considérées comme non-voisines de cette inconnue aux niveaux 0 et 1, la figure 1.19e montre les banlieues au niveau 1 du degré de liberté pointé, et la figure 1.19f montre la portion voisine au niveau 1. On voit clairement que les banlieues au niveau 1 sont obtenus par soustraction ensembliste des portions de maillage non-voisines au niveau 1 et 0.

Le calcul de la composante du produit matrice-vecteur correspondant au degré de liberté pointé se fera alors :

- via la méthode multipôle au niveau 0 pour la portion rouge de la figure 1.19c ;
- via la FMM au niveau 1 pour la portion rouge de la figure 1.19e ;
- via une méthode classique (non-multipôle) pour la portion rouge de la figure 1.19f.

1.3.1.5 Algorithme continu

On va écrire un premier algorithme multipôle à deux niveaux, sans tenir compte pour l'instant des problèmes de discrétisation ou de nombre de pôles.

Pour toute cellule $\mathcal{C}'^{(1)}$ fixée, et pour toute fonction de base φ_j localisée dans $\mathcal{C}'^{(1)}$, on cherche à calculer la j-ième composante du produit matrice-vecteur $(A.t)$. Elle s'écrit :

$$(A.t)_j = \int_{x\in\Gamma} \int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} G(|y-x|)t(x)\varphi_j(y)dxdy$$

Chacun des trois sous-ensembles identifiés sur les figures 1.18 et 1.19 va être traité séparément.

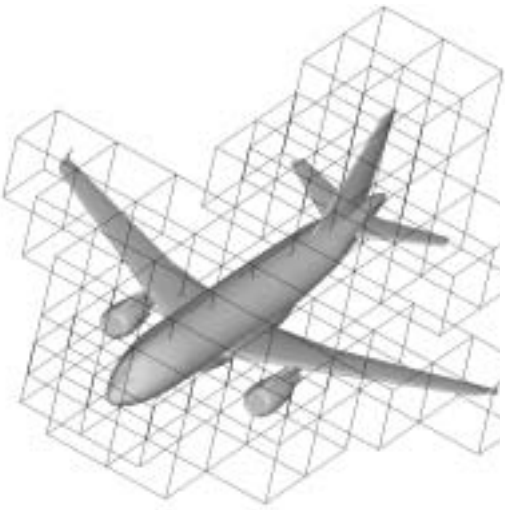
1.3.1.5.1 Les cellules voisines Comme dans le cas mono-niveau, on traite tout d'abord l'interaction de $\mathcal{C}'^{(1)}$ avec ses voisines via un produit matrice-vecteur classique. Le terme correspondant s'écrit :

$$\sum_{\mathcal{C}^{(1)}\in v(\mathcal{C}'^{(1)})} \int_{x\in\Gamma\cap\mathcal{C}^{(1)}} \int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} G(|y-x|)t(x)\varphi_j(y)dxdy$$

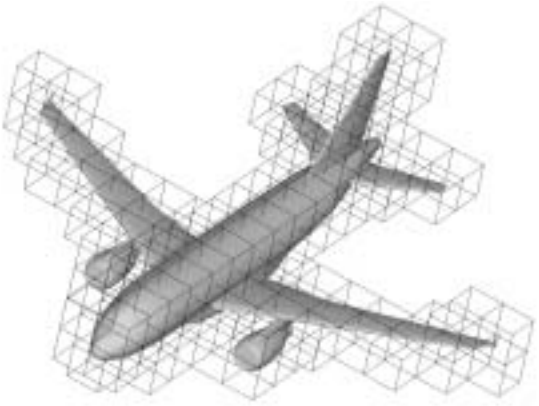
1.3.1.5.2 Les cellules banlieues On traite ensuite l'interaction de $\mathcal{C}'^{(1)}$ avec ses banlieues via une FMM au niveau 1. Le terme correspondant s'écrit :

$$\sum_{\mathcal{C}^{(1)}\in b(\mathcal{C}'^{(1)})} \int_{x\in\Gamma\cap\mathcal{C}^{(1)}} \int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} G(|y-x|)t(x)\varphi_j(y)dxdy$$

On utilise la décomposition du noyau (1.4), en mettant de côté la constante $ik/16\pi^2$, et en supposant L fixé. En réordonnant les intégrales, le terme précédent peut s'écrire :



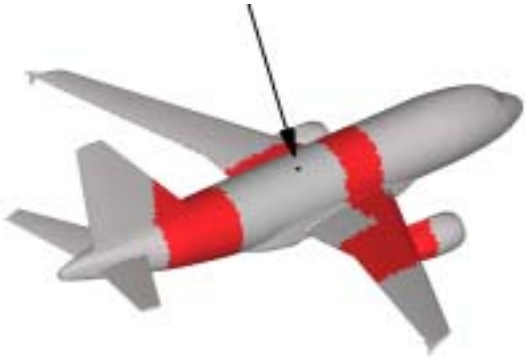
(a) Découpage du niveau 0



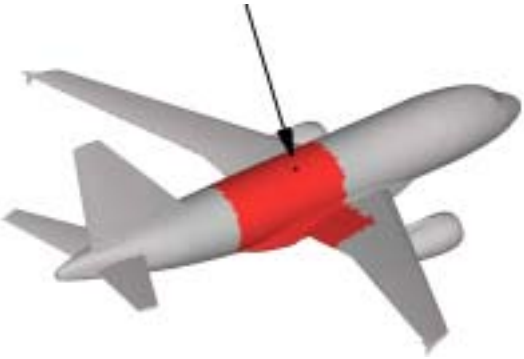
(b) Découpage du niveau 1

(c) Non-voisins au niveau 0

(d) Non-voisins au niveau 1



(e) Banlieues au niveau 1



(f) Voisins au niveau 1

FIG. 1.19 – *Méthode multipôle à deux niveaux sur un airbus A318*

$$\int_{\vec{s}\in\mathcal{S}}\int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} e^{ik\vec{s}.\vec{M'}y}\left[\sum_{\mathcal{C}^{(1)}\in b(\mathcal{C}'^{(1)})} T^{L(1)}_{\vec{M}\vec{M}'}(\vec{s})\left(\int_{x\in\Gamma\cap\mathcal{C}^{(1)}} e^{ik\vec{s}.x\vec{M}}t(x)dx\right)\right]\varphi_j(y)dyd\vec{s}$$

On retrouve les trois étapes de la FMM mono-niveau :

1. Initialisation des fonctions de radiation pour les cellules banlieues de $\mathcal{C}'^{(1)}$;
2. Transfert de ces fonctions vers $\mathcal{C}'^{(1)}$;
3. Intégration du résultat sur $\mathcal{C}'^{(1)}$.

1.3.1.5.3 Les cellules distantes Il reste à calculer l'intégrale pour $x \in \Gamma \cap d(\mathcal{C}'^{(1)})$. En utilisant (1.49), le terme d'interaction s'écrit :

$$\sum_{\mathcal{C}^{(0)}\notin v(p(\mathcal{C}'^{(1)}))}\int_{x\in\Gamma\cap\mathcal{C}^{(0)}}\int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} G(|y-x|)t(x)\varphi_j(y)dxdy$$

On utilise la décomposition du noyau (1.4) écrite au niveau 0 entre $\mathcal{C}^{(0)}$ centrée en $M^{(0)}$ et $\mathcal{C}'^{(0)}$ (parent de $\mathcal{C}'^{(1)}$) centrée en $M'^{(0)}$. On obtient :

$$\int_{\vec{s}\in\mathcal{S}}\int_{y\in\Gamma\cap\mathcal{C}'^{(1)}} e^{ik\vec{s}.\vec{M'^{(0)}}y}\left[\sum_{\mathcal{C}^{(0)}\notin v(p(\mathcal{C}'^{(1)}))} T^{L(0)}_{\vec{M^{(0)}}\vec{M'^{(0)}}}(\vec{s})\left(\int_{x\in\Gamma\cap\mathcal{C}^{(0)}} e^{ik\vec{s}.x\vec{M^{(0)}}}t(x)dx\right)\right]\varphi_j(y)dyd\vec{s} \tag{1.50}$$

On retrouve les trois étapes de la méthode multipôle au niveau 0, que l'on va légèrement adapter :

1. Initialisation des fonctions de radiation pour les cellules $\mathcal{C}^{(0)} \notin v(p(\mathcal{C}'^{(1)}))$. On va utiliser la formule suivante :

   $$\int_{x\in\Gamma\cap\mathcal{C}^{(0)}} e^{ik\vec{s}.x\vec{M^{(0)}}}t(x)dx = \sum_{\mathcal{C}^{(1)}\in e(\mathcal{C}^{(0)})} e^{ik\vec{s}.\vec{M^{(1)}M^{(0)}}}\left(\int_{x\in\Gamma\cap\mathcal{C}^{(1)}} e^{ik\vec{s}.x\vec{M^{(1)}}}t(x)dx\right)$$

   qui s'écrit plus simplement :

   $$\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s}) = \sum_{\mathcal{C}^{(1)}\in e(\mathcal{C}^{(0)})} e^{ik\vec{s}.\vec{M^{(1)}M^{(0)}}}\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s}) \tag{1.51}$$

   On va donc d'abord initialiser les fonctions de radiation $\mathcal{F}_{\mathcal{C}^{(1)}}$ au niveau 1, puis utiliser (1.51) pour remonter au niveau 0 et calculer $\mathcal{F}_{\mathcal{C}^{(0)}}$.
2. Transfert de ces fonctions vers $p(\mathcal{C}'^{(1)})$. Ces transferts ont lieu au niveau 0.
3. Intégration du résultat. On note $\mathcal{G}_{\mathcal{C}'^{(0)}}$ le terme entre crochets dans (1.50). Comme pour l'initialisation, on va d'abord changer de niveau en écrivant :

   $$\mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s}) = e^{ik\vec{s}.\vec{M'^{(0)}M'^{(1)}}}\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s}) \tag{1.52}$$

   Puis on intègre le résultat sur $\mathcal{C}'^{(1)}$ avec l'équation usuelle :

   $$\int_{y\in\Gamma\cap\mathcal{C}'^{(1)}}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s})e^{ik\vec{s}.\vec{M'^{(1)}}y}\varphi_j(y)d\vec{s}dy$$

1.3.1.5.4 Synthèse Le traitement des cellules distantes et des cellules banlieues met en jeu des phases d'initialisation et d'intégration au niveau 1 qui sont similaires, on va donc les mettre en commun. Mettant de côté les interactions proches, on voit apparaître une méthode multipôle bi-niveau en 6 étapes présentées figure 1.20.

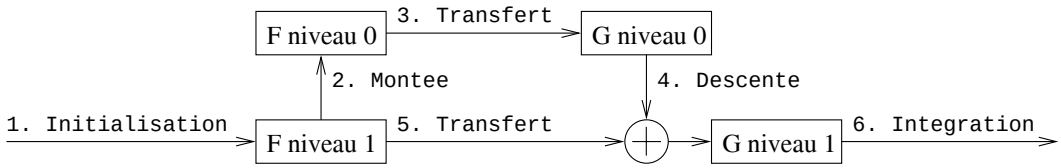


FIG. 1.20 – *Algorithme FMM à deux niveaux*

1. *Initialisation* des fonctions de radiation $\mathcal{F}_{\mathcal{C}^{(1)}}$ au niveau 1.
2. *Montée* : on calcule les fonctions de radiation $\mathcal{F}_{\mathcal{C}^{(0)}}$ au niveau 0 en sommant les contributions des enfants avec (1.51).
3. *Transfert* au niveau 0 : on calcule les fonctions de radiation $\mathcal{G}_{\mathcal{C}^{(0)}}$ sommant les contributions des cellules non-voisines.
4. *Descente* : on calcule la première partie de $\mathcal{G}_{\mathcal{C}^{(1)}}$ en descendant la contribution du parent avec (1.52).
5. *Transfert* au niveau 1 : on rajoute à $\mathcal{G}_{\mathcal{C}^{(1)}}$ la contribution des cellules banlieues. Ce rajout est symbolisé sur la figure 1.20 par le $\bigoplus$
6. *Intégration* au niveau 1 des fonctions $\mathcal{G}_{\mathcal{C}^{(1)}}$.

1.3.1.6 Montée et descente

1.3.1.6.1 Difficultés liées au changement de niveau L'algorithme multipôle bi-niveau met à jour deux nouveaux type d'opérations par rapport au cas mono-niveau : les montées et les descentes. Dans l'algorithme continu, ces équations se réduisent à un changement de centre, comme dans (1.51) et (1.52). Par exemple, une multiplication par $e^{ik\vec{s}.\overrightarrow{M^{(0)}M^{(1)}}}$ transforme une fonction de radiation attachée à une cellule centrée en $M^{(0)}$ en fonction centrée en $M^{(1)}$. On appellera cette opération une *translation*, le vecteur de translation étant bien sûr $\overrightarrow{M^{(0)}M^{(1)}}$ ici.

Lorsqu'on passe de l'algorithme continu à l'algorithme discret, on doit tenir compte du nombre de pôles aux niveaux 0 et 1, notés $L^{(0)}$ et $L^{(1)}$. Ce nombre de pôles est calculé à partir du pas du découpage, il dépend donc du niveau. Quelle que soit la manière choisie pour calculer L, on aura toujours $L^{(0)} > L^{(1)}$. Si on choisit d'utiliser la formule simple $L = kr$ (où $r = \sqrt{3}a$ est le diamètre d'une cellule), alors $L^{(0)}$ sera le double de $L^{(1)}$.

Si le changement de niveau s'accompagne d'un changement du nombre de pôles, il s'accompagne aussi d'un changement de discrétisation de la sphère unité $\mathcal{S}$. Cette dernière est choisie pour pouvoir intégrer exactement les fonctions de largeur de bande $2L$ (cf. section 1.2.2.2.2).

On note $(\vec{s_{k'}}^{(0)},\omega_{k'}^{(0)})$ et $(\vec{s_k}^{(1)},\omega_k^{(1)})$ les quadratures respectivement des niveaux 0 et 1. On doit donc être capable de passer de l'une à l'autre de ces quadratures de la manière la plus précise possible.

1.3.1.6.2 Extrapolation On appelle extrapolation le passage de la grille $(\vec{s_k}^{(1)},\omega_k^{(1)})$ à la grille $(\vec{s_{k'}}^{(0)},\omega_{k'}^{(0)})$. On oublie provisoirement le changement de centre qui va normalement de pair avec le passage d'une cellule à la cellule parent. Nous allons voir dans un premier temps l'algorithme basique pour réaliser cette opération, puis nous présenterons l'algorithme rapide utilisé dans notre implémentation FMM.

1.3.1.6.2.1 Algorithme basique Soit une cellule $\mathcal{C}^{(1)}$ au niveau 1, et la fonction de radiation $\mathcal{F}_{\mathcal{C}^{(1)}}$ définie par :

$$\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}^{(1)}} e^{ik\vec{s}.x\vec{M}^{(1)}} t(x)dx \tag{1.53}$$

On ne connaît cette fonction que par ses valeurs sur la grille $\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s_k}^{(1)})$. On sait néanmoins que c'est une fonction de largeur de bande $L^{(1)}/2$ (à une erreur ϵ près). En effet, à la section 1.2.2.2.2.1, on a tronqué la série (1.38) au rang $L^{(1)}$ dans le cas où $\vec{r}$ défini par (1.9) vérifiait seulement la condition (1.11) : $|\vec{r}| \leq \sqrt{3}a$ (où a est l'arête du découpage au niveau considéré). Ici le vecteur dans l'exponentielle relie un point de $\mathcal{C}^{(1)}$ à son centre, donc il vérifie la condition plus restrictive : $|x\vec{M}^{(1)}| \leq \sqrt{3}a/2$. On peut alors tronquer la série au rang $L^{(1)}/2$ tout en conservant la même erreur ϵ.

Partant de là, on peut écrire $\mathcal{F}_{\mathcal{C}^{(1)}}$ sous la forme :

$$\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(1)}/2}} A_{l,m}Y_{l,m}(\vec{s})$$

Pour calculer les coefficients $A_{l,m}$, il suffit d'utiliser l'orthonormalité des harmoniques sphériques, qui permet d'écrire :

$$A_{l,m} = \int_{\vec{s}\in\mathcal{S}} \mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s})Y^*_{l,m}(\vec{s})d\vec{s} \tag{1.54}$$

On a au niveau 1 une quadrature de $\mathcal{S}$ qui intègre exactement les fonctions de $\mathcal{L}^2(\mathcal{S})$ de largeur de bande $\leq 2L^{(1)}$. Ici, la fonction sous le signe intégrale est de largeur de bande seulement $L^{(1)}/2 + l \leq L^{(1)}$. Donc notre quadrature l'intègre exactement, et on a même « de la marge ». On va utiliser cette marge, et considérer désormais $\mathcal{F}_{\mathcal{C}^{(1)}}$ comme une fonction de largeur de bande $L^{(1)}$. On y gagne en précision, et on verra par la suite qu'il n'y a pas de surcoût associé à ce changement. La fonction intégrée $\mathcal{F}_{\mathcal{C}^{(1)}}Y^*_{l,m}$ est alors de largeur de bande $L^{(1)} + l \leq 2L^{(1)}$, et notre quadrature l'intègre exactement :

$$A_{l,m} = \sum_{\vec{s_k}^{(1)}} \omega_k^{(1)}\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s_k}^{(1)})Y^*_{l,m}(\vec{s_k}^{(1)}) \tag{1.55}$$

Une fois calculés les coefficients $(A_{l,m})$ pour $-l \leq m \leq l$ et $0 \leq l \leq L^{(1)}$, on peut calculer $\mathcal{F}_{\mathcal{C}^{(0)}}$ sur la grille du niveau 0 en écrivant tout simplement :

$$\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s_{k'}}^{(0)}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(1)}}} A_{l,m}Y_{l,m}(\vec{s_{k'}}^{(0)}) \tag{1.56}$$

Sachant que $\mathcal{F}_{\mathcal{C}^{(0)}}$ est de largeur de bande $L^{(0)}$, cette équation revient à compléter les coefficients $(A_{l,m})$ par des zéros pour $L^{(1)} < l \leq L^{(0)}$. En insérant (1.55) dans (1.56), on obtient :

$$\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s_{k'}}^{(0)}) = \sum_{\vec{s_k}^{(1)}} \left[\sum_{\substack{-l \leq m \leq l \\ 0 \leq l \leq L^{(1)}}} Y^*_{l,m}(\vec{s_k}^{(1)}) Y_{l,m}(\vec{s_{k'}}^{(0)}) \right] \omega_k^{(1)} \mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s_k}^{(1)}) \tag{1.57}$$

Cette opération est donc un simple produit matrice-vecteur, par une matrice dont le nombre de colonnes est le nombre de points de quadrature au niveau 1, et dont le nombre de lignes est le nombre de points de quadrature au niveau 0.

Cette matrice est plus simple qu'il n'y paraît, puisque le terme entre crochets peut s'écrire :

$$\begin{aligned} \sum_{\substack{-l \leq m \leq l \\ 0 \leq l \leq L^{(1)}}} Y^*_{l,m}(\vec{s_k}^{(1)}) Y_{l,m}(\vec{s_{k'}}^{(0)}) &= \sum_{0 \leq l \leq L^{(1)}} \frac{2l+1}{4\pi} P_l(\cos\theta) \\ &= \frac{L^{(1)}+1}{4\pi(1-\cos\theta)} \left(P_{L^{(1)}}(\cos\theta) - P_{L^{(1)}+1}(\cos\theta) \right) \end{aligned} \tag{1.58}$$

où l'on note θ l'angle que font les vecteurs unitaires $\vec{s_{k'}}^{(0)}$ et $\vec{s_k}^{(1)}$:

$$\cos\theta = \vec{s_{k'}}^{(0)}.\vec{s_k}^{(1)}$$

La deuxième formule ci-dessus n'est valable que si $\cos\theta \neq 1$. Le nombre d'opérations de cette extrapolation est égal au produit des tailles des quadratures de $\mathcal{S}$ aux niveau 0 et 1, soit environ $4(L^{(0)}L^{(1)})^2$ avec notre discrétisation usuelle (définie à la section 1.2.2.2.2). Cette formulation de l'extrapolation est la plus simple à implémenter, on va maintenant voir une méthode pour accélérer ces calculs.

1.3.1.6.2.2 Algorithme rapide La formulation précédente fonctionne quel que soit le type de quadrature $(\vec{s_k},\omega_k)$ utilisé. Il existe une méthode plus rapide pour réaliser l'extrapolation dans le cas où les points $\vec{s_k}$ sur $\mathcal{S}$ forment une grille de la forme (θ_i,ϕ_j) qui est en plus uniforme en ϕ (c'est à dire de la forme $\phi_j = 2\pi j/K \qquad 0 \leq j < K$). Cette condition est vérifiée à la fois par la discrétisation basique et par la discrétisation usuelle présentées à la section 1.2.2.2.2. En revanche, si on utilise une discrétisation optimale (comme celle de McLaren à 72 points), ce qui va suivre ne s'applique pas.

On va tenter de simplifier (1.57). On note de la manière suivante les grilles de points de quadrature aux niveaux 0 et 1 :

$$\begin{cases} (\theta_{i'}^{(0)},\phi_{j'}^{(0)}) & 0 \leq i' \leq L^{(0)}, \qquad 0 \leq j' \leq 2L^{(0)}, \\ (\theta_i^{(1)},\phi_j^{(1)}) & 0 \leq i \leq L^{(1)}, \qquad 0 \leq j \leq 2L^{(1)} \end{cases}$$

On note $F_{i,j}^{(1)}$ et $F_{i',j'}^{(0)}$ les vecteurs en entrée et en sortie du produit matrice-vecteur, avec :

$$\begin{cases} F_{i,j}^{(1)} = \omega_k^{(1)} \mathcal{F}_{\mathcal{C}^{(1)}}(\theta_i^{(1)},\phi_j^{(1)}), \\ F_{i',j'}^{(0)} = \mathcal{F}_{\mathcal{C}^{(0)}}(\theta_{i'}^{(0)},\phi_{j'}^{(0)}) \end{cases}$$

où dans la première équation $\omega_k^{(1)}$ est le poids associé au point de quadrature $(\theta_i^{(1)},\phi_j^{(1)})$ considéré. Vis-à-vis du produit matrice-vecteur, $F_{i,j}^{(1)}$ et $F_{i',j'}^{(0)}$ sont des vecteurs. Néanmoins, nous les considérerons dans la suite plutôt comme des tableaux à deux dimensions puisqu'ils représentent les valeurs des fonctions de radiation $\mathcal{F}_\mathcal{C}$ sur les grilles (θ,ϕ).

Enfin, les harmoniques sphériques s'écrivent :

$$\begin{cases} Y_{l,m}(\theta,\phi) = C_{l,m}P_l^m(cos\theta)e^{im\phi}, \\ C_{l,m} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} \end{cases}$$

Pour alléger les notations, on note $Q_l^m(cos\theta)$ le polynôme $C_{l,m}P_l^m(cos\theta)$. Le produit matrice-vecteur (1.57) devient :

$$\begin{aligned} F_{i',j'}^{(0)} &= \sum_{i,j}\left[\sum_{0\le l\le L^{(1)}}\sum_{-l\le m\le l} Q_l^m(cos\theta_i^{(1)})e^{-im\phi_j^{(1)}}Q_l^m(cos\theta_{i'}^{(0)})e^{im\phi_{j'}^{(0)}}\right]F_{i,j}^{(1)} \\ &= \sum_{-L^{(1)}\le m\le L^{(1)}} e^{im\phi_{j'}^{(0)}}\left[\sum_i\sum_{|m|\le l\le L^{(1)}} Q_l^m(cos\theta_i^{(1)})Q_l^m(cos\theta_{i'}^{(0)})\left(\sum_j e^{-im\phi_j^{(1)}}F_{i,j}^{(1)}\right)\right] \end{aligned}$$

En réordonnant ainsi les termes, on voit apparaître trois étapes dans ce calcul.

Fourier directe : on fait une transformée de Fourier directe sur les lignes de $F^{(1)}$. On note $\tilde{F}_{i,m}^{(1)}$ le résultat avec $-L^{(1)} \le m \le L^{(1)}$:

$$\tilde{F}_{i,m}^{(1)} = \sum_{0\le j\le 2L^{(1)}} e^{-im\phi_j^{(1)}}F_{i,j}^{(1)}$$

Produits matrice-vecteur : pour i' et m fixés, on note $\tilde{F}_{i',m}^{(0)}$ le terme entre crochets :

$$\tilde{F}_{i',m}^{(0)} = \sum_i\left[\sum_{|m|\le l\le L^{(1)}} Q_l^m(cos\theta_i^{(1)})Q_l^m(cos\theta_{i'}^{(0)})\right]\tilde{F}_{i,m}^{(1)} \tag{1.59}$$

On voit apparaître un produit matrice-vecteur par la matrice dont l'élément (i',i) est entre crochets. Malheureusement, cette matrice dépend de m. La formule de Christoffel-Darboux qui suit va permettre d'ôter cette dépendance. Pour tout $(x,x') \in [-1,1]^2$, avec $x \ne x'$, on a :

$$\begin{aligned} \sum_{|m|\le l\le L^{(1)}} Q_l^m(x)Q_l^m(x') &= \sqrt{\frac{(L^{(1)}+1)^2-m^2}{4(L^{(1)}+1)^2-1}} \\ &\times\left[\frac{Q_{L^{(1)}+1}^m(x')Q_{L^{(1)}}^m(x)}{x'-x} - \frac{Q_{L^{(1)}+1}^m(x)Q_{L^{(1)}}^m(x')}{x'-x}\right] \end{aligned} \tag{1.60}$$

On pose :

$$\begin{cases} x_i = cos\theta_i^{(1)}, \\ x_{i'} = cos\theta_{i'}^{(0)}, \\ C_{i',i} = \dfrac{1}{x_{i'}-x_i} \end{cases}$$

Pour simplifier, dans (1.60), on va mettre de côté la constante et ne considérer que le premier terme entre crochets. Avec ces restrictions, la somme (1.59) (que l'on continue à noter $\tilde{F}^{(0)}_{i',m}$ pour ne pas allourdir davantage les formules) se calcule de la manière suivante :

$$\begin{aligned}\tilde{F}^{(0)}_{i',m} &= \sum_i \left[Q^m_{L^{(1)}+1}(x_{i'})Q^m_{L^{(1)}}(x_i)C_{i',i}\right]\tilde{F}^{(1)}_{i,m} \\ &= Q^m_{L^{(1)}+1}(x_{i'})\left[\sum_i C_{i',i}\left(Q^m_{L^{(1)}}(x_i)\tilde{F}^{(1)}_{i,m}\right)\right]\end{aligned} \tag{1.61}$$

Autrement dit, on a trois étapes :

1. On multiplie $F^{(1)}_{i,m}$ par $Q^m_{L^{(1)}}(x_i)$ composante par composante (m fixé, $i = 0, \ldots, L^{(1)}$).
2. On fait le produit du résultat par la matrice C de taille $(L^{(0)}+1)\times(L^{(1)}+1)$.
3. On multiplie le résultat par $Q^m_{L^{(1)}+1}(x_{i'})$ composante par composante (m fixé, $i' = 0, \ldots, L^{(0)}$).

L'avantage par rapport à la formulation (1.59) est que la matrice $(L^{(0)}+1)\times(L^{(1)}+1)$ est indépendante de m et ne dépend que des nombres de pôles $L^{(0)}$ et $L^{(1)}$. On ne gagne donc ni en précision, ni en vitesse, mais en stockage : c'est donc plutôt un algorithme de compression de matrice qu'une accélération du produit (1.59). L'autre avantage est que l'on peut calculer les produits matrice-vecteur de C avec une FMM 1D. Néanmoins, nous n'avons pas réalisé cette amélioration car en l'état actuel de notre implémentation, les phases de montées/descentes ont un coût faible par rapport aux autres parties du calcul, et l'effort nécessaire n'était pas justifié.

Fourier inverse : on fait une transformée de Fourier inverse sur les lignes du résultat pour $j' = 0, \ldots, 2L^{(0)}$:

$$F^{(0)}_{i',j'} = \sum_{-L^{(1)}\leq m\leq L^{(1)}} e^{im\phi^{(0)}_{j'}}\tilde{F}^{(0)}_{i',m}$$

On a une transformée de Fourier de taille $2L^{(0)}+1$. Le fait que $-L^{(1)} \leq m \leq L^{(1)}$ n'est pas un problème, cela revient juste à dire que les coefficients $\tilde{F}^{(0)}_{i',m}$ pour $m = -L^{(0)}, \ldots, -L^{(1)}-1$ et $m = L^{(1)}+1, \ldots, L^{(0)}$ sont nuls.

En terme de nombre d'opérations, la partie la plus coûteuse de cette extrapolation rapide est le produit par la matrice C, de taille environ $L^{(0)}L^{(1)}$, on fait 2 produits pour chaque $m = -L^{(1)}, \ldots, L^{(1)}$ soit en tout environ $4L^{(0)}(L^{(1)})^2$ opération, contre $4(L^{(0)}L^{(1)})^2$ opérations avec l'extrapolation basique. Avec $L^{(0)} \geq 10$ au minimum, le gain est notable.

1.3.1.6.3 Réduction On appelle réduction le passage de la grille $(\vec{s_{k'}}^{(0)},\omega^{(0)}_{k'})$ à la grille $(\vec{s_k}^{(1)},\omega^{(1)}_k)$. Comme pour l'extrapolation, on met entre parenthèse le changement de centre qui va normalement de pair avec le passage d'une cellule à une des cellules enfants.

Soit une cellule $\mathcal{C}'^{(0)}$ au niveau 0, et la fonction de radiation $\mathcal{G}_{\mathcal{C}'^{(0)}}$ définie par :

$$\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s}) = \sum_{\mathcal{C}^{(0)}\notin v(\mathcal{C}'^{(0)})} T^{L^{(0)}}_{M^{(0)}\vec{M}'^{(0)}}(\vec{s})\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s})$$

On ne connaît cette fonction que par ses valeurs sur la grille $\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s_{k'}}^{(0)})$. On sait néanmoins que c'est une fonction de largeur de bande $L^{(0)}$ (à une erreur ϵ près). En effet, la fonction de transfert $T^{L^{(0)}}_{M^{(0)}\vec{M}'^{(0)}}$ est de largeur de bande $L^{(0)}$, donc on peut négliger les harmoniques de rang supérieur.

Partant de là, on peut écrire $\mathcal{G}_{\mathcal{C}'^{(0)}}$ sous la forme :

$$\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(0)}}} A_{l,m}Y_{l,m}(\vec{s})$$

Pour calculer les coefficients $A_{l,m}$, il suffit d'utiliser l'orthonormalité des harmoniques sphériques, qui permet d'écrire :

$$A_{l,m} = \int_{\vec{s}\in\mathcal{S}} \mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s})Y^*_{l,m}(\vec{s})d\vec{s} \tag{1.62}$$

La fonction à calculer $\mathcal{G}_{\mathcal{C}'^{(1)}}$ étant amenée a être ajoutée au résultat des transferts au niveau 1 (de largeur de bande $L^{(1)}$) avant d'être intégrée, seuls les coefficients $A_{l,m}$ avec $0 \leq l \leq L^{(1)}$ nous intéressent. Sous cette condition, dans (1.62) la fonction sous le signe intégrale est de largeur de bande maximale $L^{(0)} + L^{(1)} < 2L^{(0)}$, donc notre quadrature au niveau 0 l'intègre exactement :

$$A_{l,m} = \sum_{\vec{s_{k'}}^{(0)}} \omega^{(0)}_{k'}\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s_{k'}}^{(0)})Y^*_{l,m}(\vec{s_{k'}}^{(0)}) \tag{1.63}$$

Une fois calculés les coefficients $(A_{l,m})$ pour $-l \leq m \leq l$ et $0 \leq l \leq L^{(1)}$, on peut calculer $\mathcal{G}_{\mathcal{C}'^{(1)}}$ sur la grille du niveau 1 en écrivant tout simplement :

$$\mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s_k}^{(1)}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(1)}}} A_{l,m}Y_{l,m}(\vec{s_k}^{(1)}) \tag{1.64}$$

Sachant que $\mathcal{G}_{\mathcal{C}'^{(0)}}$ est de largeur de bande $L^{(0)}$, cette équation revient à annuler les coefficients $(A_{l,m})$ pour $L^{(1)} < l \leq L^{(0)}$. En insérant (1.63) dans (1.64), on obtient :

$$\mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s_k}^{(1)}) = \sum_{\vec{s_{k'}}^{(0)}} \left[\sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(1)}}} Y^*_{l,m}(\vec{s_{k'}}^{(0)})Y_{l,m}(\vec{s_k}^{(1)}) \right] \omega^{(0)}_{k'}\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s_{k'}}^{(0)})$$

Cette opération est donc un produit matrice-vecteur par la matrice transposée de celle utilisée pour les montées.

On peut bien sûr également utiliser une variante « transposée » de l'algorithme rapide utilisé pour les extrapolations (en échangeant les étapes 1 et 3 du calcul de (1.61), et en utilisant ${}^tC$ à la place de C).

1.3.1.6.4 Formulation à deux composantes Lors d'un calcul d'extrapolation, la fonction de radiation définie par (1.53) a la même largeur de bande que la fonction $e^{ik\vec{s}.x\vec{M}^{(1)}}$. Cela est vrai parce qu'on multiplie cette exponentielle par $t(x)$ qui ne dépend pas de $\vec{s}$. Dans le cas d'une formulation à deux composantes, le terme $t(x)$ est remplacé par l'un des deux termes :

$$\left\{\begin{array}{rcl} \vec{t}(x).\vec{s_\theta} & = & t_x(x)\cos\theta\cos\phi + t_y(x)\cos\theta\sin\phi - t_z(x)\sin\theta, \\ \vec{t}(x).\vec{s_\phi} & = & -t_x(x)\sin\phi + t_y(x)\cos\phi \end{array}\right.$$

Pour que les développements réalisés aux paragraphes précédents restent valables, il faut que les termes dépendants de θ et de ϕ rajoutés soient de largeur de bande finie. On doit donc essayer de développer sur la base des harmoniques sphériques les quantités suivantes :

$$X = \left[\begin{array}{c} \cos\theta\cos\phi \\ \cos\theta\sin\phi \\ \sin\theta \\ \sin\phi \\ \cos\phi \end{array}\right] \tag{1.65}$$

Rappelons les relations permettant de calculer les harmoniques sphériques. Pour $m \geq 0$, on a :

$$\left\{\begin{array}{l} Y_{l,m}(\theta,\phi) = C_{l,m}P_l^m(cos\theta)e^{im\phi}, \\ C_{l,m} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}, \\ P_l^m(cos\theta) = (-1)^m(\sin\theta)^m(\dfrac{d}{dx})^m P_l(\cos\theta) \end{array}\right.$$

et pour $m < 0$, on utilise : $Y_{l,-m}(\theta,\phi) = (-1)^m Y^*_{l,m}(\theta,\phi)$ (où * désigne le complexe conjugué).

$Y_{l,m}(cos\theta)$ apparaît donc comme étant le produit d'un polynôme en $\cos\theta$ de degré $(l-|m|)$ par $(\sin\theta)^{|m|}e^{im\phi}$. On peut vérifier sur les premières harmoniques sphériques :

$$\left\{\begin{array}{rcl} Y_{0,0}(\theta,\phi) & = & \sqrt{\frac{1}{4\pi}}, \\ Y_{1,-1}(\theta,\phi) & = & \sqrt{\frac{3}{8\pi}}\sin\theta e^{-i\phi}, \\ Y_{1,0}(\theta,\phi) & = & \sqrt{\frac{3}{4\pi}}\cos\theta, \\ Y_{1,1}(\theta,\phi) & = & -\sqrt{\frac{3}{8\pi}}\sin\theta e^{i\phi}, \\ Y_{2,-2}(\theta,\phi) & = & \sqrt{\frac{15}{8\pi}}\sin^2\theta e^{-2i\phi}, \\ Y_{2,-1}(\theta,\phi) & = & \sqrt{\frac{15}{2\pi}}cos\theta\sin\theta e^{-i\phi}, \\ Y_{2,0}(\theta,\phi) & = & \sqrt{\frac{5}{4\pi}}\frac{3cos^2\theta-1}{2}, \\ Y_{2,1}(\theta,\phi) & = & -\sqrt{\frac{15}{2\pi}}cos\theta\sin\theta e^{i\phi}, \\ Y_{2,2}(\theta,\phi) & = & \sqrt{\frac{15}{8\pi}}\sin^2\theta e^{2i\phi} \end{array}\right.$$

Les composantes du vecteur (1.65) ne peuvent pas se décomposer simplement sur cette base, car il faut que l'exposant en $\sin\theta$ et en $e^{\pm i\phi}$ soit le même. En revanche, le vecteur

$\sin\theta.X$ vérifie cette propriété :

$$\sin\theta.X = \begin{bmatrix} \sin\theta\cos\theta\cos\phi \\ \sin\theta\cos\theta\sin\phi \\ \sin^2\theta \\ \sin\theta\sin\phi \\ \sin\theta\cos\phi \end{bmatrix} = \begin{bmatrix} -\sqrt{\frac{2\pi}{15}}(Y_{2,1}-Y_{2,-1})/2 \\ -\sqrt{\frac{2\pi}{15}}(Y_{2,1}+Y_{2,-1})/2i \\ \frac{2}{3}(\sqrt{4\pi}Y_{0,0}-\sqrt{\frac{4\pi}{5}}Y_{2,0}) \\ -\sqrt{\frac{8\pi}{3}}(Y_{1,1}+Y_{1,-1})/2i \\ -\sqrt{\frac{8\pi}{3}}(Y_{1,1}-Y_{1,-1})/2 \end{bmatrix}$$

Les composantes du vecteur $\sin\theta.X$ sont de largeur de bande 2. Donc la fonction de radiation $\mathcal{F}_{\mathcal{C}^{(1)}}$ définie par :

$$\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}^{(1)}} e^{ik\vec{s}.x\vec{M}^{(1)}} \sin\theta\vec{t}(x).\vec{s_\theta}dx$$

est de largeur de bande finie $L^{(1)}+2$. La méthode utilisée dans le cas classique peut alors être utilisée. Il faut juste penser à rediviser par $\sin\theta$ après l'extrapolation.

En toute rigueur, vu que la largeur de bande des fonctions manipulées augmente, il faudrait augmenter le nombre de points de quadrature de $\mathcal{S}$ en conséquence. En pratique, on n'observe aucune dégradation de précision si on conserve la même grille. Cela n'est guère surprenant dans la mesure où les choix effectués pour les nombres de pôles et les quadratures sont en partie empiriques, et relève autant d'un savoir-faire que de la théorie.

Numériquement, avant de faire le produit matrice-vecteur (1.57) (ou son équivalent rapide), il suffit de multiplier le vecteur en entrée $\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s_k}^{(1)})$ par $\sin\theta_k^{(1)}$ (composante par composante), et de diviser les composantes du vecteurs résultats $\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s_{k'}}^{(0)})$ par $\sin\theta_{k'}^{(0)}$. Cela revient aussi à multiplier l'élément $(\vec{s_{k'}}^{(0)},\vec{s_k}^{(1)})$ de la matrice utilisée par $\frac{\sin\theta_k^{(1)}}{\sin\theta_{k'}^{(0)}}$.

La réduction étant l'opération transposée, il convient de diviser par $\sin\theta_{k'}^{(0)}$ la fonction de radiation en entrée du produit matrice-vecteur, et de multiplier par $\sin\theta_k^{(1)}$ le résultat du produit. Bien sûr, on peut aussi utiliser la matrice modifiée transposée.

1.3.1.6.5 Changement de centre Le changement de niveau dans l'algorithme multipôle associe toujours un changement de quadrature de $\mathcal{S}$ et un changement de centre de radiation (On a choisit d'appeler « translation » l'opération de changement de centre). Dans le cas d'une montée, cela se voit dans (1.51). Pour une descente, c'est la formule (1.52). A chaque fois, on a la possibilité de faire la translation avant ou après le changement de grille (extrapolation ou réduction).

Si on note $\mathcal{X}$ l'opérateur d'extrapolation, la montée peut s'écrire de l'une des deux manières suivantes :

$$\begin{cases} \mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s}) = \mathcal{X}\left[\sum_{\mathcal{C}^{(1)}\in e(\mathcal{C}^{(0)})} e^{ik\vec{s}.M^{(1)}\vec{M}^{(0)}}\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s})\right], \\ \mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s}) = \sum_{\mathcal{C}^{(1)}\in e(\mathcal{C}^{(0)})} e^{ik\vec{s}.M^{(1)}\vec{M}^{(0)}}\mathcal{X}\left[\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s})\right] \end{cases}$$

La première équation correspond au schéma de gauche sur la figure 1.21, la seconde au schéma de droite. Une translation augmente la largeur de bande de la fonction translatée, donc

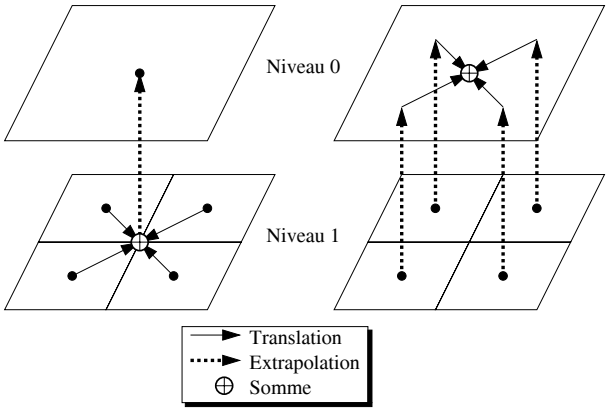


FIG. 1.21 – *Phase de montée*

si la translation précède le changement de grille, le calcul est en théorie moins précis que si on fait le contraire. On pourrait compenser en augmentant le nombre d'harmoniques sphériques conservées, mais là encore, en pratique, on ne voit pas de différence.

Il y a en revanche une nette différence en terme de temps d'exécution. Dans l'algorithme multipôle, l'opération « montée » envoie toutes les fonctions de radiation des enfants de $\mathcal{C}^{(0)}$ vers celle-ci. On peut donc :

- soit translater chaque enfant puis sommer et extrapoler le résultat ;
- soit extrapoler chaque enfant puis translater et sommer.

La différence est dans le nombre d'extrapolations (la phase la plus coûteuse ici). La première méthode met en œuvre une extrapolation pour tous les enfants, contre une extrapolation par enfant dans la seconde. Cette dernière est donc sensiblement plus lente. De même, dans le cas de la descente, on a intérêt à réduire la fonction de radiation du parent avant de la translater vers chacun des enfants.

1.3.1.7 Algorithme à deux niveaux

Nous allons récapituler tous les points évoqués dans cette section, et écrire complètement l'algorithme multipôle à deux niveaux pour la formulation EFIE. On se donne un courant surfacique scalaire $t(x)$ en entrée, et un découpage de Γ à travers deux grilles imbriquées. Le produit $A.t$ se réalise en deux parties :

Interactions proches : elles sont traitées au niveau 1, exactement comme dans le cas mono-niveau (cf. équation (1.12)).

Interactions Lointaines : le calcul se fait en six étapes.

1. **Initialisation :** pour toute cellule $\mathcal{C}^{(1)}$ du niveau 1, on calcule la fonction de radiation

$$\mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}^{(1)}} e^{ik\vec{s}.x\vec{M}^{(1)}} t(x)dx$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau 1.

2. **Montée :** pour toute cellule $\mathcal{C}^{(0)}$ du niveau 0, on calcule la fonction de radiation $\mathcal{F}_{\mathcal{C}^{(0)}}$ à partir de celles des enfants :

$$\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s}) = \sum_{\mathcal{C}^{(1)} \in e(\mathcal{C}^{(0)})} e^{ik\vec{s}.\overrightarrow{M^{(1)}M^{(0)}}} \mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s})$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau 0.

3. **Transfert au niveau 0 :** pour toute cellule $\mathcal{C}'^{(0)}$ fixée, on passe en revue les cellules $\mathcal{C}^{(0)}$ non-voisine de $\mathcal{C}'^{(0)}$ pour calculer :

$$\mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s}) = \sum_{\mathcal{C}^{(0)} \notin v(\mathcal{C}'^{(0)})} T^{L^{(0)}}_{\overrightarrow{M^{(0)}M'^{(0)}}}(\vec{s}).\mathcal{F}_{\mathcal{C}^{(0)}}(\vec{s})$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau 0.

4. **Descente :** pour toute cellule $\mathcal{C}'^{(1)}$ du niveau 1, on calcule la première partie de la fonction de radiation $\mathcal{G}_{\mathcal{C}'^{(1)}}$ à partir du parent :

$$\mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s}) = e^{ik\vec{s}.\overrightarrow{M'^{(0)}M'^{(1)}}} \mathcal{G}_{\mathcal{C}'^{(0)}}(\vec{s})$$

5. **Transfert au niveau 1 :** pour toute cellule $\mathcal{C}'^{(1)}$ du niveau 1, on calcule la deuxième partie de la fonction de radiation $\mathcal{G}_{\mathcal{C}'^{(1)}}$ en passant en revue les cellules banlieues $\mathcal{C}^{(1)}$ pour calculer :

$$\sum_{\mathcal{C}^{(1)} \in b(\mathcal{C}'^{(1)})} T^{L^{(1)}}_{\overrightarrow{M^{(1)}M'^{(1)}}}(\vec{s}) \mathcal{F}_{\mathcal{C}^{(1)}}(\vec{s})$$

sur la quadrature de $\mathcal{S}$ associée au nombre de pôles $L^{(1)}$. Cette somme se rajoute à la partie de $\mathcal{G}_{\mathcal{C}'^{(1)}}$ calculée dans la phase « descente ».

6. **Intégration :** pour toute cellule $\mathcal{C}'^{(1)}$, et pour toute fonction de base φ_j localisée dans $\mathcal{C}'^{(1)}$, on calcule l'intégrale :

$$\frac{ik}{16\pi^2} \int_{y \in \Gamma \cap \mathcal{C}'^{(1)}} \int_{\vec{s} \in \mathcal{S}} \mathcal{G}_{\mathcal{C}'^{(1)}}(\vec{s}) e^{ik\vec{s}.\overrightarrow{M'^{(1)}y}} \varphi_j(y) d\vec{s} dy$$

Le résultat de cette intégration constitue la partie « interaction lointaine » de la j-ième composante du vecteur $A.t$, et se rajoute naturellement à la partie « interaction proche ».

Si on veut utiliser une formulation multipôle à deux composantes, il faut penser à modifier les montées-descentes (comme vu précédemment), et rajouter les conversions cartésiens/sphériques dans les initialisations/intégrations. Pour la MFIE, seule l'intégration change. Dans tous les cas, la structure générale de l'algorithme reste la même.

1.3.2 Approfondissement

L'étude de la FMM à deux niveaux a permis de souligner les principales nouveautés de l'algorithme multi-niveau par rapport à la version mono-niveau. Nous allons maintenant étudier le cas général d'une FMM à n niveaux ($n \geq 2$).

1.3.2.1 Construction d'un octree

1.3.2.1.1 Méthode Pour construire les structures dont on a besoin, l'idée est de procéder de manière récursive. On va simultanément créer une suite de découpages imbriqués et une structure d'arbre associée. Le découpage de Γ en grilles imbriquées est représenté en version 2D sur la figure 1.22, et en 3D sur la figure 1.23.

On commence donc par créer le niveau 0 du découpage en englobant Γ dans une boîte cubique suffisamment grande. Cette boîte est ensuite divisée en huit boîtes identiques. On ne conserve alors que celles de ces boîtes ayant une intersection non-vide avec Γ. Elles constituent le niveau 1 de l'arbre. On répète ce processus de division pour obtenir le niveau 2 de l'arbre. A chaque nouveau niveau créé, la taille des boîtes est divisée par deux par rapport au niveau précédent.

On arrête d'itérer soit quand on a construit un nombre de niveaux fixé par avance, soit lorsque la taille des boîtes atteint un certain seuil. Dans le cadre d'une méthode multipôle, ce critère s'exprime en nombre de longueurs d'onde. Une troisième méthode d'arrêt consiste à fixer la taille des plus petites boîtes, puis à multiplier celle-ci par deux jusqu'à dépasser la taille de l'objet. On détermine ainsi la taille de la boîte initiale, ainsi que le nombre de niveaux. C'est la méthode que l'on a choisie dans notre implémentation. La taille des plus petites boîtes est un paramètre important dans la précision de la méthode, il est donc utile de pouvoir la contrôler.

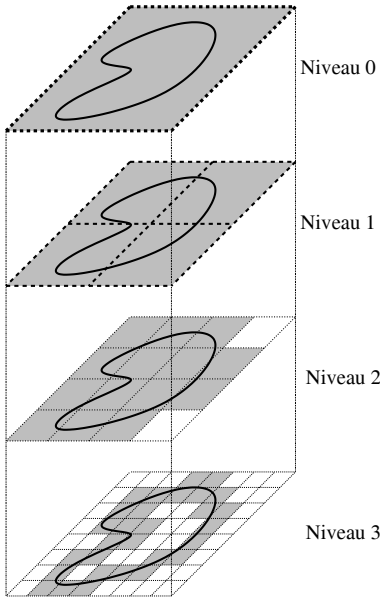


FIG. 1.22 – *Découpage multi-niveau de* Γ *(version 2D)*

La figure 1.24 représente le résultat d'un découpage multi-niveau appliqué à un maillage d'airbus A318 comportant 23676 inconnues. Les boîtes ont des arêtes dont la dimension varie de 16 longueurs d'onde (au niveau 0) à une demi longueur d'onde (au niveau 5). Le niveau 6 n'a pas été représenté, bien qu'il soit utilisé en pratique.

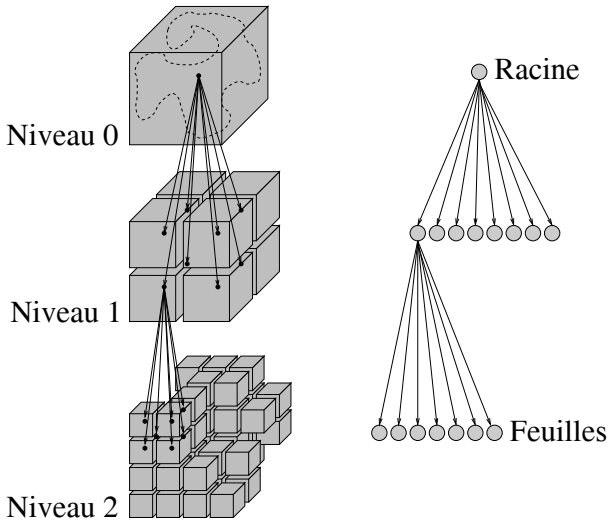


FIG. 1.23 – *Découpage multi-niveau de* Γ *(version 3D) et arbre associé*

1.3.2.1.2 Quelques points de vocabulaire Dans l'arbre obtenu, chaque cellule possède au plus huit enfants, d'où le nom d'octree. L'équivalent 2D représenté figure 1.22 s'appelle un quadtree. Il n'y a aucun quadtree dans notre code FMM, on s'en sert ici dans un but purement illustratif. Le niveau 0 comporte une seule cellule, c'est la « racine » de l'arbre. Les cellules du dernier niveau s'appellent les « feuilles » de l'arbre. Comme tout arbre qui se respecte, la racine constitue le « haut » de l'arbre, et les feuilles le « bas ». On « remonte » dans l'arbre lorsque le numéro de niveau décroît, et inversement une descente correspond à une incrémentation du numéro de niveau.

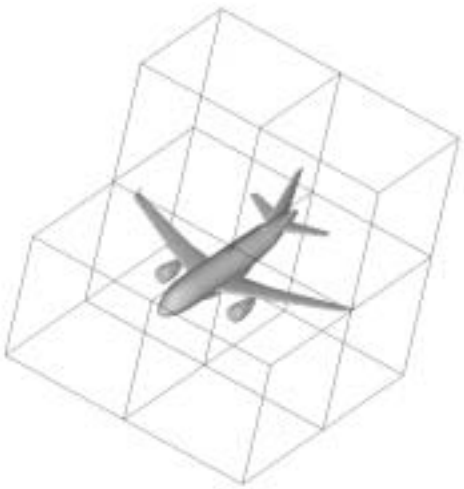
Enfants : on appelle toujours « enfants » de $\mathcal{C}$, et l'on note $e(\mathcal{C})$ les cellules issues de la subdivision de $\mathcal{C}$. Une cellule possède au plus huit enfants. Les feuilles n'ont pas d'enfant, les autres cellules ont toujours au moins un enfant (sinon cela impliquerait que cette cellule ne coupe pas Γ).

Parent : c'est bien sûr la relation inverse. Toutes les cellules ont exactement un parent, sauf la racine qui n'en a pas. On note $p(\mathcal{C})$ le parent de $\mathcal{C}$.

Voisines : on appelle voisines de $\mathcal{C}$ toutes les cellules du même niveau de l'arbre que $\mathcal{C}$ ayant au moins un sommet en commun avec $\mathcal{C}$. Le nombre de voisines d'une cellule est toujours au moins un (elle-même), et au plus $3 \times 3 \times 3 = 27$. Aux niveaux 0 et 1, toutes les cellules sont voisines. Deux cellules situées à des niveaux différents ne seront pas considérées comme voisines, même si elles se touchent. On note $v(\mathcal{C})$ l'ensemble des cellules voisines de $\mathcal{C}$.

Banlieues : on appelle banlieues des cellules qui ne sont pas voisines mais dont les parents respectifs le sont. Cette définition implique que deux cellules banlieues sont toujours au même niveau de l'arbre. Aux niveaux 0 et 1, aucune cellule n'est banlieue (puisqu'elles sont toutes voisines). Le nombre de banlieues peut être zéro, mais n'excède jamais $8 \times 27 - 27 = 189$. En effet, le parent de $\mathcal{C}$ a au plus 27 voisines, qui ont chacune au plus 8 enfants, soit 8×27 « candidats », parmi lesquels il faut ôter les voisines de $\mathcal{C}$, d'où le résultat de 189 (cf. figure 1.25 sur laquelle pour simplifier on a conservé toutes les

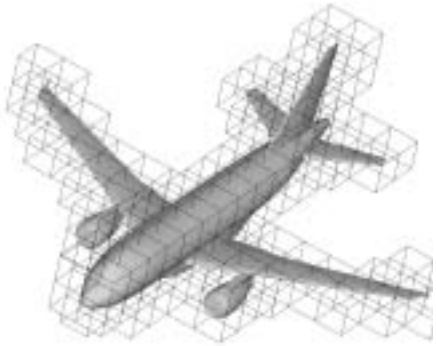
(a) Niveau 0



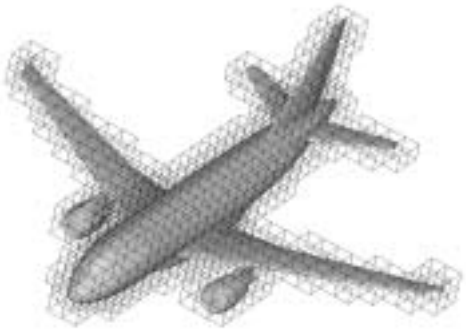
(b) Niveau 1

(c) Niveau 2

(d) Niveau 3



(e) Niveau 4



(f) Niveau 5

FIG. 1.24 – *Découpage multi-niveau d'un airbus A318*

cellules). On note $b(\mathcal{C})$ l'ensemble des cellules banlieues de $\mathcal{C}$. Soulignons que les banlieues d'une cellule $\mathcal{C}$ ne sont pas les « voisins de ses voisins ».

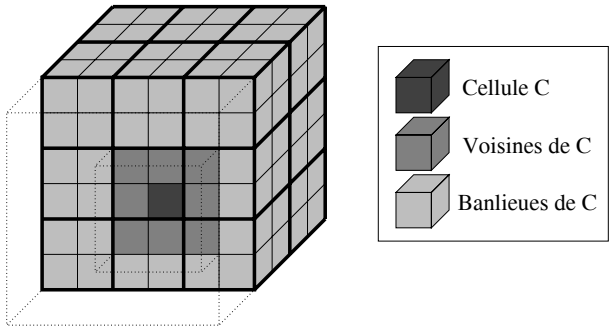


FIG. 1.25 – *Voisins et banlieues dans un octree*

Sur la figure 1.26, on a représenté (dans le cas de l'airbus A318) en rouge les portions de maillage non-voisines et/ou banlieues aux niveaux 0 à 4 par rapport à un degré de liberté de référence (pointé par la flèche). Remarquons qu'aux niveaux 0 et 1 ces ensembles sont vides, et qu'au niveau 2 ils sont égaux.

1.3.2.2 Algorithme multi-niveau complet

1.3.2.2.1 Conception On découpe notre objet à l'aide d'un octree, soit N le nombre de niveaux de cet arbre. La racine est le niveau 0, les feuilles forment le niveau $N-1$. Comparons de manière synthétique les deux algorithmes FMM déjà étudiés. D'une part, l'algorithme FMM à un niveau (le niveau $N-1$ en l'occurrence) comporte trois étapes (figure 1.27) :

1. Initialisation au niveau $N-1$.
2. Transfert au niveau $N-1$ entre cellules non-voisines.
3. Intégration au niveau $N-1$.

D'autre part, l'algorithme FMM à deux niveaux (les niveaux $N-1$ et $N-2$) comporte six étapes (figure 1.28) :

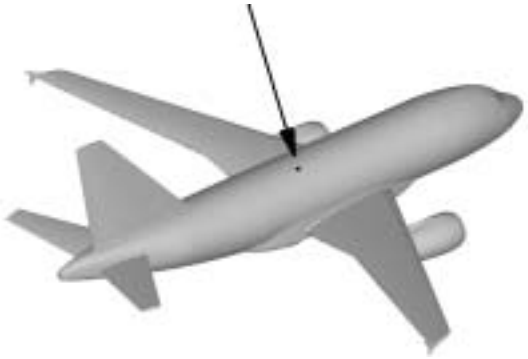
1. Initialisation au niveau $N-1$.
2. Montée vers le niveau $N-2$.
3. Transfert au niveau $N-2$ entre cellules non-voisines.
4. Descente au niveau $N-1$.
5. Transfert au niveau $N-1$ entre cellules banlieues.
6. Intégration au niveau $N-1$.

Les étapes initiale et finale sont les mêmes. Pour passer du premier l'algorithme au second, on a simplement remplacé :

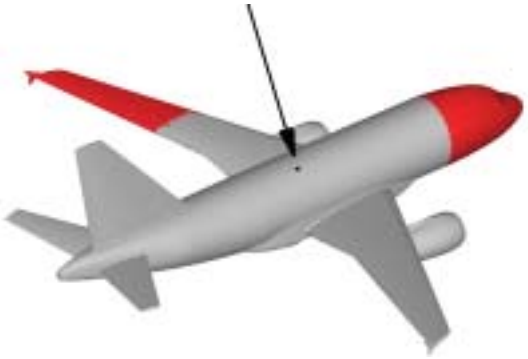
| Transfert au niveau d entre cellules non-voisines

par :

| Montée vers le niveau $d-1$
| Transfert au niveau $d-1$ entre cellules non-voisines
| Descente au niveau d
| Transfert au niveau d entre cellules banlieues



(a) Non-voisins (ou banlieues) au niveau 0 et 1

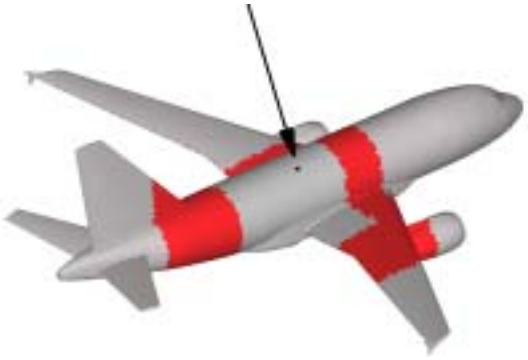


(b) Non-voisins (ou banlieues) au niveau 2

(c) Non-voisins au niveau 3

(d) Banlieues au niveau 3

(e) Non-voisins au niveau 4



(f) Banlieues au niveau 4

FIG. 1.26 – *Découpage multi-niveau d'un airbus A318*

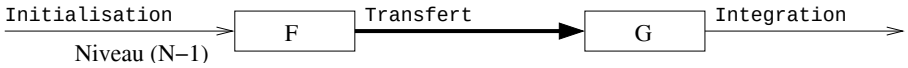


FIG. 1.27 – *Algorithme FMM à un niveau*

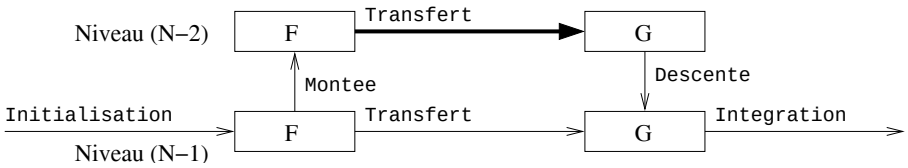


FIG. 1.28 – *Algorithme FMM à 2 niveaux*

avec $d = N - 1$ bien sûr. On peut utiliser une substitution de ce type sur l'étape 3 (avec $d = N - 2$) et obtenir ainsi un algorithme à 3 niveaux. Cette opération est en fait exactement celle qui a été détaillée lors de la description de la FMM à deux niveaux à la section 1.3.1.5 consacrée à l'algorithme continu. On retrouve en effet ici la distinction entre cellules banlieues et cellules distantes.

La méthode proposée ci-dessus ne s'applique plus lorsqu'on arrive au niveau $d = 2$. Il n'y a pas de cellules non-voisines au niveau 1, donc l'étape « Transfert au niveau 2 entre cellules non-voisines » ne pourra jamais être transformée. Une autre manière de justifier cela est de dire qu'au niveau 2, « banlieue » est synonyme de « non-voisine ». Dans notre exploration de l'arbre, on ne dépasse donc jamais le niveau 2. Cela correspond à l'algorithme multipôle « complet ». On appellera « niveau plafond » et on notera N_{plaf} le plus haut niveau exploré dans l'arbre lors du calcul multipôle. Ce niveau sera toujours compris entre 2 (algorithme complet) et $N - 1$ (algorithme mono-niveau).

1.3.2.2.2 Algorithme complet Nous allons écrire l'algorithme multipôle multi-niveau dans le cas de la formulation EFIE. On se donne un courant surfacique scalaire $t(x)$ en entrée, et un découpage récursif de Γ à l'aide d'un octree à N niveaux ($N \geq 3$). Les niveaux sont numérotés $0, 1, \ldots, N - 1$. On note N_{plaf} le niveau plafond.

Le produit $A.t$ se réalise en deux parties :

Interactions proches : elles sont traitées au niveau $N - 1$, exactement comme dans le cas mono-niveau (cf. équation (1.12)).

Interactions Lointaines : il y a cinq phases dans ce calcul.

1. **Initialisation :** pour toute cellule $\mathcal{C}^{(N-1)}$ du niveau $N - 1$, on calcule la fonction de radiation
$$\mathcal{F}_{\mathcal{C}^{(N-1)}}(\vec{s}) = \int_{x \in \Gamma \cap \mathcal{C}^{(N-1)}} e^{ik\vec{s}.xM^{(\vec{N}-1)}} t(x) dx$$
en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau $N - 1$.
2. **Montées :** pour tous les niveaux $d = N - 2, \ldots, N_{plaf}$ (dans cet ordre), et pour toute cellule $\mathcal{C}^{(d)}$ du niveau d, on calcule la fonction de radiation $\mathcal{F}_{\mathcal{C}^{(d)}}$ à partir de celles des enfants :
$$\mathcal{F}_{\mathcal{C}^{(d)}}(\vec{s}) = \sum_{\mathcal{C}^{(d+1)} \in e(\mathcal{C}^{(d)})} e^{ik\vec{s}.M^{(d+\vec{1})}M^{(d)}} \mathcal{F}_{\mathcal{C}^{(d+1)}}(\vec{s})$$

en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau d. Il y a $N - N_{plaf} - 1$ étapes de montées.

3. **Transferts :** le niveau plafond est traité à part.
   Pour le niveau $d = N_{plaf}$, et pour toute cellule $\mathcal{C}'^{(d)}$ fixée, on passe en revue les cellules $\mathcal{C}^{(d)}$ *non-voisines* de $\mathcal{C}'^{(d)}$ pour calculer :
   $$\mathcal{G}_{\mathcal{C}'^{(d)}}(\vec{s}) = \sum_{\mathcal{C}^{(d)} \notin v(\mathcal{C}'^{(d)})} T^{L^{(d)}}_{M^{(d)}\vec{M}'^{(d)}}(\vec{s}).\mathcal{F}_{\mathcal{C}^{(d)}}(\vec{s})$$
   en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau d.
   Pour tous les autres niveaux $d = N - 1, \ldots, N_{plaf} + 1$, et pour toute cellule $\mathcal{C}'^{(d)}$ fixée, on passe en revue les cellules $\mathcal{C}^{(d)}$ *banlieues* de $\mathcal{C}'^{(d)}$ pour calculer la première partie de la fonction $\mathcal{G}_{\mathcal{C}'^{(d)}}$:
   $$\mathcal{G}_{\mathcal{C}'^{(d)}}(\vec{s}) = \sum_{\mathcal{C}^{(d)} \in b(\mathcal{C}'^{(d)})} T^{L^{(d)}}_{M^{(d)}\vec{M}'^{(d)}}(\vec{s}).\mathcal{F}_{\mathcal{C}^{(d)}}(\vec{s}) \tag{1.66}$$
   en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$ pour le niveau d.
   Il y a en tout $N - N_{plaf}$ étapes de transfert.
4. **Descentes :** pour tous les niveaux $d = N_{plaf} + 1, \ldots, N - 1$ (dans cet ordre), et pour toute cellule du niveau d notée $\mathcal{C}'^{(d)}$, on calcule la deuxième partie de la fonction de radiation $\mathcal{G}_{\mathcal{C}'^{(d)}}$ à partir du parent :
   $$\mathcal{G}_{\mathcal{C}'^{(d)}}(\vec{s}) = e^{ik\vec{s}.M'^{(d-\vec{1})}M'^{(d)}}\mathcal{G}_{\mathcal{C}'^{(d-1)}}(\vec{s}) \tag{1.67}$$
   sur la quadrature de $\mathcal{S}$ associée au nombre de pôles $L^{(d)}$. Cette somme se rajoute à la partie de $\mathcal{G}_{\mathcal{C}'^{(d)}}$ calculée dans la phase « transfert ». Il y a $N - N_{plaf} - 1$ étapes de descentes.

(a) Non-voisins de $\mathcal{C}'^{(d)}$ (b) Non-voisins du parent (c) Banlieues de $\mathcal{C}'^{(d)}$

FIG. 1.29 – *Contribution du parent et des banlieues pour le calcul de* $\mathcal{G}_{\mathcal{C}'^{(d)}}$

Sur la figure 1.29, on a repéré par une flèche une inconnue située dans $\mathcal{C}'^{(d)}$. La fonction $\mathcal{G}_{\mathcal{C}'^{(d)}}$ représente l'action sur $\mathcal{C}'^{(d)}$ de toute la partie non-voisine du maillage (en rouge sur la figure 1.29a). Elle est la somme deux contributions :

- l'une provient du parent de $\mathcal{C}'^{(d)}$ via l'équation (1.67). Elle prend en compte l'influence de toute la partie du maillage non-voisine du parent (en rouge sur 1.29b).

– l'autre provient des banlieues de $\mathcal{C}'^{(d)}$ via l'équation (1.66). Elle prend en compte l'influence de toute la partie du maillage banlieue (en rouge sur 1.29c).

5. Intégration : pour toute cellule $\mathcal{C}'^{(N-1)}$, et pour toute fonction de base φ_j localisée dans $\mathcal{C}'^{(N-1)}$, on calcule l'intégrale :

$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'^{(N-1)}}\int_{\vec{s}\in\mathcal{S}}\mathcal{G}_{\mathcal{C}'^{(N-1)}}(\vec{s})e^{ik\vec{s}.M'^{(\vec{N}-1)}y}\varphi_j(y)d\vec{s}dy$$

Le résultat de cette intégration constitue la partie « interaction lointaine » de la j-ième composante du vecteur $A.t$, et se rajoute naturellement à la partie « interaction proche ».

Il y a en tout $3(N-N_{plaf})$ étapes dans ce calcul, où $(N-N_{plaf})$ est le nombre de niveaux effectivement utilisés par la FMM. On retrouve donc les 3 étapes de la méthode utilisant 1 niveau, et les 6 étapes de la méthode à 2 niveaux. Les figures 1.27 à 1.31 schématisent les différents algorithmes possibles en fonction du choix du niveau plafond.

Au niveau plafond, les transferts ont lieu entre toutes les cellules non-voisines, alors qu'aux autres niveaux, les transferts n'ont lieux qu'entre cellules banlieues. C'est la seule chose à prendre en compte lorsqu'on fait varier le niveau plafond. Cela traduit le fait qu'au niveau plafond, il faut traiter toutes les interactions n'ayant pas encore été traitées aux niveaux inférieurs. On a symbolisé cela sur les graphiques 1.27 à 1.31 en mettant une flèche plus épaisse pour les transferts au plus haut niveau exploré.

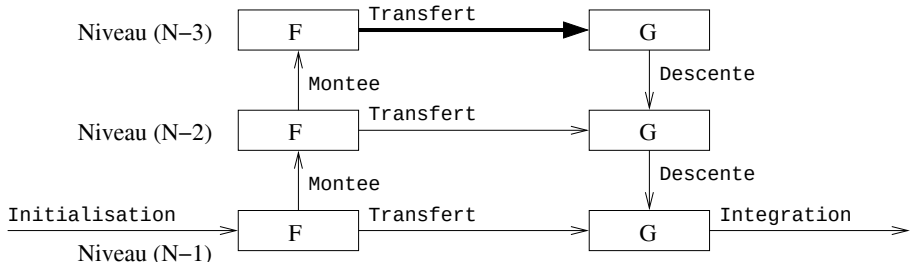


FIG. 1.30 – *Algorithme FMM à 3 niveaux ($N_{plaf}=N-3$)*

Remarquons que l'on peut, si on le souhaite, calculer les fonctions de radiation $\mathcal{F}$ et $\mathcal{G}$ au niveau 0 et 1. Mais $\mathcal{G}$ représentant l'action des cellules non-voisines, elle sera nulle, et $\mathcal{F}$ représentant l'action sur les cellules non-voisines, elle ne sera pas nulle, mais ne servira à rien dans la suite du calcul. C'est pourquoi les niveaux 0 et 1 sont en pointillé sur la figure 1.31.

1.3.2.3 Nombre d'opérations

Comme à la section 1.2.2.4, nous allons tenter d'estimer le nombre d'opérations flottantes réalisées lors d'un calcul multipôle multi-niveau.

1.3.2.3.1 Notation On reprend les notations du paragraphe 1.2.2.4.3 en ajoutant en exposant le niveau considéré chaque fois que c'est nécessaire. Concernant les données du calcul, les paramètres n_{dl}, λ, d restent les mêmes. Le seul changement concerne l'arête des cellules de la grille désormais notée $a^{(l)}$ où l est le niveau de l'octree considéré. On a la relation :

$$a^{(l-1)}=2a^{(l)}$$

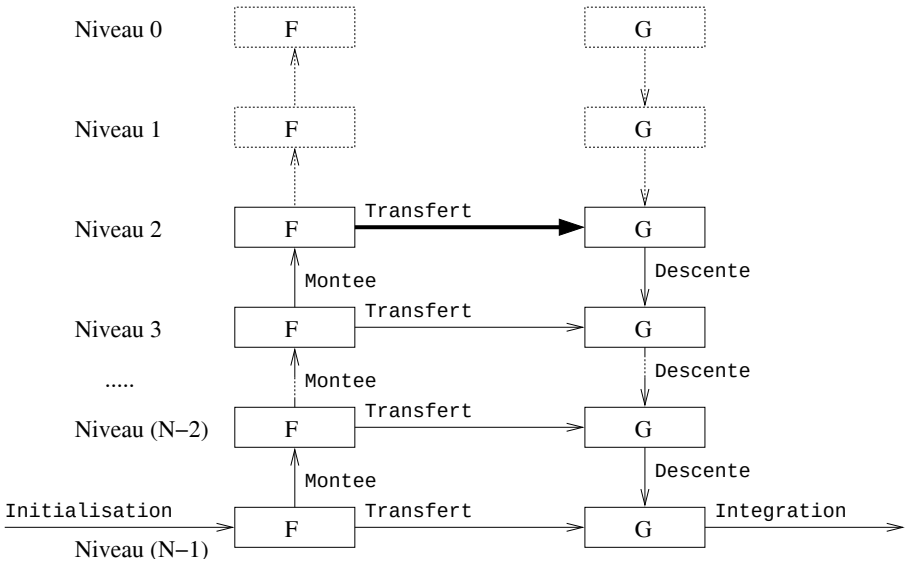


FIG. 1.31 – *Algorithme FMM multi-niveau complet ($N_{plaf} = 2$)*

On a également les relations suivantes :

- Nombre de cellules : chaque cellule du niveau $l-1$ possède en moyenne quatre enfants au niveau l, cf. figure 1.32, donc ${n_{\mathcal{C}}}^{(l-1)} = {n_{\mathcal{C}}}^{(l)}/4$
- Nombre de triangles par cellule : ${n_{\mathcal{T}}^{c}}^{(l-1)} = 4{n_{\mathcal{T}}^{c}}^{(l)}$
- Nombre de degrés de liberté par cellule : ${n_{dl}^{c}}^{(l-1)} = 4{n_{dl}^{c}}^{(l)}$
- Nombre de directions sur la sphère unité : ${n_{\vec{s}}}^{(l-1)} = 4{n_{\vec{s}}}^{(l)}$
- Nombre de pôles : $L^{(l-1)} = 2L^{(l)}$

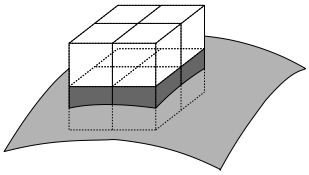


FIG. 1.32 – *Quatre enfants par parent en moyenne*

1.3.2.3.2 Coût des nouvelles étapes L'algorithme multi-niveau met en œuvre de nouveaux types d'opération. On s'intéresse ici à leur coûts respectifs. Pour les transferts entre cellules non-voisines, il est donné par la formule (1.44) que l'on rappelle ci-dessous. Regardons le coût des montées, descentes, et transferts entre banlieues dans le cas d'une formulation à 2 termes.

Transfert entre cellules non-voisines :

$$\begin{aligned} f_{trans1}^{(l)} &= 4n_{\mathcal{C}}^{(l)}(n_{\mathcal{C}}^{(l)} - 9)n_{\vec{s}}^{(l)} \\ &= n_{dl}\frac{96\pi^2}{\sqrt{3}d^2}\left[\frac{n_{dl}}{\sqrt{3}d^2(a^{(l)})^2} - 9\right] \end{aligned}$$

Montée : pour chacune des $2n_{\mathcal{C}}^{(l)}$ fonctions de radiation, on réalise une translation ($n_{\vec{s}}^{(l)} \times$, 0+), une sommation (0 $\times$, $n_{\vec{s}}^{(l)}$+). On réalise ensuite $2n_{\mathcal{C}}^{(l-1)}$ multiplications par $\sin\theta$ ($n_{\vec{s}}^{(l)} \times$, 0+) puis autant d'extrapolations. Le coût de chaque extrapolation est $4L^{(l-1)}(L^{(l)})^2$. Enfin, on divise par $\sin\theta$ les fonctions obtenues ($n_{\vec{s}}^{(l-1)} \times$, 0+).
Total :

$$\begin{aligned} f_{montee}^{(l)} &= 2n_{\mathcal{C}}^{(l)}(2n_{\vec{s}}^{(l)}) + 2n_{\mathcal{C}}^{(l-1)}(n_{\vec{s}}^{(l)} + 4L^{(l-1)}(L^{(l)})^2 + n_{\vec{s}}^{(l-1)}) \\ &= \pi^2 \frac{n_{dl}}{d^2}\left[52\sqrt{3} + 96\pi a^{(l)}\right] \end{aligned} \tag{1.68}$$

Transfert entre banlieues : chaque cellule est transférée sur toutes les cellules banlieues soit $27n_{\mathcal{C}}^{(l)}$ transferts. Chaque transfert opère sur deux fonctions de radiation ($(\mathcal{F}_{\mathcal{C}})_\theta$ et $(\mathcal{F}_{\mathcal{C}})_\phi$) à $n_{\vec{s}}^{(l)}$ points chacune, avec (1 $\times$, 1+) à chaque fois, soit un total de $4n_{\vec{s}}^{(l)}$ opérations par transferts.
Total :

$$\begin{aligned} f_{trans2}^{(l)} &= 108 n_{\mathcal{C}}^{(l)} n_{\vec{s}}^{(l)} \\ &= 864\sqrt{3}\pi^2 \frac{n_{dl}}{d^2} \end{aligned} \tag{1.69}$$

Un point à souligner est que ce coût est indépendant du niveau.

Descente : pour chacune des $2n_{\mathcal{C}}^{(l-1)}$ fonctions de radiation, on réalise une division par $\sin\theta$ ($n_{\vec{s}}^{(l-1)} \times$, 0+), une réduction dont le coût est $4L^{(l-1)}(L^{(l)})^2$, et une multiplication par $\sin\theta$ ($n_{\vec{s}}^{(l)} \times$, 0+).
Puis, pour chacune des $2n_{\mathcal{C}}^{(l)}$ fonctions de radiation du niveau l, on réalise une translation et une sommation, soit ($2n_{\vec{s}}^{(l)} \times$, 0+).
Total :

$$\begin{aligned} f_{descente}^{(l)} &= 2n_{\mathcal{C}}^{(l-1)}(2n_{\vec{s}}^{(l-1)}) + 2n_{\mathcal{C}}^{(l-1)}(n_{\vec{s}}^{(l-1)} + 4L^{(l-1)}(L^{(l)})^2 + n_{\vec{s}}^{(l)}) \\ &= \pi^2 \frac{n_{dl}}{d^2}\left[52\sqrt{3} + 96\pi a^{(l)}\right] \end{aligned} \tag{1.70}$$

On retrouve exactement le même coût que pour la montée.

1.3.2.3.3 Ajout d'un niveau Pour un maillage donné (n_{dl} et d fixés), on suppose avoir écrit une méthode multipôle dont le niveau plafond est l, on note toujours $a^{(l)}$ l'arête des boîtes à ce niveau. La question à laquelle on souhaite répondre est : est-il rentable de rajouter un niveau à notre FMM ? Le rajout d'un niveau au calcul multipôle revient à effectuer la substitution décrite page 65. Nous allons comptabiliser le gain obtenu à travers cette modification de l'algorithme. $a^{(l)}$ sera notre paramètre.

Le nombre d'opérations supplémentaires s'écrit :

$$\begin{aligned} \Delta(a^{(l)}) &= \left[f_{montee}^{(l)} + f_{trans1}^{(l-1)} + f_{descente}^{(l)} + f_{trans2}^{(l)}\right] - f_{trans1}^{(l)} \\ &= \pi^2 \frac{n_{dl}}{d^2}\left[968\sqrt{3} + 192\pi a^{(l)} - 24\frac{n_{dl}}{d^2(a^{(l)})^2}\right] \end{aligned}$$

$\Delta(a^{(l)})$ représente le nombre d'opérations que l'on rajoute au calcul lorsqu'on explore un niveau supplémentaire de l'octree. Si $\Delta(a^{(l)}) > 0$, alors le calcul modifié est plus long que

l'original. Si $\Delta(a^{(l)}) < 0$, alors le rajout d'un niveau au calcul est bénéfique. Etudions donc les variations de $\Delta(a^{(l)})$. Pour $a^{(l)} > 0$, la dérivée de cette fonction est toujours positive :

$$\frac{\partial \Delta(a^{(l)})}{\partial a^{(l)}} = \pi^2 \frac{n_{dl}}{d^2} \left[192\pi + 48 \frac{n_{dl}}{d^2 (a^{(l)})^3} \right]$$

Donc $\Delta(a^{(l)})$ s'annule en un seul point. En utilisant Maple, avec d=10, on trouve les solutions suivantes pour différentes valeurs de n_{dl} (tableau 1.2). Les valeurs indiquées sont des tailles de boîtes, exprimées en nombre de longueurs d'onde.

n_{dl}	Racine de Δ
10^3	0,356
10^4	1,023
10^5	2,696
10^6	6,536

TAB. 1.2 – *Racine de* $\Delta(a^{(l)})$

Si la taille de boîtes au plus haut niveau $a^{(l)}$ est plus petite que la racine, $\Delta(a^{(l)})$ est négatif, le rajout d'un niveau dans l'algorithme permettra d'accélérer le calcul. Au contraire, si $a^{(l)}$ est plus grand que la racine, ce rajout est déconseillé. Dans la mesure où l'on souhaite réaliser des transferts au niveau $l-1$, on a forcément $l-1 \geq 2$ i.e. $l \geq 3$.

Pour exploiter l'étude qui précède, il faut connaître $(a^{(0)}\lambda)$, c'est à dire la taille (en mètre) de la boîte cubique englobant tout Ω. On connaît la surface du maillage : $|\Gamma| = \frac{\lambda^2 n_{dl}}{2\sqrt{3}d^2}$. En général, $|\Gamma|$ est proportionnel à $(a^{(0)}\lambda)^2$ (i.e. la surface de l'objet croit comme le carré de son diamètre). Le coefficient de proportionnalité dépend de la forme de l'objet. Pour une sphère, il vaut π. Dans le cas d'un objet élancé comme un avion, il est plus proche de 1/2. Par exemple, pour l'airbus A318 représenté figure 1.33, la surface du maillage est de 713 m^2, alors que la boîte englobante a une arête de 34,1 m (déterminée par l'envergure de l'appareil). Le coefficient de proportionnalité est donc de 0,61.

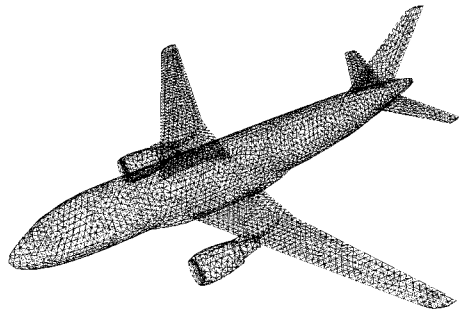


FIG. 1.33 – *Airbus A318*

Plaçons nous dans le cas : $|\Gamma| = 2.(a^{(0)}\lambda)^2$ qui correspond à un objet plutôt compact (avion de chasse, nacelle de réacteur, voiture). On a alors :

$$\begin{cases} a^{(0)} = \dfrac{\sqrt{|\Gamma|/2}}{\lambda} = \dfrac{1}{d}\sqrt{\dfrac{n_{dl}}{4\sqrt{3}}}, \\ a^{(3)} = \dfrac{a^{(0)}}{8} \end{cases} \tag{1.71}$$

n_{dl}	$a^{(0)}$	$a^{(3)}$
10^3	1,2	0,15
10^4	3,8	0,47
10^5	12,0	1,5
10^6	38,0	4,7

TAB. 1.3 – *Taille des boîtes aux niveaux 0 et 3*

Le tableau 1.3 donne les valeurs de $a^{(0)}$ et de $a^{(3)}$ pour les mêmes valeurs de n_{dl} que dans le tableau 1.2. A chaque fois, on constate que $a^{(3)}$ est inférieur à la racine de Δ. Il en serait de même pour $a^{(4)}$, $a^{(5)}$ et tous les suivants. Ce qui signifie que quel que soit le niveau $l \geq 3$ considéré $\Delta(a^{(l)}) < 0$, on gagne du temps à faire passer l'algorithme au niveau $l-1$.

Par conséquent, l'algorithme multipôle le plus rapide est l'algorithme complet, qui explore tous les niveaux depuis les feuilles jusqu'au niveau 2. Bien sûr, c'est un résultat obtenu après de nombreuses simplifications. Nous verrons aux chapitres suivants ce qu'il en est en pratique (cf. section 3.4.7 dans le cas séquentiel, et 5.3.1 dans le cas parallèle).

1.3.2.3.4 Nombre total d'opérations On note a_f la taille des feuilles, N le nombre de niveaux de l'octree, et n_{dl} le nombre de degrés de liberté du calcul. On s'intéresse au coût total de la méthode multipôle en fonction de ces trois paramètres.

Les différentes étapes du calcul ont les coûts suivants :

Interactions proches : d'après la formule (1.46), on a :

$$f_{proche} = n_{dl} 18\sqrt{3} d^2 a_f^2$$

Initialisations : on reprend la formule (1.43) :

$$f_{init} = n_{dl}\left[288\pi^2 a_f^2 + \frac{240\pi^2}{\sqrt{3}d^2}\right]$$

Intégrations : le coût des intégrations nous est donné par l'équation (1.45) :

$$f_{integ} = n_{dl} 24\pi^2 \left[\frac{6\sqrt{3}}{d^2} + 66 a_f^2\right]$$

Transferts : d'après l'équation (1.69), ils ont un coût indépendant du niveau. On le multiplie par le nombre de niveaux où des transferts ont lieu : $N-2$, et l'on obtient le coût total des transferts :

$$f_{transfert} = 864\sqrt{3}\pi^2 \frac{n_{dl}}{d^2}(N-2)$$

Montées et descentes : d'après (1.68) et (1.70), les opérations de descente et de montée ont le même coût : $\pi^2 n_{dl}/d^2 \left[52\sqrt{3} + 96\pi a^{(l)}\right]$. Dans cette formule, l désigne le niveau inférieur. Donc l varie de $N-1$ à 3, et conjointement $a^{(l)}$ varie de a_f à $2^{N-4}a_f$. Le coût total s'écrit :

$$\begin{aligned} f_{mont/desc} &= 2\sum_{l=3}^{N-1} \pi^2 \frac{n_{dl}}{d^2}\left[52\sqrt{3} + 96\pi a^{(l)}\right] \\ &= 2\pi^2 \frac{n_{dl}}{d^2}\left[52\sqrt{3}(N-3) + 96\pi a_f(1+2+4+\cdots+2^{N-4})\right] \\ &= 2\pi^2 \frac{n_{dl}}{d^2}\left[52\sqrt{3}(N-3) + 96\pi a_f(2^{N-3}-1)\right] \end{aligned} \tag{1.72}$$

Pour exploiter tout cela, il faut rappeler que a_f et N sont liés par la condition :

$$2^{N-1}a_f = a^{(0)}$$

Reprenons la valeur de $a^{(0)}$ donnée par (1.71), on obtient la relation :

$$2^{N-1}a_f = \frac{1}{d\sqrt{4\sqrt{3}}}\sqrt{n_{dl}}$$

On détermine la valeur de N :

$$\begin{aligned} N &= 1 + \log_2\left(\frac{1}{d\sqrt{4\sqrt{3}}}\frac{\sqrt{n_{dl}}}{a_f}\right) \\ &= 1 - \log_2(d\sqrt{4\sqrt{3}}) + \frac{1}{2}\log_2(n_{dl}) - \log_2(a_f) \end{aligned} \tag{1.73}$$

On en déduit le nombre total d'opérations :

$$\begin{aligned} f_{total}(n_{dl},a_f,d) = n_{dl}\Bigg[& 18\sqrt{3}d^2a_f^2 - \frac{848\sqrt{3}\pi^2}{d^2} + 1872\pi^2 a_f^2 \\ & + 968\sqrt{3}\pi^2\frac{1}{d^2}\left(-\log_2(d\sqrt{4\sqrt{3}}) + \frac{1}{2}\log_2(n_{dl}) - \log_2(a_f)\right) \\ & + \frac{48\pi^3}{d^3\sqrt{4\sqrt{3}}}\sqrt{n_{dl}} - \frac{\pi^2}{d^2}192\pi a_f\Bigg] \end{aligned}$$

La formule donnant le nombre total d'opérations $f_{total}(n_{dl},a_f,d)$ est bien sûr très théorique, et l'on verra plus loin ce qu'il en est en pratique. Néanmoins nous allons utiliser cette expression pour en déduire des résultats intéressants concernant la taille de feuille optimale, et la complexité asymptotique de la méthode.

1.3.2.3.5 Choix de la taille des feuilles L'objectif est de déterminer la valeur optimale de a_f pour un maillage donné. La dépendance de f_{total} par rapport à a_f fait apparaître des termes en a_f, en a_f^2, en $\log a_f$ et des termes constants. On note pour simplifier :

$$\begin{cases} f_{total}(a_f) = \alpha_1 a_f^2 + \alpha_2 a_f + \alpha_3 \log a_f + \alpha_4, \\ \alpha_1 > 0, \\ \alpha_2 < 0, \\ \alpha_3 < 0 \end{cases}$$

Pour étudier les variations de $f_{total}(a_f)$, on calcule sa dérivée :

$$f'_{total}(a_f) = 2\alpha_1 a_f + \alpha_2 + \alpha_3 \frac{1}{a_f}$$

Elle croît de $-\infty$ à $+\infty$ pour a_f variant de 0 à $+\infty$. Elle ne s'annule que pour

$$a_f = \frac{-\alpha_2 + \sqrt{\alpha_2^2 - 8\alpha_1\alpha_3}}{4\alpha_1} \tag{1.74}$$

La fonction f_{total} est donc décroissante jusqu'à ce point, où elle atteint son minimum avant de recroître. Donnons les valeurs des coefficients α_i :

$$\begin{cases} \alpha_1 = n_{dl}(18\sqrt{3}d^2 + 1872\pi^2), \\ \alpha_2 = n_{dl}(-\frac{\pi^2}{d^2}192\pi), \\ \alpha_3 = n_{dl}(-968\sqrt{3}\pi^2\frac{1}{d^2}), \\ \alpha_4 = \frac{n_{dl}\pi^2}{d^2}\left(-848\sqrt{3} + 968\sqrt{3}(-\log_2(d\sqrt{4\sqrt{3}}) + \frac{1}{2}\log_2(n_{dl})) + \frac{24\pi\sqrt{n_{dl}}}{d3^{1/4}}\right) \end{cases}$$

Les termes en $\log n_{dl}$ et $\sqrt{n}_{dl}$ sont tous dans α_4, les trois autres coefficients dépendent linéairement de n_{dl}. Dès lors, il est clair que la valeur optimale de a_f donnée par (1.74) ne dépend pas du nombre de degrés de liberté.

Numériquement, avec $d = 10$, on trouve la valeur optimale $a_f = 0{,}075$. Mais comme dans le cas mono-niveau, notre étude « favorise » les petites boîtes en y sous-estimant le nombre de pôles. Pour bien faire, une telle étude devra être réalisée directement par le logiciel de FMM. En pratique, une valeur de a_f comprise entre 0,25 et 0,5 donne de bons résultats.

1.3.2.3.6 Complexité de l'algorithme Pour simplifier les expressions, on utilise la valeur numérique $d = 10$. Avec un logiciel comme Maple, on obtient l'expression suivante pour le nombre total d'opérations :

$$\begin{aligned} f_{total} = & \left[2{,}16.10^4 a^2 - 59{,}5a - 9{,}26.10^2 - 2{,}39.10^2 \ln(a)\right] n_{dl} \\ & + 1{,}19.10^2 n_{dl} \ln(n_{dl}) \\ & + 0{,}56 {n_{dl}}^{3/2} \end{aligned}$$

Pour des valeurs de n_{dl} petite, le terme prépondérant est celui en $n_{dl}\ln(n_{dl})$. Lorsque n_{dl} devient grand, c'est celui en ${n_{dl}}^{3/2}$. La transition se fait lorsque $119 n_{dl} \ln n_{dl} = 0{,}56 n_{dl}^{3/2}$, soit $n_{dl} = 1{,}18.10^7$. Ce seuil ne dépend pas de la taille des feuilles a_f.

Dix millions d'inconnues pour un calcul de ce type est une valeur considérable. Par conséquent, on a un algorithme dont la complexité asymptotique est $n_{dl}^{3/2}$, mais pour des valeurs usuelles (au jour d'aujourd'hui) le nombre d'opérations est proportionnel à $n_{dl} \ln n_{dl}$. Si on remonte les calculs pour voir la provenance de cet exposant 3/2, on constate qu'il provient des opérations d'extrapolation et de réduction. Même si elles ne prennent aujourd'hui que peu de temps, il faut donc garder à l'esprit que pour de très grandes valeurs de n_{dl}, elles constituent l'étape critique (on a d'ailleurs évoqué la possibilité d'y inclure une FMM 1D pour diminuer la complexité).

1.3.2.3.7 Complexité optimale Pour en terminer avec l'étude du nombre d'opérations, nous allons regarder quelle serait la complexité de la méthode si on y implémentait une extrapolation/réduction utilisant une FMM mono-dimensionnelle.

L'extrapolation (et la réduction) ont dans leur implémentation actuelle une complexité en $(L^{(l)})^3$ liée à la présence du produit matrice-vecteur (1.59) modifié sous la forme (1.61). Si on réalise le produit par C via une méthode multipôle, sa complexité se ramène à $(L^{(l)})^2 \log(L^{(l)})$. Dans ce cas, le terme en $a^{(l)}$ dans les équations (1.68) et (1.70) donnant le coût individuel d'une montée et descente devient $\log(a^{(l)})$. Dans l'équation (1.72) donnant le coût cumulé des montées/descentes, le terme :

$$a_f(1 + 2 + 4 + \cdots + 2^{N-4})$$

devient :

$$\begin{aligned}
&\log(a_f) + \log(2a_f) + \ldots + \log(2^{N-4}a_f) \\
&= (N-3)\log(a_f) + (1 + 2 + \ldots + (N-4))\log 2 \\
&= (N-3)\log(a_f) + (N-3)(N-4)/2 \log 2
\end{aligned}$$

Avec a_f fixé et N donné par l'équation (1.73), la somme ci-dessus a une complexité asymptotique en N^2 i.e. $\log^2(n_{dl})$. Dans le coût total, le terme en $(n_{dl})^{3/2}$ est remplacé par un terme en $n_{dl}\log^2(n_{dl})$. La complexité de la méthode multipôle multi-niveau complète est donc asymptotiquement $n_{dl}\log^2(n_{dl})$. Bien sûr, cette complexité n'est atteinte que pour de très grandes valeurs de n_{dl} supérieures à dix millions.

1.3.2.3.8 Stockage Les deux principales données à stocker sont la matrice des interactions proches et les fonctions de radiation. Avec a_f fixé, la taille de la matrice des interactions proches s'écrit :

$$\begin{aligned}
T_{proche} &= n_{\mathcal{C}}^{(N-1)}(n_{dl}^c)^2 \\
&= n_{dl}\sqrt{3}d^2{a_f}^2
\end{aligned}$$

Elle est donc proportionnelle à n_{dl}.

Par ailleurs, on a ${n_{\mathcal{C}}}^{(l-1)}{n_{\vec{s}}}^{(l-1)} = {n_{\mathcal{C}}}^{(l)}{n_{\vec{s}}}^{(l)}$, ce qui signifie que la taille cumulée sur un niveau des fonctions de radiation est la même aux niveau l et $l-1$. La taille cumulée sur tout l'arbre, pour les fonctions $\mathcal{F}$ et $\mathcal{G}$, vaut donc :

$$\begin{aligned}
T_{radiation} &= 2n_{\mathcal{C}}^{(N-1)}{n_{\vec{s}}}^{(N-1)}(N-2) \\
&= 48\pi^2\frac{n_{dl}}{\sqrt{3}d^2}(\log_2(d\sqrt{4\sqrt{3}}) + \frac{1}{2}\log_2(n_{dl}) - \log_2(a_f) - 1)
\end{aligned}$$

Elle est donc proportionnelle à $n_{dl}\log n_{dl}$.

L'espace de stockage nécessaire à la méthode multipôle multi-niveau est donc globalement en $n_{dl}\log n_{dl}$.

Conclusion

On a présenté une méthode permettant d'accélérer les produits matrice-vecteur issus de la résolution itérative des équations de Maxwell par formulations intégrales. Là où une méthode

classique requiert un temps d'exécution et une taille de stockage en ${n_{dl}}^2$, la méthode multipôle à un niveau ramène ces quantités à ${n_{dl}}^{3/2}$, et la méthode multipôle multi-niveau complète à $n_{dl} \log n_{dl}$. En choisissant correctement le nombre de pôles, on peut obtenir un écart relatif de l'ordre de 10^{-3} entre produit multipôle et produit classique.

La méthode multipôle est d'une grande complexité mathématique et algorithmique, nous avons tenté d'en faire une présentation théorique claire et progressive. La plupart des développements présentés ici ne sont pas nouveaux sur le fond, mais sur la forme nous avons essayé d'en faire un exposé aussi didactique que possible.

Le comptage exhaustif des opérations induites par un produit multipôle a conduit à des résultats intéressant concernant l'optimalité de la FMM complète et la taille de feuille optimale. Il faut souligner que cette méthode, relativement récente, fait encore l'objet de nombreuses recherches.

Chapitre 2

Implémentation et Optimisations

Sommaire

Introduction **79**
2.1 Utilisation d'une liste de tâches **80**
2.2 Traitement des opérations de transfert **86**
2.3 Traitement des opérations de montée/descente **100**
2.4 Traitement des autres opérations **107**
2.5 Optimisation de la mémoire **111**
Conclusion **120**

Introduction

La méthode multipôle rapide, dans sa variante adaptée à la formulation intégrale des équations de Maxwell, met en œuvre une grande richesse mathématique et algorithmique. La complexité mathématique provient notamment de l'utilisation de fonctions spéciales, de polynômes de Legendre, d'harmoniques sphériques, de quadratures de la sphère unité et de transformées de Fourier. La complexité algorithmique apparaît à travers l'exploration d'un arbre, et les notions de parent, d'enfants, de voisines, et de banlieues qui lui sont associées. Bien sûr, les développements liés à la méthode multipôle se superposent à ceux issus de la méthode classique de résolution (formulation intégrale, discrétisation, maillage, solveur itératif, préconditionneur éventuel). Mais parallèlement à cette grande richesse, la FMM se doit d'être optimisée. En effet, sa principale utilité est de permettre de résoudre des problèmes de grande taille qui étaient auparavant inaccessibles pour des raisons d'occupation mémoire ou de temps d'exécution. On a donc affaire à des utilisateurs qui vont vouloir pousser la méthode « dans ses retranchements » sur ces deux points. Enfin, étant donné que la méthode multipôle réalise un produit matrice-vecteur qui se substitue au produit classique, on souhaite faire le minimum de modifications au solveur itératif utilisé. L'algorithme multipôle doit donc être une « boîte noire » vis-à-vis des autres parties du logiciel.

Au moment de réaliser l'implémentation de cette méthode, on est donc soumis à trois contraintes (complexité, optimisation, boîte noire) qui non seulement sont contradictoires, mais qui en plus risquent de conduire à la rédaction d'un code excessivement compliqué (plus communément appelé « usine à gaz »). On se propose dans la suite d'exposer les techniques

mises en œuvre pour réaliser conjointement l'implémentation et l'optimisation de la méthode multipôle.

2.1 Utilisation d'une liste de tâches

2.1.1 Principe de la méthode

2.1.1.1 Objectifs

La méthode multipôle est constituée d'un ensemble de tâches élémentaires : initialisation, montée, transfert, descente, intégration. A ces opérations déjà connues se rajouteront par la suite d'autres types de tâches permettant de fonctionner en parallèle ou en mode out-of-core. Une procédure déterminant et réalisant directement au moment de l'exécution l'ensemble des tâches à effectuer pour faire un produit multipôle optimisé, parallèle, out-of-core serait excessivement lourde à rédiger et à maintenir. De plus, une telle approche serait lente, car elle conduirait à « réfléchir » l'algorithme à chaque produit matrice-vecteur.

On a choisi une approche qui sépare la création de l'algorithme et son exécution. On procède donc de la manière suivante : l'ensemble du calcul à venir est généré sous la forme d'une liste de tâches, conservée sur disque. Cette liste contient d'abord de quoi faire une FMM basique, puis elle est enrichie en fonctionnalités (parallélisme, out-of-core), puis optimisée (réordonnancement des transferts, des montées/descentes, des communications), et enfin sauvegardée. Le calcul FMM proprement dit se contente de relire et d'exécuter cette liste.

Cette approche permet au développeur d'introduire les améliorations (gestion out-of-core, etc.) une par une, sous forme de modules séparés se contentant de réordonner et/ou d'enrichir la liste. En outre, ce mode de fonctionnement réduit au strict minimum la réalisation effective du produit multipôle car toute la richesse algorithmique est contenue dans la liste. On gagne donc en temps d'exécution, en temps de développement, et en clarté du code source.

2.1.1.2 Paramètres de contrôles : contributions et utilisations

Les tâches élémentaires évoquées ci-dessus sont dépendantes les unes des autres. On peut intervertir leur ordre d'exécution dans une certaine mesure, mais il existe des contraintes. Par exemple, on ne peut pas transférer une fonction de radiation $\mathcal{F}$ tant que celle-ci n'a pas été calculée soit par une initialisation, soit par une montée de ses enfants. Par ailleurs, on souhaiterait ne garder les données en mémoire que le temps de leur utilisation. Ainsi, dès lors qu'une fonction de radiation $\mathcal{F}$ a subi tous les transferts et toutes les montées prévues, on peut l'éliminer. On va proposer un mécanisme permettant de savoir à chaque instant si une fonction de radiation est « valide » ou « à éliminer ». On associe à chaque fonction de radiation $\mathcal{F}$ ou $\mathcal{G}$ deux compteurs : un nombre de contributions, et un nombre d'utilisations.

Le nombre de contributions de $\mathcal{F}$ noté $C_{\mathcal{F}}^{\Downarrow}$ correspond au nombre de tâches élémentaires qui vont contribuer à calculer la fonction en question (d'où le sens de la flèche dans l'écriture $C_{\mathcal{F}}^{\Downarrow}$, qui va vers $\mathcal{F}$). Par exemple, si la cellule associée à $\mathcal{F}$ a n enfants, alors $C_{\mathcal{F}}^{\Downarrow} = n$. Si cette cellule est une feuille, $C_{\mathcal{F}}^{\Downarrow} = 1$ (pour l'initialisation). Pour les fonctions de radiation $\mathcal{G}$, $C_{\mathcal{G}}^{\Downarrow}$ est égal au nombre de fonctions $\mathcal{F}$ à transférer vers $\mathcal{G}$, plus éventuellement 1 si on descend la fonction $\mathcal{G}$ du parent.

Le nombre d'utilisations de $\mathcal{F}$ noté $C_{\mathcal{F}}^{\Uparrow}$ désigne naturellement le nombre de tâches élémentaires qui vont utiliser cette fonction. $C_{\mathcal{F}}^{\Uparrow}$ est égal au nombre de transferts de $\mathcal{F}$ à

effectuer, plus éventuellement 1 si on remonte la fonction $\mathcal{F}$ vers son parent. Pour une fonction $\mathcal{G}$, le nombre d'utilisations $C_{\mathcal{G}}^{\Uparrow}$ est égal au nombre d'enfants, ou à un dans le cas d'une feuille (pour l'intégration).

L'intérêt du compteur de contributions est de surveiller la validité du calcul. En particulier, lors des modifications de la liste de tâches pour y rajouter ou y réordonner les opérations, ce compteur permet de s'assurer que les tâches s'exécutent dans un ordre cohérent. Le compteur d'utilisation permet pour sa part de savoir si une fonction de radiation va encore servir à l'avenir ou si elle peut être tout de suite libérée. Indirectement, il permet de vérifier qu'aucune tâche n'a été malencontreusement effacée lors des modifications de la liste de tâches.

Dans la pratique, ces compteurs existent sous deux formes : une valeur « courante » et une valeur « maximale ». Les valeurs maximales des compteurs de contribution et d'utilisation de toutes les fonctions de radiation sont fixés au moment de la création de la liste de tâches. Les valeurs courantes sont mises à zéro au début de chaque calcul multipôle, et sont incrémentées au fur et à mesure de l'exécution des opérations. Lorsque le nombre de contributions « courant » d'une fonction de radiation atteint la valeur « maximale », cela signifie que le calcul de cette fonction est fini, que son contenu est « valide » et peut être utilisé. Lorsque le nombre d'utilisations « courant » d'une fonction de radiation atteint la valeur « maximale », cela signifie que cette fonction ne servira plus dans la suite du calcul, et qu'elle peut être libérée.

Ces compteurs sont avant tout des paramètres de contrôle. Si le programme marche, ils ne servent presque pas. Mais étant donné la complexité de l'ensemble, ils nous permettent de contrôler la validité du calcul dans deux cas principalement :

- Si on essaie d'utiliser une fonction non valide ;
- Si on termine le calcul avec une fonction pour laquelle le nombre d'utilisations n'a pas atteint son maximum.

Dans les deux cas, cela signifie qu'on a commis une erreur dans la construction de l'algorithme. Cette méthode fournit une aide très précieuse pour la mise au point d'un code FMM.

Terminons avec une remarque : grâce aux compteurs de contribution, on peut savoir au moment de l'exécution quelles sont les fonctions valides, et donc on peut déterminer quelles sont les tâches de notre liste que l'on peut accomplir, et celles pour lesquelles il nous manque encore des données. Dès lors, on pourrait envisager un mode de fonctionnement « désordonné » (ou asynchrone), dans lequel la liste de tâches deviendrait un ensemble de tâches « à accomplir ». Le produit multipôle consisterait à extraire de cet ensemble des tâches réalisables et à les exécuter. Si tout se passe bien, on finit par obtenir un ensemble vide. Cette méthode est plus simple à programmer, puisqu'on n'a plus à se soucier des problèmes d'ordonnancement des opérations. En pratique, les performances sont assez mauvaises, puisque la machine passe son temps à chercher une tâche à accomplir (un cas à un million de degrés de liberté peut facilement comporter plusieurs millions de tâches élémentaires). On a donc privilégié un fonctionnement déterministe, sous forme de liste, certes moins souple mais plus sûr et plus rapide.

2.1.1.3 Sous-listes

La méthode multipôle dans son ensemble est constituée d'une succession d'étapes, symbolisées par des flèches sur la figure 1.31. Chacune de ces étapes comporte un seul type de tâches élémentaires (que des initialisations, ou que des montées, ...). De la même manière, on divise

la liste de tâches principale en sous-listes associées à ces différentes étapes. Les modifications de la liste de tâches se feront alors sous-liste par sous-liste, ce qui simplifiera leur réalisation.

2.1.2 Tâches élémentaires

Le but de cette partie est de présenter le contenu et les paramètres associés à chaque type de tâche élémentaire. Le traitement effectif de ces opérations sera abordé plus loin dans le chapitre.

2.1.2.1 Initialisation($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$.
Action :	Calcule la fonction de radiation $\mathcal{F}_{\mathcal{C}}$.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{F}_{\mathcal{C}}}$.
Utilisation :	Néant.

2.1.2.2 Montée($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$.
Action :	Remonte la fonction $\mathcal{F}_{\mathcal{C}}$ de la cellule $\mathcal{C}$ vers son parent $p(\mathcal{C})$.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{F}_{p(\mathcal{C})}}$.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{F}_{\mathcal{C}}}$.

2.1.2.3 Descente($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$.
Action :	Descend la fonction $\mathcal{G}_{p(\mathcal{C})}$ de $p(\mathcal{C})$ vers $\mathcal{C}$.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{G}_{\mathcal{C}}}$.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{G}_{p(\mathcal{C})}}$.

Dans le cas d'une montée, on donne en paramètre la cellule source (la montée se fait de $\mathcal{C}$ vers $p(\mathcal{C})$). Ici, dans le cas d'une descente, on donne en paramètre la cellule destination (la descente se fait de $p(\mathcal{C})$ vers $\mathcal{C}$). La raison de cette différence est simple : si on donnait la cellule parent en argument, il faudrait en plus préciser l'enfant concerné par la descente en second argument. Il est donc plus simple de ne préciser que l'enfant concerné par la montée ou la descente.

2.1.2.4 Transfert($\mathcal{C}$,$\mathcal{C}'$)

Paramètres :	Deux cellules $\mathcal{C}$ et $\mathcal{C}'$.
Action :	Transfère la fonction $\mathcal{F}_{\mathcal{C}}$ vers $\mathcal{G}_{\mathcal{C}'}$.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{G}_{\mathcal{C}'}}$.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{F}_{\mathcal{C}}}$.

2.1.2.5 Intégration($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$.
Action :	Intègre la fonction de radiation $\mathcal{G}_{\mathcal{C}}$.
Contribution :	Néant.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{G}_{\mathcal{C}}}$.

2.1.3 Création de la liste

On va transcrire l'algorithme multipôle multi-niveau de la section 1.3.2.2.2 sous la forme d'une liste de tâches divisée en sous-listes numérotées de 1 à $3(N - N_{plaf})$ (où N est le nombre de niveaux de l'arbre, et N_{plaf} est le niveau plafond i.e. le plus haut niveau exploré par la FMM). On note $\mathcal{L}_i$ la i-ème sous-liste. On peut voir l'ordre de numérotation des étapes dans le cas d'une FMM à 4 niveaux sur la figure 2.1. Le choix de cette numérotation sera justifié à la section 2.5.

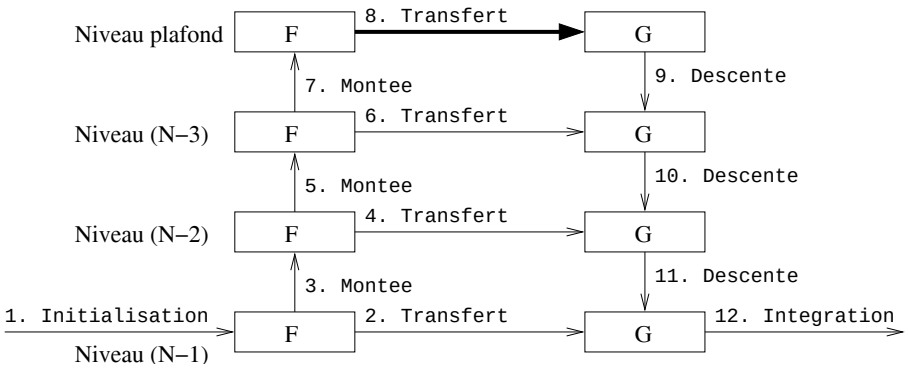


FIG. 2.1 – *Algorithme FMM à 4 niveaux*

2.1.3.1 Sous-liste des initialisations

On travaille au niveau des feuilles, à savoir $N - 1$. La sous-liste des initialisations porte le numéro 1, et elle est créée par l'algorithme 2.1.

$\mathcal{L}_1 \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau $N - 1$ **do**
  $\mathcal{L}_1 = \mathcal{L}_1 \cup$ `initialisation`($\mathcal{C}$)
end for

Algorithme 2.1: Initialisation

Par la notation « $\mathcal{L}_1 = \mathcal{L}_1 \cup$ `initialisation`($\mathcal{C}$) » on désigne l'ajout de l'opération « `initialisation`($\mathcal{C}$) » à la sous-liste $\mathcal{L}_1$. L'usage du symbole ensembliste $\cup$ est un petit abus de notation ici, car $\mathcal{L}_1$ n'est pas un ensemble au sens mathématique, c'est une liste ordonnée.

2.1.3.2 Sous-listes des montées

Pour $l = N-1, \ldots, N_{plaf}+1$, la sous-liste des montées du niveau l au niveau $l-1$ porte le numéro $i = 2(N-l)+1$. Elle est créée par l'algorithme 2.2.

$\mathcal{L}_i \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau l **do**
  $\mathcal{L}_i = \mathcal{L}_i \cup$ `montée`$(\mathcal{C})$
end for

Algorithme 2.2: Montée

2.1.3.3 Sous-listes des descentes

Pour $l = N_{plaf}+1, \ldots, N-1$, la sous-liste des descentes du niveau $l-1$ au niveau l porte le numéro $i = 2(N-N_{plaf})+l-N_{plaf}$. Elle est créée par l'algorithme 2.3.

$\mathcal{L}_i \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau l **do**
  $\mathcal{L}_i = \mathcal{L}_i \cup$ `descente`$(\mathcal{C})$
end for

Algorithme 2.3: Descente

2.1.3.4 Sous-listes des transferts

Pour $l = N-1, \ldots, N_{plaf}$, la sous-liste des transferts au niveau l porte le numéro $i = 2(N-l)$. Pour sa construction, dans l'algorithme 2.4 on distingue le niveau plafond (où les transferts se font entre cellules non-voisines) et les autres niveaux (où les transferts se font entre cellules banlieues).

2.1.3.5 Sous-liste des intégrations

On travaille à nouveau au niveau des feuilles, à savoir $N-1$. La sous-liste des intégrations porte le numéro $i = 3(N-N_{plaf})$, et elle est crée par l'algorithme 2.5.

2.1.3.6 Initialisation des compteurs

Les valeurs maximales des compteurs de contribution et d'utilisation sont calculées au fur et à mesure de la création des listes. Par exemple, si on ajoute « `transfert`$(\mathcal{C},\mathcal{C}')$ » à la liste $\mathcal{L}_i$, il faut rajouter une unité au compteur de contribution de $\mathcal{G}_{\mathcal{C}'}$ et au compteur d'utilisation de $\mathcal{F}_{\mathcal{C}}$, conformément au paragraphe 2.1.2.4. Une fois les sous-listes créées, les valeurs maximales de ces compteurs ne bougent plus. Seules sont modifiées les valeurs dites courantes de ces compteurs.

$\mathcal{L}_i \leftarrow \varnothing$
if $(l = N_{plaf})$ **then**
  for all $\mathcal{C}, \mathcal{C}'$ cellules du niveau l **do**
    if ($\mathcal{C}$ non-voisine de $\mathcal{C}'$) **then**
      $\mathcal{L}_i = \mathcal{L}_i \cup$ `transfert`$(\mathcal{C}, \mathcal{C}')$
    end if
  end for
else
  for all $\mathcal{C}, \mathcal{C}'$ cellules du niveau l **do**
    if ($\mathcal{C}$ banlieue de $\mathcal{C}'$) **then**
      $\mathcal{L}_i = \mathcal{L}_i \cup$ `transfert`$(\mathcal{C}, \mathcal{C}')$
    end if
  end for
end if

Algorithme 2.4: Transferts

$\mathcal{L}_i \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau $N - 1$ **do**
  $\mathcal{L}_i = \mathcal{L}_1 \cup$ `intégration`$(\mathcal{C})$
end for

Algorithme 2.5: Intégrations

2.2 Traitement des opérations de transfert

On présente dans cette section le traitement des transferts de fonction de radiation dans la méthode multipôle rapide. On aborde tout d'abord l'expression générale d'une fonction de transfert, puis son écriture creuse. On détaille ensuite la manière de diminuer le nombre de fonctions utilisées. On présente enfin les écarts en temps et en précision entre ces différentes variantes.

2.2.1 Expressions de la matrice de transfert

2.2.1.1 Cadre général

On va étudier le traitement des opérations de transfert au sein d'un calcul multipôle. Rappelons qu'on appelle « transfert » l'opération consistant à calculer l'action d'une cellule source $\mathcal{C}_1$ sur une cellule destination $\mathcal{C}_2$ qui est non-voisine (mais du même niveau) de $\mathcal{C}_1$. On dispose donc de la fonction d'émission $\mathcal{F}_{\mathcal{C}_1}$ de $\mathcal{C}_1$, et on cherche à calculer sa contribution à la fonction de réception $\mathcal{G}_{\mathcal{C}_2}$ de $\mathcal{C}_2$. Cette contribution s'écrit $T^L_{O_2-O_1}.\mathcal{F}_{\mathcal{C}_1}$, où O_1 et O_2 sont les centres respectifs des cellules $\mathcal{C}_1$ et $\mathcal{C}_2$. $T^L_{O_2-O_1}$ est l'opérateur de transfert de la cellule $\mathcal{C}_1$ vers la cellule $\mathcal{C}_2$.

Numériquement, $\mathcal{F}_{\mathcal{C}_1}$ et $\mathcal{G}_{\mathcal{C}_2}$ sont définies sur la sphère unité $\mathcal{S}$ sur une grille de points notés (θ_i,ϕ_j) avec $0 \leq i \leq L$ et $0 \leq j \leq 2L$ où L est le nombre de pôles à ce niveau de l'arbre. La matrice de transfert $T^L_{O_2-O_1}$ est définie sur la même grille. L'opération de transfert revient à multiplier terme à terme la fonction de transfert par la fonction de radiation $\mathcal{F}_{\mathcal{C}_1}$, et à rajouter le résultat à $\mathcal{G}_{\mathcal{C}_2}$. C'est donc une opération vectorielle qui s'écrit :

```
DO I=1,N
  G2(I) = G2(I) + T(I)*F1(I)
ENDDO
```

Cette opération, quoique basique, n'est pas une opération d'algèbre linéaire standard disponible dans les bibliothèques optimisées de type BLAS. On s'intéresse dans la suite à la manière d'accélérer le traitement de ce type d'opération dans le calcul multipôle.

2.2.1.2 Ecriture classique d'une matrice de transfert

2.2.1.2.1 Expression mathématique L'expression générale de l'opérateur de transfert est la suivante :

$$T^L_{\vec{R}}(\vec{s}) = \sum_{0\leq l\leq L} (2l+1) i^l h_l^{(1)}(k.R) P_l(cos(\vec{s},\vec{R})) \tag{2.1}$$

où L est le nombre de pôles au niveau de l'arbre considéré, $\vec{R}$ est le vecteur de transfert liant le centre de la cellule source au centre de la cellule destination, $h_l^{(1)}$ est la fonction de Hankel sphérique du premier type, k est le nombre d'onde du problème fréquentiel traité, P_l est le polynôme de Legendre de rang l, et $\vec{s} \in \mathcal{S}$ est la variable appartenant à la sphère unité $\mathcal{S}$. Dans le pratique, $\vec{s}$ parcourt notre grille de points précalculés (θ_i,ϕ_j) de $\mathcal{S}$.

2.2.1.2.2 Algorithme Le calcul des fonctions de Hankel peut être une source d'erreurs ou d'inexactitudes. Nous avons utilisé la fonction `SPHBES` extraite de *Numerical Recipes* [30] avec laquelle nous n'avons rencontré aucun problème particulier. Pour les polynômes de Legendre P_l, nous avons utilisé la formule de récurrence :

$$\begin{cases} P_l(x) = \dfrac{2l-1}{l} x P_{l-1}(x) - \dfrac{l-1}{l} P_{l-2}(x) & \forall l \geq 2, \\ P_0(x) = 1, \\ P_1(x) = x \end{cases}$$

On cherche à minimiser les opérations coûteuses (en particulier trigonométrie et division). Le calcul effectif se fait ainsi :

1. Indépendamment de $\vec{s}$, pour un vecteur de transfert $\vec{R}$ donné, on calcule tout d'abord les coefficients $(2l+1)i^l h_l^{(1)}(k.R)$ ainsi que les coefficients de la récurrence $\frac{2l-1}{l}$ et $\frac{l-1}{l}$ pour tout $l \geq 2$.
2. Pour chaque direction $\vec{s}$, on a directement $cos(\vec{s},\vec{R}) = \vec{s}.\frac{\vec{R}}{\|R\|}$. On note x ce produit scalaire. On calcule par récurrence tous les $P_l(x)$ pour $0 \leq l \leq L$. Il ne reste plus qu'à sommer les produits $(2l+1)i^l h_l^{(1)}(k.R)P_l(x)$ pour obtenir $T_{\vec{R}}^L(\vec{s})$.

Remarques : Les polynômes de Legendre sont alternativement pairs et impairs, on a en effet $P_l(-x) = (-1)^l P_l(x)$, donc en notant pour simplifier x_l la quantité :

$$x_l = (2l+1)i^l h_l^{(1)}(k.R)P_l(x)$$

on obtient :

$$\begin{cases} T_{\vec{R}}^L(\vec{s}) = \displaystyle\sum_{0 \leq l \leq L} x_l, \\ T_{\vec{R}}^L(-\vec{s}) = \displaystyle\sum_{0 \leq l \leq L} (-1)^l x_l \end{cases}$$

Grâce à cette remarque, on parvient à diviser par deux le nombre d'opérations. Il faut juste s'assurer que $\vec{s}$ et $-\vec{s}$ sont tous deux dans la grille de points (θ_i,ϕ_j) utilisée (dans le cas de la quadrature usuelle du paragraphe 1.2.2.2.2.4, cela impose un nombre de points ϕ_j qui soit pair).

2.2.1.3 Ecriture creuse d'une matrice de transfert

2.2.1.3.1 Objectif Dans la pratique, l'opération de transfert va représenter une part prépondérante dans un produit matrice-vecteur multipôle. Il est donc crucial de minimiser le nombre d'éléments non nuls contenus dans la matrice de transfert. Cela se fait bien sûr en ajustant le nombre de pôles à chaque niveau de l'arbre (Si L est le nombre de pôles, la taille de la matrice est $O(L^2)$). Une autre idée est de chercher à rendre creuse ces matrices de transfert.

2.2.1.3.2 Première tentative

2.2.1.3.2.1 Introduction Mathématiquement, les termes $T_{\vec{R}}^L(\vec{s})$ sont tous non-nuls. Numériquement, cette fonction atteint son maximum lorsque $\vec{s}$ est colinéaire à $\vec{R}$, et décroît lorsque $\vec{s}$ s'en éloigne. Cette décroissance est d'autant plus rapide que $\|\vec{R}\|$ est grand devant la longueur d'onde λ. On peut donc envisager de réaliser un seuillage : on choisit un seuil

$\epsilon = 10^{-3}$ par exemple, on détermine le terme de plus grand module de la matrice de transfert $T^L_{\vec{R}}(\vec{s_{max}})$, et on met à zéro tous les termes dont le module est inférieur à $\epsilon.\|T^L_{\vec{R}}(\vec{s_{max}})\|$. Étudions l'efficacité de cette opération.

2.2.1.3.2.2 Taux de remplissage Le tableau 2.1 donne le pourcentage d'éléments non-nuls après seuillage pour différentes valeurs de ϵ. Ce taux de remplissage est une moyenne calculée sur l'ensemble des fonctions de radiation susceptibles d'apparaître lors d'un calcul multipôle (elles sont en nombre fini, comme on le verra plus loin). On donne les résultats pour différentes tailles de cellule correspondant aux différents niveaux de l'arbre.

Rayon des cellules	Nombre de Pôles	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$
$\lambda/8$	8	95.72	99.91	100.00	100.00	100.00
$\lambda/4$	12	90.28	99.61	100.00	100.00	100.00
$\lambda/2$	19	71.84	96.98	99.87	100.00	100.00
λ	32	46.32	87.33	98.61	99.98	100.00
2λ	56	32.94	70.78	95.50	99.84	100.00
4λ	101	23.48	50.66	89.29	98.99	99.98
8λ	190	17.26	36.09	82.28	97.96	99.95
16λ	367	13.72	25.03	62.30	94.44	99.54
32λ	717	12.13	19.54	47.50	90.84	98.97

TAB. 2.1 – *Taux de remplissage des matrices de transfert*

Comme prévu, le nombre d'éléments sous le seuil augmente avec la taille des cellules. Malheureusement, seule la plus grande valeur de ϵ permet d'obtenir un taux de remplissage inférieur à 50 % pour des cellules de petites tailles. Dans la pratique, un seuillage avec une telle valeur de ϵ est bien trop brutal. Les autres valeurs de ϵ testées sont utilisables en pratique, mais le taux de remplissage reste très élevé, sauf pour des tailles de cellule vraiment importantes (correspondant en pratique à des calculs sur plusieurs dizaines ou centaines de millions d'inconnues).

2.2.1.3.2.3 Précision On regarde maintenant l'impact de ce seuillage sur la précision globale du calcul. Pour ce faire, on calcule l'écart relatif en norme $\mathcal{L}^2$ entre un produit matrice-vecteur « classique » et un produit matrice-vecteur multipôle pour différentes valeurs de ϵ. On réalise ce test sur quatre maillages :

- une sphère de diamètre 4λ maillée avec 17328 inconnues ;
- une sphère de diamètre 10λ maillée avec 112908 inconnues ;
- un avion d'envergure 14λ maillé avec 23676 inconnues ;
- un avion d'envergure 29λ maillé avec 94704 inconnues.

On indique dans le tableau 2.2 la taille de la plus grande fonction de transfert utilisée lors du calcul FMM sur chacun de ces maillages. Le vecteur utilisé représente une illumination par une onde plane. Soulignons que le but ici n'est pas de regarder la précision de la méthode multipôle en elle-même, mais plutôt la manière dont cette précision évolue sous l'effet du seuillage des matrices de transfert.

Maillage Plus grande cellule	Sphère 17328 $\lambda/2$	Sphère 112908 2λ	Avion 23676 2λ	Avion 94704 4λ
$\epsilon = 0$	$1{,}29.10^{-3}$	$1{,}39.10^{-3}$	$7{,}30.10^{-3}$	$8{,}53.10^{-3}$
$\epsilon = 10^{-3}$	$1{,}29.10^{-3}$	$1{,}39.10^{-3}$	$7{,}30.10^{-3}$	$8{,}53.10^{-3}$
$\epsilon = 3.10^{-3}$	$1{,}29.10^{-3}$	$1{,}39.10^{-3}$	$7{,}30.10^{-3}$	$8{,}53.10^{-3}$
$\epsilon = 10^{-2}$	$1{,}29.10^{-3}$	$1{,}39.10^{-3}$	$7{,}30.10^{-3}$	$8{,}53.10^{-3}$
$\epsilon = 3.10^{-2}$	$1{,}66.10^{-3}$	$2{,}04.10^{-3}$	$7{,}62.10^{-3}$	$8{,}94.10^{-3}$
$\epsilon = 10^{-1}$	$1{,}27.10^{-1}$	$1{,}32.10^{-1}$	$2{,}34.10^{-1}$	$2{,}54.10^{-1}$

TAB. 2.2 – *Variation de l'erreur en fonction du seuillage*

2.2.1.3.2.4 Conclusion On constate que la différence ne devient sensible qu'à partir de $\epsilon = 3\%$, et seule la valeur 10^{-1} induit une erreur vraiment excessive. Malheureusement, si on rapproche ces résultats de ceux du tableau 2.1, on constate que les valeurs « admissibles » de ϵ ne génèrent pas de bons taux de remplissage. Par exemple, la sphère à 112908 inconnues pour $\epsilon = 10^{-2}$ bénéficiera au mieux d'un taux de remplissage de 95,50% pour les transferts entre cellules de rayon 2λ.

Par conséquent, le seuillage direct des fonctions de transfert n'apporte rien.

2.2.1.3.3 Version améliorée

2.2.1.3.3.1 Présentation On peut améliorer les résultats de la section précédente en changeant le mode de troncature de la série définissant $T^L_{\vec{R}}(\vec{s})$. Au lieu de couper net au rang L, on se propose de rajouter encore quelques termes au delà du rang L, en les faisant précéder d'un coefficient de lissage. On définit donc :

$$\begin{aligned} T^{L,L'}_{\vec{R}}(\vec{s}) = & \sum_{0\leq l\leq L} \left[(2l+1)i^l h^{(1)}_l(k.R)\right] P_l(cos(\vec{s},\vec{R})) \\ & + \sum_{L+1\leq l\leq L'} C^{L,L'}_l \left[(2L+1)i^L h^{(1)}_L(k.R)\right] P_l(cos(\vec{s},\vec{R})) \end{aligned} \tag{2.2}$$

Le coefficient de lissage $C^{L,L'}_l$ est donné par :

$$C^{L,L'}_l = \cos^2(\frac{l-L}{L'-L}\frac{\pi}{2})$$

$C^{L,L'}_l$ varie continûment de 1 à 0 lorsque l varie de L à L'. Dans la pratique on choisit L' sous la forme $L' = \alpha L$ où α est bien sûr supérieur ou égal à 1, le cas $\alpha = 1$ nous ramène à la section précédente. Dans la deuxième somme de (2.2), le terme entre crochets dépend de L et non de l'indice de sommation l, afin d'éviter la divergence des fonctions de Hankel $h^{(1)}_l(k.R)$ pour $l \gg k.R$.

La justification de ce choix se trouve par exemple dans [15]. Elle y est donnée dans le cadre d'une optimisation des translations, mais l'argument est le même ici. L'idée en 1D est qu'une fonction « abrupte », comme une fonction indicatrice $1_{[-L,L]}(x)$, aura une transformée de Fourier à décroissance lente loin de l'origine, tandis que la même fonction « lissée » (par exemple avec un $\cos^2$) aura une décroissance plus rapide. C'est une illustration du fait que la

transformation de Fourier échange la régularité et la décroissance à l'infini. Dans notre cas, sur $\mathcal{S}$, pour obtenir une meilleur décroissance de $T^L_{\vec{r_0}}(\vec{s})$ lorsque $\vec{s}$ s'éloigne de $\frac{\vec{r_0}}{r_0}$, on va chercher à lisser les coefficients des harmoniques sphériques de T^L plutôt que de les couper brutalement au rang L comme on le fait dans (2.1).

D'après [29], on a :

$$P_l(cos(\vec{s},\vec{r_0})) = \frac{4\pi}{2l+1} \sum_{-l \leq m \leq l} Y_{l,m}(\vec{s}) Y^*_{l,m}(\frac{\vec{r_0}}{r_0})$$

La fonction de transfert $T^L_{\vec{R}}(\vec{s})$ peut alors s'écrire :

$$T^L_{\vec{r_0}}(\vec{s}) = \sum_{0 \leq l \leq L} \sum_{-l \leq m \leq l} \left[4\pi i^l h^{(1)}_l(k.|\vec{r_0}|) Y^*_{l,m}(\frac{\vec{r_0}}{r_0}) \right] Y_{l,m}(\vec{s})$$

Dans l'expression de $T^L_{\vec{R}}(\vec{s})$, le terme $P_l(cos(\vec{s},\vec{R}))$ est le seul responsable de la présence des harmoniques sphériques de rang l. Rajouter des termes en P_l pour $l = L+1, \ldots, L'$ est donc un bon moyen d'enrichir le « spectre » de la fonction de transfert sans interférer avec les coefficients déjà présents. Et pour assurer une décroissance lisse, on fait précéder P_l du facteur $C^{L,L'}_l \left[(2L+1) i^L h^{(1)}_L(k.R) \right]$ qui représente le produit du coefficient de P_L et d'un terme allant de 1 à 0 de manière suffisamment lisse. D'où l'expression (2.2).

Nous allons étudier le taux de remplissage et la précision multipôle obtenus pour α variant de 1,2 à 2 par pas de 0,2.

2.2.1.3.3.2 Taux de remplissage On présente à nouveau le taux de remplissage moyen des matrices de transfert après seuillage, en fonction de $\epsilon = 10^{-3}$ et de α, pour les cellules de rayon λ uniquement (tableau 2.2). On constate que les valeurs de α supérieures à 1 améliorent sensiblement l'efficacité du seuillage des matrices de transfert, même pour une taille de cellule raisonnable comme ici.

Regardons les résultats pour les cellules de rayon 4λ (tableau 2.3). A l'exception du cas $\epsilon = 10^{-3}$, on obtient de très bon taux de remplissage, souvent inférieurs à 25 %.

Regardons enfin le cas des cellules de rayon 16λ (tableau 2.4). Les matrices de transfert de cette taille interviendront pour des objets de diamètre 100λ et plus, pour lesquels il est crucial d'optimiser le traitement des niveaux élevés de l'arbre.

Selon la valeur de ϵ et de α, on divise par 4, 10 voire même 20 le nombre d'éléments non nuls, accélérant d'autant les opérations de transfert.

2.2.1.3.3.3 Précision Il convient de voir dans quelle mesure la modification de l'expression des termes de la matrice de transfert, ainsi que le seuillage qui leur est appliqué dégrade la précision du produit multipôle.

Les tableaux suivants présentent, pour différents cas-tests et pour différentes valeurs de α et de ϵ, l'écart relatif L_2 existant entre un produit matrice-vecteur classique et multipôle. On note $E_{\alpha,\epsilon}$ cette erreur relative. Le cas $\alpha = 1$ et $\epsilon = 0$ est notre valeur de référence (correspondant à des matrices de transfert non modifiées). Dans la mesure où notre seuillage doit nécessairement se faire au prix d'une dégradation de la précision, il faut fixer un critère de rejet. Dans l'optique d'écrire un code multipôle pouvant ajuster sa précision à la demande,

α	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$
1.0	46.32	87.33	98.61	99.98	100.00
1.2	20.41	44.27	79.79	97.03	99.73
1.4	15.34	33.18	64.04	94.24	99.47
1.6	10.84	26.23	51.53	89.49	98.75
1.8	8.73	22.06	44.92	84.95	97.98
2.0	6.79	19.74	41.10	80.23	97.15

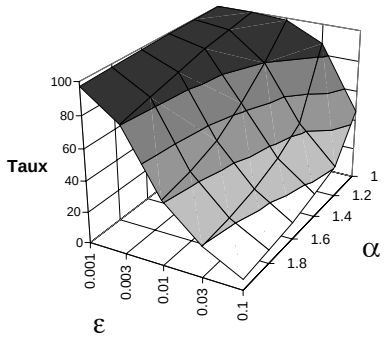


FIG. 2.2 – *Taux de remplissage des matrices de transfert de rayon λ (Version améliorée)*

α	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$
1.0	23.48	50.66	89.29	98.99	99.98
1.2	11.61	20.08	32.86	64.00	91.38
1.4	6.36	15.69	25.44	47.95	79.07
1.6	3.85	13.86	21.89	40.40	69.91
1.8	2.45	12.32	19.84	35.28	63.81
2.0	1.53	10.73	18.22	31.70	58.87

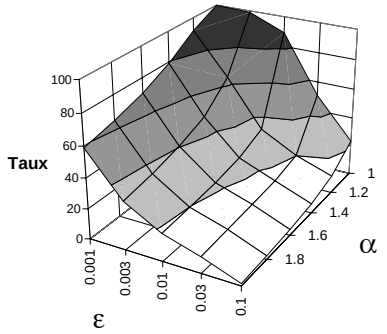


FIG. 2.3 – *Taux de remplissage des matrices de transfert de rayon 4λ (Version améliorée)*

α	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$
1.0	13.72	25.03	62.30	94.44	99.54
1.2	2.49	10.05	13.38	20.95	38.04
1.4	0.47	5.82	11.46	16.03	26.84
1.6	0.13	3.77	10.41	14.44	22.27
1.8	0.09	2.43	9.49	13.56	20.20
2.0	0.07	1.55	8.35	12.86	18.58

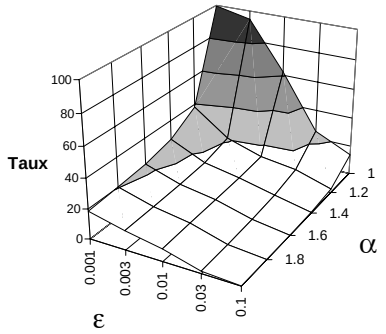


FIG. 2.4 – *Taux de remplissage des matrices de transfert de rayon* 16λ *(Version améliorée)*

on se donne un critère « large » égal à 8 fois l'erreur de référence $E_{1,0}$ et un critère « strict » égal à 2 fois l'erreur de référence.

On a tracé la grandeur $log_2(E_{\alpha,\epsilon}/E_{1,0})$ en fonction de α et de ϵ. La zone blanche correspond aux couples (α,ϵ) passant le critère strict : $log_2(E_{\alpha,\epsilon}/E_{1,0}) \in [0,1]$. La bande gris clair correspond au critère large : $log_2(E_{\alpha,\epsilon}/E_{1,0}) \in [1,3]$. Enfin la zone gris foncé correspond à des valeurs de (α,ϵ) ne satisfaisant aucun critère.

Regardons tout d'abord la sphère à 17328 inconnues (figure 2.5). La colonne $\epsilon = 0$ supprime le seuillage et permet de tester le seul paramètre α. On y voit que l'erreur tend à augmenter avec α. Les combinaisons (α,ϵ) passant le critère strict sont ($\alpha = 1{,}0; \epsilon \leq 3.10^{-2}$), ($\alpha = 1{,}2; \epsilon \leq 3.10^{-3}$), ($\alpha = 1{,}4; \epsilon \leq 3.10^{-3}$) et ($\alpha = 1{,}6; \epsilon \leq 10^{-3}$). Pour le critère large, on rajoute les combinaisons $(1{,}2; 10^{-2})$, $(1{,}4; 10^{-2})$et $(1{,}6; 3.10^{-3})$. Sur ce cas, on rejette définitivement la valeur $\alpha = 2{,}0$ et $\epsilon = 0{,}1$.

Regardons maintenant le cas de l'avion à 23676 inconnues (figure 2.6). Cet objet est beaucoup plus gros et complexe que la sphère précédemment testée. Le maillage est moins fin ($\lambda/7{,}5$ contre $\lambda/10$ pour les sphères), moins régulier. L'erreur relative y est plus importante. Néanmoins on constate que les couples (α,ϵ) passant le critère strict ou large sont les mêmes à quelques exceptions près : pour le critère strict, on rajoute les combinaisons $(1{,}2; 10^{-2})$, $(1{,}4; 10^{-2})$ et $(1{,}6; 3.10^{-3})$. Pour le critère large, on rajoute les combinaisons $(1{,}6; 10^{-2})$, $(1{,}8; 10^{-2})$ et $(1{,}8; 3.10^{-3})$.

Passons maintenant à des maillages plus importants. Regardons tout d'abord le cas sphérique à 112908 inconnues (figure 2.7). Il est important de souligner que sur nos machines de tests (deux stations DEC 500 MHz en réseau), le produit multipôle nécessite seulement 40 secondes, contre près de 20 heures pour le produit classique. On constate que les combinaisons (α, ϵ) passant les critères larges et stricts sont exactement les mêmes sur ce maillage que sur la sphère à 17328 inconnues.

Regardons enfin un maillage d'avion à 94704 inconnues, correspondant à une envergure de

α	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$	$\epsilon = 0$
1.0	$1{,}27.10^{-1}$	$1{,}66.10^{-3}$	$1{,}29.10^{-3}$	$1{,}29.10^{-3}$	$1{,}29.10^{-3}$	$1{,}29.10^{-3}$
1.2	$1{,}90.10^{-1}$	$4{,}82.10^{-2}$	$4{,}61.10^{-3}$	$1{,}30.10^{-3}$	$1{,}29.10^{-3}$	$1{,}29.10^{-3}$
1.4	$2{,}60.10^{-1}$	$9{,}69.10^{-2}$	$9{,}24.10^{-3}$	$1{,}33.10^{-3}$	$1{,}33.10^{-3}$	$1{,}33.10^{-3}$
1.6	$3{,}01.10^{-1}$	$1{,}29.10^{-1}$	$2{,}03.10^{-2}$	$3{,}43.10^{-3}$	$2{,}20.10^{-3}$	$2{,}21.10^{-3}$
1.8	$3{,}47.10^{-1}$	$1{,}53.10^{-1}$	$5{,}23.10^{-2}$	$1{,}77.10^{-2}$	$1{,}74.10^{-2}$	$1{,}74.10^{-2}$
2.0	$2{,}80.10^{-1}$	$1{,}77.10^{-1}$	$1{,}21.10^{-1}$	$1{,}06.10^{-1}$	$1{,}05.10^{-1}$	$1{,}05.10^{-1}$

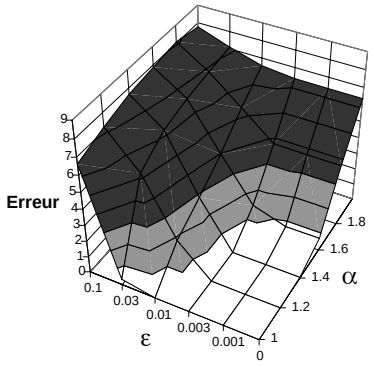


FIG. 2.5 – *Sphère 17328 : erreur due aux transferts*

α	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$	$\epsilon = 0$
1.0	$7{,}62.10^{-3}$	$7{,}30.10^{-3}$	$7{,}30.10^{-3}$	$7{,}30.10^{-3}$	$7{,}30.10^{-3}$
1.2	$6{,}96.10^{-2}$	$1{,}02.10^{-2}$	$7{,}31.10^{-3}$	$7{,}32.10^{-3}$	$7{,}32.10^{-3}$
1.4	$1{,}74.10^{-1}$	$1{,}68.10^{-2}$	$7{,}45.10^{-3}$	$7{,}44.10^{-3}$	$7{,}44.10^{-3}$
1.6	$1{,}83.10^{-1}$	$3{,}57.10^{-2}$	$9{,}31.10^{-3}$	$8{,}28.10^{-3}$	$8{,}28.10^{-3}$
1.8	$2{,}23.10^{-1}$	$8{,}37.10^{-2}$	$3{,}17.10^{-2}$	$3{,}13.10^{-2}$	$3{,}13.10^{-2}$

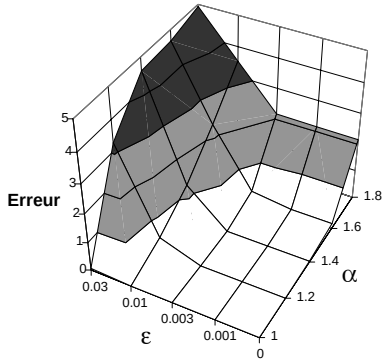


FIG. 2.6 – *Airbus 23676 : erreur due aux transferts*

α	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$	$\epsilon = 0$
1.0	$2{,}04.10^{-3}$	$1{,}39.10^{-3}$	$1{,}39.10^{-3}$	$1{,}39.10^{-3}$	$1{,}39.10^{-3}$
1.2	$4{,}66.10^{-2}$	$5{,}04.10^{-3}$	$1{,}40.10^{-3}$	$1{,}39.10^{-3}$	$1{,}39.10^{-3}$
1.4	$1{,}05.10^{-1}$	$1{,}09.10^{-2}$	$1{,}56.10^{-3}$	$1{,}42.10^{-3}$	$1{,}42.10^{-3}$
1.6	$1{,}19.10^{-1}$	$2{,}33.10^{-2}$	$3{,}44.10^{-3}$	$2{,}27.10^{-3}$	$2{,}27.10^{-3}$
1.8	$1{,}40.10^{-1}$	$4{,}98.10^{-2}$	$1{,}68.10^{-2}$	$1{,}64.10^{-2}$	$1{,}64.10^{-2}$

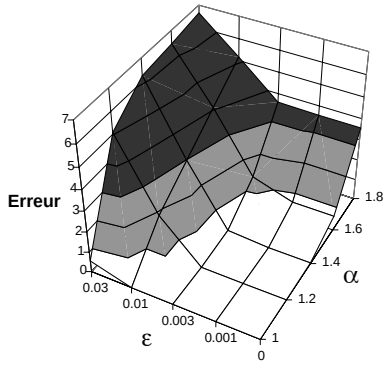


FIG. 2.7 – *Sphère 112908 : erreur due aux transferts*

α	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$	$\epsilon = 0$
1.0	$8{,}94.10^{-3}$	$8{,}53.10^{-3}$	$8{,}53.10^{-3}$	$8{,}53.10^{-3}$	$8{,}53.10^{-3}$
1.2	$8{,}13.10^{-2}$	$1{,}23.10^{-2}$	$8{,}53.10^{-3}$	$8{,}53.10^{-3}$	$8{,}53.10^{-3}$
1.4	$2{,}05.10^{-1}$	$2{,}05.10^{-2}$	$8{,}67.10^{-3}$	$8{,}61.10^{-3}$	$8{,}61.10^{-3}$
1.6	$2{,}09.10^{-1}$	$4{,}17.10^{-2}$	$1{,}06.10^{-2}$	$9{,}40.10^{-3}$	$9{,}40.10^{-3}$
1.8	$2{,}47.10^{-1}$	$9{,}60.10^{-2}$	$3{,}27.10^{-2}$	$3{,}20.10^{-2}$	$3{,}20.10^{-2}$

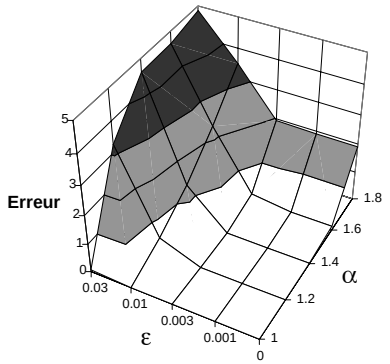


FIG. 2.8 – *Airbus 94704 : erreur due aux transferts*

29λ (figure 2.8). Les combinaisons (α,ϵ) passant le critère strict sont ($\alpha = 1{,}0; \epsilon \leq 3.10^{-2}$), ($\alpha = 1{,}2; \epsilon \leq 10^{-2}$), ($\alpha = 1{,}4; \epsilon \leq 3.10^{-3}$) et ($\alpha = 1{,}6; \epsilon \leq 3.10^{-3}$). Pour le critère large, on rajoute les combinaisons ($1{,}4; 10^{-2}$), ($1{,}6; 10^{-2}$) et ($1{,}8; \epsilon \leq 3.10^{-3}$).

2.2.1.3.3.4 Synthèse des résultats On a résumé tous ces résultats dans le tableau 2.3 en retenant pour chaque combinaison (α,ϵ) le critère le plus sévère sur les 4 cas étudiés. On a reproduit pour les cas pertinents le taux de remplissage des matrices de transfert pour $r = 16\lambda$, 4λ , λ.

α	rayon	$\epsilon = 10^{-1}$	$\epsilon = 3.10^{-2}$	$\epsilon = 10^{-2}$	$\epsilon = 3.10^{-3}$	$\epsilon = 10^{-3}$	$\epsilon = 0$
1.0		aucun	strict	strict	strict	strict	strict
	$r = 16\lambda$		25.03	62.30	94.44	99.54	100.
	$r = 4\lambda$		50.66	89.29	98.99	99.98	100.
	$r = \lambda$		87.33	98.61	99.98	100.	100.
1.2		aucun	aucun	large	strict	strict	strict
	$r = 16\lambda$			13.38	20.95	38.04	100.
	$r = 4\lambda$			32.86	64.00	91.38	100.
	$r = \lambda$			79.79	97.03	99.73	100.
1.4		aucun	aucun	large	strict	strict	strict
	$r = 16\lambda$			11.46	16.03	26.84	100.
	$r = 4\lambda$			25.44	47.95	79.07	100.
	$r = \lambda$			64.04	94.24	99.47	100.
1.6		aucun	aucun	aucun	large	strict	strict
	$r = 16\lambda$				14.44	22.27	100.
	$r = 4\lambda$				40.40	69.91	100.
	$r = \lambda$				89.49	98.75	100.
1.8		aucun	aucun	aucun	aucun	aucun	aucun
	$r = 16\lambda$						
	$r = 4\lambda$						
	$r = \lambda$						
2.0		aucun	aucun	aucun	aucun	aucun	aucun
	$r = 16\lambda$						
	$r = 4\lambda$						
	$r = \lambda$						

TAB. 2.3 – *Critère de précision et Taux de remplissage en fonction de α et ϵ*

Le choix optimal est celui qui, pour un critère de précision donné, minimise le taux de remplissage des matrices de transfert.

- Pour la FMM « stricte », le choix optimal est ($\alpha = 1{,}4; \epsilon = 3.10^{-3}$).
- Pour la FMM « large », le choix optimal est ($\alpha = 1{,}4; \epsilon = 10^{-2}$).

Pour les trois niveaux les plus bas de l'arbre, quelle que soit la valeur de α, le taux de remplissage reste proche de 100%. On prend donc ($\alpha = 1; \epsilon = 0$) à ces niveaux (i.e. on y utilise les matrices de transfert classiques).

2.2.2 Optimisation visant à diminuer le nombre de matrices calculées

Considérons le cas de notre maillage d'avion à 23676 inconnues. Un produit matrice-vecteur multipôle sur ce maillage fait appel à près de 130.000 opérations de transfert, et à autant de matrices de transfert à calculer. Par défaut, cela va représenter 80 % du temps nécessaire à une itération de FMM. Nous allons présenter une série d'améliorations qui va ramener le nombre de matrices de transfert calculées à 137 (pour ce cas à 23676 inconnues), et diviser le temps nécessaire aux transferts par 10.

2.2.2.1 Réordonnancement des opérations

Le vecteur de transfert $\vec{R}$ relie le centre de la cellule source au centre de la cellule destination. Ces dernières étant intégrées à la structure d'un octree, leurs centres sont placés sur une grille tridimensionnelle dont le pas dans les directions x, y, et z est égale à l'arête notée a des 2 cellules concernées. Par conséquent, le vecteur $\vec{R}$ a des coordonnées qui sont des multiples entiers de a. On peut donc écrire :

$$\vec{R} = \begin{pmatrix} n_x.a \\ n_y.a \\ n_z.a \end{pmatrix}$$

où n_x, n_y et n_z sont entiers relatifs. La taille de l'objet étant finie, il existe un nombre fini de matrices de transfert à un niveau donné de l'arbre.

Dans le cas d'une méthode multipôle multi-niveau complète, c'est-à-dire explorant la totalité de l'arbre, le vecteur de transfert $\vec{R}$ relie deux cellules non-voisines dont les parents respectifs sont voisins. Le fait que les cellules soient non-voisines implique :

$$(n_x,n_y,n_z) \notin [-1,+1]^3$$

Que les pères soient voisins impose en outre

$$(n_x,n_y,n_z) \in [-3,+3]^3$$

Le nombre de triplets (n_x,n_y,n_z) vérifiant cela est $7^3 - 3^3 = 316$.

Dans le cas d'une méthode multipôle partiellement multi-niveau, la situation est légèrement différente au niveau « plafond », puisque la condition sur les pères disparaît. Si on est au niveau n de l'arbre, on peut juste affirmer que

$$(n_x,n_y,n_z) \in [-(2^n-1),2^n-1]^3$$

Le nombre de matrices de transfert différentes est donc inférieur à $(2^{n+1}-1)^3 - 3^3$.

Dans tous les cas le nombre de matrices de transfert utilisées est largement inférieur au nombre de transferts effectivement réalisés. Par conséquent, chaque matrice de transfert assemblée doit pouvoir servir plusieurs fois. Il suffit pour cela de réordonner les transferts de façon à regrouper ceux utilisant la même matrice. Concrètement, à chaque niveau de l'arbre (c'est-à-dire pour chaque sous-liste de transfert) et pour chaque transfert d'une cellule $\mathcal{C}_1$ vers une cellule $\mathcal{C}_2$, on calcule le vecteur de transfert $\vec{R} = M_{\mathcal{C}_2} - M_{\mathcal{C}_1}$, puis le triplet (n_x,n_y,n_z). On trie ensuite toutes ces opérations à l'intérieur de la sous-liste sur la base de ce triplet (un algorithme de tri rapide comme quicksort fait ça très bien). Le gain est immédiat : au lieu de calculer une matrice par transfert (soit environ 50 matrices par cellule), on calcule au plus 316 matrices par niveau (dans le cas d'un calcul multi-niveau complet).

2.2.2.2 Utilisation des symétries

Il existe un autre moyen de diminuer le nombre de matrices de transfert calculées. Regardons pour cela l'expression générale de l'opérateur $T^L_{\vec{R}}$:

$$T^L_{\vec{R}}(\vec{s}) = \sum_{0 \leq m \leq L} (2m+1) i^m h^{(1)}_m(k.R) P_m(cos(\vec{s},\vec{R}))$$

Dans cette écriture, $\vec{s}$ appartient à la sphère unité $\mathcal{S}$. On peut donc l'écrire sous la forme

$$\vec{s} = \begin{bmatrix} \sin\theta.\cos\phi \\ \sin\theta.\sin\phi \\ \cos\theta \end{bmatrix}$$

avec la notation abrégée $\vec{s} = (\theta,\phi)$. On note par ailleurs le vecteur de transfert

$$\vec{R} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = a. \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

Soit P un plan passant par l'origine, et S_P la symétrie orthogonale par rapport à ce plan. On a alors :

$$\begin{cases} \|S_P(\vec{R})\| = \|\vec{R}\|, \\ cos(\vec{s},S_P(\vec{R})) = cos(S_P(\vec{s}),\vec{R}) \end{cases}$$

D'où l'on déduit le résultat suivant :

$$T^L_{S_P(\vec{R})}(\vec{s}) = T^L_{\vec{R}}(S_P(\vec{s}))$$

Autrement dit, si l'on connaît $T^L_{\vec{R}}$, on connaît immédiatement $T^L_{S_P(\vec{R})}$ pour tout plan P passant par l'origine. Dans la pratique, $\vec{s}$ appartient à une grille (θ_i,ϕ_j) donc il faut choisir P pour que $S_P(\vec{s})$ soit aussi dans cette grille. On suppose dans la suite que l'on utilise la grille dite usuelle de la section 1.2.2.2.2.4. Il faut bien sûr choisir P pour que $S_P(\vec{R})$ soit lui-même un vecteur de transfert. Les coordonnées de $\vec{R}$ étant des multiples entiers de l'arête des cellules a, toute symétrie S_P qui échange 2 coordonnées de $\vec{R}$, ou qui change le signe d'une coordonnées de $\vec{R}$, transformera $\vec{R}$ en un autre vecteur de transfert « candidat ». Voici quatre plans P répondant à cette double attente :

Plan (xOy)**:** La symétrie S_{xOy} change le signe de la coordonnées z, et transforme θ en $\pi-\theta$:

$$\begin{cases} S_{xOy}(\vec{R}) = (r_x,r_y,-r_z), \\ S_{xOy}(\vec{s}) = (\pi-\theta,\phi) \end{cases}$$

Plan (xOz)**:** La symétrie S_{xOz} change le signe de la coordonnées y, et transforme ϕ en $2\pi-\phi$:

$$\begin{cases} S_{xOz}(\vec{R}) = (r_x,-r_y,r_z), \\ S_{xOz}(\vec{s}) = (\theta,2\pi-\phi) \end{cases}$$

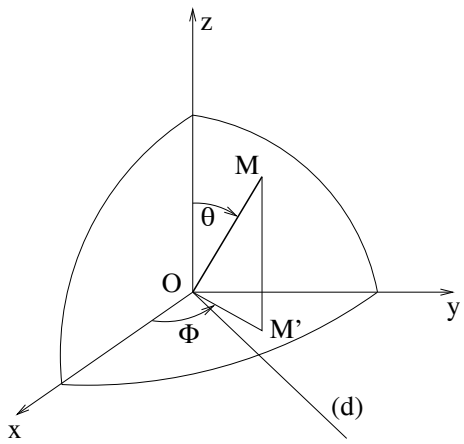


FIG. 2.9 – *Coordonnées cartésiennes et sphériques*

Plan (yOz)**:** La symétrie S_{yOz} change le signe de la coordonnées x, et transforme ϕ en $\pi-\phi$:

$$\begin{cases} S_{yOz}(\vec{R}) = (-r_x,r_y,r_z), \\ S_{yOz}(\vec{s}) = (\theta,\pi-\phi) \end{cases}$$

$(\pi-\phi_j)$ sera dans la grille (ϕ_j) dès lors que le nombre de points selon ϕ sera pair.

Plan (zOd)**:** On note d la bissectrice de l'angle (xOy). La symétrie S_{zOD} échange les coordonnées x et y, et transforme ϕ en $\pi/2-\phi$:

$$\begin{cases} S_{zOD}(\vec{R}) = (r_y,r_x,r_z), \\ S_{zOD}(\vec{s}) = (\theta,\pi/2-\phi) \end{cases}$$

$(\pi/2-\phi_j)$ sera dans la grille (ϕ_j) dès lors que le nombre de points selon ϕ sera multiple de 4.

Deux matrices de transfert $T^L_{\vec{R}}$ et $T^L_{\vec{R}'}$ peuvent être déduites l'une de l'autre dès qu'il existe une combinaison de ces 4 symétries qui transforme $\vec{R}$ en $\vec{R}'$. Il est nécessaire et suffisant pour cela d'avoir :

$$\begin{cases} \max(|r_x|,|r_y|) = \max(|r'_x|,|r'_y|), \\ \min(|r_x|,|r_y|) = \min(|r'_x|,|r'_y|), \\ |r_z| = |r'_z| \end{cases}$$

En effet, si ces trois égalités sont vérifiées, l'application de S_{zOD} permet d'ôter les opérateurs max et min, puis les 3 autres symétries permettent d'ôter les valeurs absolues. Au final, au lieu de devoir calculer les fonctions de radiation pour tout triplet (n_x,n_y,n_z) dans $[-3,+3]^3-[-1,+1]^3$, on peut se contenter de calculer ces matrices pour $(n_x,n_y,n_z) \in [0,+3]^3-[0,+1]^3$ avec $n_x \leq n_y$. Cela représente seulement 34 triplets possibles (au lieu de 316). A partir de ces 34 matrices de transfert, on peut par simple permutation des éléments calculer toutes les autres.

Dans le cas (évoqué en introduction) d'un avion à 23676 inconnues, la FMM utilise effectivement 5 niveaux, donc le nombre de fonctions de transfert est ramené à (au plus) $5\times34=170$. En pratique, sur ce cas on en calcule seulement 137.

Pour tirer pleinement partie de cela, au lieu de trier les opérations de transfert dans chaque sous-liste comme expliqué au paragraphe précédent sur la base du triplet (n_x,n_y,n_z), on réalise ce tri sur la base du triplet $(\max(|n_x|,|n_y|), \min(|n_x|,|n_y|),|n_z|)$.

Néanmoins, comme cette optimisation impose de prendre un nombre de points selon ϕ multiple de 4, nous avons choisi de ne pas l'activer aux plus bas niveaux de l'arbre. En effet, si par exemple $L = 8$ au plus bas niveau, la discrétisation usuelle conduit à une grille (θ_i,ϕ_j) de taille $(L + 1) \times (2L + 1) = 9 \times 17$ qui devient 9×20 après arrondi au multiple de 4 supérieur, soit presque 20% de points en plus. Le gain sur le calcul des matrices de transfert sera complètement anéanti par le surcoût des transferts proprement dit. On activera cette utilisation des symétries seulement pour $L \geq 50$.

2.2.2.3 Conservation des matrices antérieures

La méthode multipôle est utilisée à l'intérieur d'un solveur itératif. Il est donc naturel de vouloir préserver des données d'une itération à l'autre, comme par exemple les matrices de transfert. Initialement nombreuses (jusqu'à 316 par niveau) et pleines, on vient de voir comment en diminuer leur nombre (au plus 34 par niveau) et leur taille. Il est donc maintenant très économique de conserver celles-ci d'une itération à l'autre.

Deux autres arguments à long terme penchent en faveur de l'implémentation de cette fonctionnalité. Tout d'abord, elle permet d'améliorer la scalabilité parallèle du code. En effet, le calcul d'une matrice de transfert donnée a de grandes chances de se produire sur plusieurs processeurs, voire sur tous. C'est une partie du calcul qui n'est donc pas du tout parallélisée. Même si elle représente une toute petite part du temps total dans un code séquentiel, son influence va aller en grandissant avec le nombre de processeurs.

Par ailleurs, le temps de calcul d'une matrice de transfert est proportionnel à L^3 où L est le nombre de pôles du niveau considéré. En effet, les nombres de ligne et de colonne sont proportionnels à L, et chaque élément est une somme de L termes (le fait que ces matrices puissent être rendues creuses n'y change rien, car le seuillage n'intervient qu'après l'assemblage). Or L est quasi proportionnel à la taille des cellules. Le nombre de pôles du niveau l est 2 fois celui du niveau $l + 1$, le temps d'assemblage des matrices de transfert du niveau l est 8 fois celui du niveau $l + 1$. Par conséquent, à chaque doublement de la taille de l'objet (en longueurs d'onde), on ajoute un niveau à l'octree et on multiplie par 8 le temps global de calcul des matrices de transfert (qui a une complexité en $(kd)^3$). Il n'est donc pas souhaitable de relancer ce calcul à chaque itération, surtout pour les niveaux élevés. Aux niveaux bas, le temps d'assemblage des matrices de transfert est par contre négligeable. C'est pourquoi on ne conservera les matrices antérieures que pour $L \geq 30$ (conjointement donc avec l'utilisation des symétries).

2.2.3 Synthèse

Les phases de transfert représentent par défaut la partie la plus coûteuse en terme de temps dans la méthode multipôle. C'est pour cette raison que de nombreux développements ont été réalisés autour du calcul et de l'utilisation des matrices de transfert.

Afin de souligner le caractère indispensable des modifications introduites ici, étudions le cas d'une sphère de rayon 7,5λ maillée avec 255.792 inconnues. Le tableau 2.4 compare la FMM « avant » et « après » l'optimisation des transferts. On y indique le temps CPU pour 1 itération de FMM (sur un pentium III 733 MHz), la part de ce temps utilisée pour le seul

traitement des transferts, le nombre de matrices de transfert calculées, leur taille cumulée en mémoire vive (dans le cas où elles sont stockées), et la précision du calcul (à savoir ici l'écart relatif en norme $\mathcal{L}^2$ avec un produit matrice-vecteur « classique »).

	Avant	Après		
		Critère « large »	Critère « strict »	Sans seuillage
Temps CPU	409 s	82 s	84 s	87 s
Dont Transfert	86 %	51 %	50 %	47 %
Nombre de matrices	924.040	170	170	170
Taille cumulée	-	172.067	270.433	332.928
Précision	$1{,}29.10^{-3}$	$7{,}59.10^{-3}$	$1{,}66.10^{-3}$	$1{,}29.10^{-3}$

TAB. 2.4 – *Sphère à 250.000 inconnues : gains obtenus*

Le gain est significatif. A titre de comparaison, le produit matrice-vecteur classique utilisé comme référence ici a nécessité 9 jours de calculs sur une station DEC Alpha 500 MHz. L'optimisation du nombre de matrices de transfert calculées est fondamentale, quelle que soit la taille du problème. La possibilité de creuser et conserver ces matrices sert surtout pour les calculs de très grande taille (typiquement à partir de 10^6 inconnues).

2.3 Traitement des opérations de montée/descente

On présente dans cette section le traitement des opérations de montée et de descente dans le cadre de la méthode multipôle rapide pour les équations de Maxwell.

2.3.1 Expressions continues des montées/descentes

Soit une cellule $\mathcal{C}$ de centre $M_{\mathcal{C}}$, de parent $p(\mathcal{C})$ centré en $M_{p(\mathcal{C})}$, et une fonction de radiation $\mathcal{F}_{\mathcal{C}}$ associée à cette cellule. L'opération de montée consiste à calculer :

$$\mathcal{F}_{p(\mathcal{C})}(\vec{s}) = e^{ik\vec{s}.M_{\mathcal{C}}\vec{M}_{p(\mathcal{C})}}\mathcal{F}_{\mathcal{C}}(\vec{s})$$

On a vu à la section 1.3.1.6 que cette opération combinait en fait deux phases : le changement de centre (aussi appelé translation), et le changement de grille (appelé extrapolation). Pour être tout à fait exact, il faut souligner que la formule ci-dessus ne donne pas directement $\mathcal{F}_{p(\mathcal{C})}$ mais seulement la partie de cette fonction provenant de $\mathcal{C}$. Chaque enfant de $p(\mathcal{C})$ apportera une contribution.

Dans le cas d'une descente, on se donne $\mathcal{G}_{p(\mathcal{C})}$ la fonction de radiation associée à la cellule parent $p(\mathcal{C})$, et on calcule la contribution de $p(\mathcal{C})$ à $\mathcal{G}_{\mathcal{C}}$:

$$\mathcal{G}_{\mathcal{C}}(\vec{s}) = e^{ik\vec{s}.\vec{M_{p(\mathcal{C})}M_{\mathcal{C}}}}\mathcal{G}_{p(\mathcal{C})}(\vec{s})$$

Cette fois-ci, on combine une translation et une réduction (opération transposée de l'extrapolation). Nous allons voir dans la suite comment traiter chacune de ces opérations, et dans quel ordre les combiner.

2.3.2 Translation

2.3.2.1 Expression des matrices de translation

On note $\vec{r}$ le vecteur de la translation, reliant $M_{\mathcal{C}}$ et $M_{p(\mathcal{C})}$ (dans un sens ou dans l'autre suivant que l'on traite une montée ou une descente). On note $(\vec{s_j})$ la quadrature de $\mathcal{S}$ au niveau considéré (qui sera celui de $\mathcal{C}$ ou celui de $p(\mathcal{C})$ cf. 2.3.4). Enfin, on se donne deux fonctions de radiations $\mathcal{F}_1$ et $\mathcal{F}_2$. L'expression discrète de l'opération de translation est :

$$\mathcal{F}_2(\vec{s_j}) = e^{ik\vec{s_j}.\vec{r}}\mathcal{F}_1(\vec{s_j})$$

On note $\tau_{\vec{r}}$ la matrice de translation définie sur notre quadrature $(\vec{s_j})$ par $\tau_{\vec{r}}(\vec{s}) = e^{ik\vec{s}.\vec{r}}$. L'application de $\tau_{\vec{r}}$ à $\mathcal{F}_1$ prend exactement la même forme que pour un transfert (cf. section 2.2.1.1).

Dans le cas où la quadrature de $\mathcal{S}$ utilisée est une grille, on peut exploiter le fait que $\tau_{\vec{r}}(\vec{s})$ est le complexe conjugué[1] de $\tau_{\vec{r}}(-\vec{s})$ pour accélérer le remplissage de $\tau_{\vec{r}}$. On pourra faire cela à condition que le nombre de points dans la direction ϕ soit pair. En effet, dans ce cas si $\vec{s} = (\theta,\phi)$ est dans notre grille « usuelle », alors $-\vec{s} = (\pi - \theta,\phi + \pi)$ y sera aussi.

2.3.2.2 Conservation des matrices

Comme dans le cas des transferts, le nombre de matrices de translation différentes est très inférieur au nombre de translations à réaliser. En effet, le vecteur de translation relie le centre d'une cellule au centre de son parent. Par conséquent, si on traite les montées ou les descentes entre les niveaux l et $l-1$, il n'existe que 8 vecteurs de translation possible, à savoir :

$$\vec{r} = \frac{a^{(l)}}{2}\begin{bmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \end{bmatrix}$$

On pourrait réordonner les sous-listes de montées et de descentes pour rassembler les translations utilisant le même vecteur. Mais compte tenu du petit nombre de matrices différentes (seulement 8), on a choisi de toutes les conserver en mémoire jusqu'à la fin de la sous-liste en cours d'exécution. Elles sont ensuite éliminées, il n'est pas utile de les conserver pour l'itération multipôle suivante car leur calcul est très rapide (contrairement aux matrices de transferts). Vu la forme du vecteur $\vec{r}$, on aurait pu également utiliser des propriétés de symétrie pour déduire les matrices de translation les unes des autres, mais là encore le bénéfice à en tirer est négligeable.

	Sphère 255972	Airbus 94704
Sans conservation		
Temps	14,4 s	11,5 s
% du total	17,9 %	25,1 %
Avec conservation		
Temps	10,9 s	9,6 s
% du total	12,9 %	22,1 %

TAB. 2.5 – *Gain obtenu en conservant les matrices de translation*

1. Cela n'est vrai que si le nombre d'ondes est réel, si k est complexe alors $\tau_{\vec{r}}(\vec{s})$ est l'inverse de $\tau_{\vec{r}}(-\vec{s})$

Le tableau 2.5 indique le gain en temps obtenu en conservant les matrices de translation. On y indique, pour une itération multipôle, le temps consacré à faire les montées et les descentes en secondes et en pourcentage du total. Le gain est modeste, mais il est simple à implémenter, et ne s'accompagne d'aucune perte de précision.

2.3.3 Extrapolation et réduction

Nous allons maintenant regarder la manière de traiter une extrapolation et l'opération transposée, la réduction. Les modifications seront présentées pour les extrapolations, mais il est sous-entendu qu'elles s'appliquent de même aux réductions. On considère le passage entre les niveaux l et $l-1$.

2.3.3.1 Méthode de base

La méthode de base consiste à utiliser le produit matrice-vecteur de l'équation (1.57). La matrice comporte autant de lignes qu'il y a de points dans la quadrature de $\mathcal{S}$ au niveau l, et autant de colonnes qu'il y a de points dans la quadrature de $\mathcal{S}$ au niveau $l-1$, soit au total environ $4{L^{(l)}}^2{L^{(l-1)}}^2$. L'élément (k,k') de la matrice vaut :

$$\begin{aligned} M_{k,k'} &= \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(l)}}} Y^*_{l,m}(\vec{s_k}^{(l)})Y_{l,m}(\vec{s_{k'}}^{(l-1)}) \\ &= \sum_{0\leq l\leq L^{(1)}} \frac{2l+1}{4\pi}P_l(\cos\theta) \end{aligned}$$

où θ est l'angle entre les vecteurs $\vec{s_k}^{(l)}$ et $\vec{s_{k'}}^{(l-1)}$, autrement dit :

$$\cos\theta = \vec{s_k}^{(l)}.\vec{s_{k'}}^{(l-1)}$$

puisque ces deux vecteurs sont unitaires. Sous la deuxième forme, c'est assez rapide à calculer. Si les points $\vec{s_k}^{(l)}$ et $\vec{s_{k'}}^{(l-1)}$ diffèrent, on peut même utiliser la formule (1.58) pour simplifier encore l'écriture de la matrice. Ceci dit, vu le nombre d'opérations mises en jeu, c'est une méthode excessivement lente comparée aux autres. Par exemple, dans le cas de la sphère 255792 du tableau 2.6, la transition entre les deux niveaux les plus hauts utilise une matrice de dimension 6612×2244 qui pèse 120 Mo, et est appelée dans 112 produits matrice-vecteur. Le seul intérêt de cette approche est sa simplicité de programmation.

2.3.3.2 Méthode rapide condensée

La deuxième méthode testée est celle décrite à la section 1.3.1.6.2.2. Elle comporte trois étapes : une transformée de Fourier directe, un produit matrice-vecteur, une FFT inverse. Elle peut s'écrire de deux manières, suivant la forme que l'on donne au produit matrice-vecteur. Nous les avons appelées « dissociée » et « condensée ». Nous regardons ici la seconde.

2.3.3.2.1 Transformée de Fourier Pour le traitement des FFT, nous avons utilisé la bibliothèque FFTW [21]. Elle offre des performances excellentes, parfois supérieures aux librairies du constructeur de la machine utilisée, et reste simple à mettre en œuvre. Le *Numerical Recipes* [30] propose un algorithme plus simple, mais qui impose des tailles de vecteur qui soient des puissances de 2.

2.3.3.2.2 Matrice des Q_l^m Rappelons l'expression (1.59) du produit matrice-vecteur qui nous intéresse :

$$\tilde{F}_{i',m}^{(l-1)} = \sum_i \left[\sum_{|m|\leq l\leq L^{(l)}} Q_l^m(cos\theta_i^{(l)}) Q_l^m(cos\theta_{i'}^{(l-1)}) \right] \tilde{F}_{i,m}^{(l)} \tag{2.3}$$

On voit apparaître des coefficients $Q_l^m(cos\theta) = C_{l,m}P_l^m(cos\theta)$ qui doivent être calculés avec soin. En effet, pour de grands indices, $C_{l,m}$ devient très petit et $P_l^m(cos\theta)$ très grand. Si on en fait le produit, des erreurs d'arrondi vont apparaître. C'est pourquoi on utilise une formule de récurrence dérivée de la formule suivante :

$$\begin{cases} (l-m)P_l^m(x) - (2l-1)xP_{l-1}^m(x) + (l+m-1)P_{l-2}^m(x) = 0 \qquad 0 \leq m \leq l, \\ P_m^m(x) = (-1)^m(1-x^2)^{m/2}\dfrac{(2m)!}{2^m m!}, \\ P_{m+1}^m(x) = (2m+1)xP_m^m(x) \end{cases}$$

Le cas $m < 0$ est obtenu par parité. La formule déduite pour Q_l^m s'écrit :

$$\begin{cases} \sqrt{l^2-m^2}Q_l^m(x) - (2l-1)xQ_{l-1}^m + \sqrt{(l-1)^2-m^2}Q_{l-2}^m = 0 \qquad 0 \leq m \leq l, \\ Q_m^m(x) = (-1)^m(1-x^2)^{m/2}\dfrac{\sqrt{(2m)!}}{2^m m!}, \\ Q_{m+1}^m(x) = \sqrt{2m+1}xQ_m^m(x) \end{cases}$$

Grâce à cette récurrence, on évite tout problème dans le calcul des Q_l^m.

Cette méthode d'extrapolation/réduction donne la même précision que la précédente. Le tableau 2.6 donne les temps obtenus avec cette écriture, on peut voir qu'ils sont nettement meilleurs.

2.3.3.3 Méthode rapide dissociée

Cette méthode est une variante de la précédente. On utilise la formule de Christofell-Darboux (1.60) pour simplifier la matrice des Q_l^m :

$$\begin{aligned} \sum_{|m|\leq l\leq L^{(l)}} Q_l^m(x)Q_l^m(x') = & \sqrt{\frac{(L^{(l)}+1)^2-m^2}{4(L^{(l)}+1)^2-1}} \\ & \times \left[\frac{Q_{L^{(l)}+1}^m(x')Q_{L^{(l)}}^m(x)}{x'-x} - \frac{Q_{L^{(l)}+1}^m(x)Q_{L^{(l)}}^m(x')}{x'-x} \right] \end{aligned} \tag{2.4}$$

Le produit matrice-vecteur est dissocié en trois parties (détaillée page 56) : on multiplie les composantes du vecteur en entrée, on fait le produit par la matrice C, enfin on multiplie les composantes du résultat.

La matrice C, de taille $(L^{(l-1)}+1)\times(L^{(l)}+1)$, est définie par :

$$C_{i',i} = \frac{1}{cos\theta_{i'}^{(l-1)} - cos\theta_i^{(l)}} \tag{2.5}$$

L'avantage de cette dissociation du produit matrice-vecteur est que l'on n'a pas à assembler la matrice (2.3) pour chaque valeur de m. En outre, on peut utiliser une FMM 1D pour faire les produits par C. On a vu que cela est indispensable pour diminuer la complexité asymptotique de la méthode.

L'inconvénient est que dans (2.4) on fait deux produits par C pour chaque produit initial. De surcroît, si $cos\theta_{i'}^{(l-1)} = cos\theta_i^{(l)}$, l'élément $C_{i',i}$ est indéfini. Or, cela a de grands risques de se produire. En effet, si le nombre de pôles est pair, le nombre de θ est impair (avec la discrétisation dite usuelle), et le point $\cos\theta = 0$ fait partie de la quadrature. Du coup, si le nombre de pôles est pair à deux niveaux consécutifs, on va rencontrer la situation $cos\theta_{i'}^{(l-1)} = cos\theta_i^{(l)} = 0$. Il faut donc gérer ce cas à part. En pratique, avec les $cos\theta_i^{(l)}$ calculés par la méthode du paragraphe 1.2.2.2.2.4, on ne trouve jamais $\cos\theta = 0$ mais $\cos\theta = \varepsilon$, et le problème mentionné n'arrive jamais. Néanmoins, il est plus prudent d'essayer de contourner cet écueil : aux niveaux les plus hauts, il suffit d'imposer à $L^{(l)}$ d'être alternativement pair et impair (avec une perte de performance minime), aux niveaux les plus bas on calcule le terme entre crochets dans (2.3) de manière explicite lorsque $cos\theta_{i'}^{(l-1)} = cos\theta_i^{(l)} = 0$.

2.3.3.4 Méthode creuse

La dernière méthode est juste évoquée ici. Elle reprend la méthode dite « de base », en tâchant de rendre creuse la matrice M. On utilise pour cela la même approche que pour les matrices de transfert. On applique un lissage plus un seuillage.

$$\begin{aligned} M_{k,k'} = & \sum_{0\le l\le L} \frac{2l+1}{4\pi} P_l(\cos\theta) \\ & + \sum_{L+1\le l\le L'} C_l^{L,L'} \frac{2l+1}{4\pi} P_l(\cos\theta) \end{aligned}$$

avec toujours le coefficient de lissage :

$$C_l^{L,L'} = \cos^2\left(\frac{l-L}{L'-L}\frac{\pi}{2}\right)$$

En toute logique, il faudrait faire la même étude ici que pour les transferts. Cela n'a pas été jugé prioritaire car à ce jour les montées/descentes demeurent beaucoup moins coûteuses que les transferts. On a pris $L' = 1{,}2L$ et pour le seuillage $\varepsilon = 10^{-3}$. Avec ces paramètres, on obtient déjà 70% d'éléments nuls dans la matrice 6612×2244 évoquée au paragraphe sur la méthode de base.

C'est théoriquement la méthode qui donne la plus faible complexité asymptotique, elle devrait donc être utilisée pour les plus hauts niveaux. Contrairement aux autres méthodes, cette approche induit une erreur due au seuillage.

Méthode	Sphère 255972	Airbus 23676
De base		
Temps	597,6 s	158,5 s
% du total	88,9 %	95 %
Rapide condensée		
Temps	10,0 s	1,8 s
% du total	13,5 %	19 %
Rapide dissociée		
Temps	14,8 s	2,3 s
% du total	16 %	22,5 %
Creuse		
Temps	492,6 s	105,2 s
% du total	87,1 %	93,3 %

TAB. 2.6 – *Comparaison des méthodes d'extrapolation/réduction*

2.3.3.5 Comparaison

Le tableau 2.6 présente les résultats de nos tests sur ces quatre types d'extrapolation/réduction. Pour chaque méthode, et pour chaque maillage, on donne le temps de calcul (en secondes) pour les seules opérations de montée/descente, et le pourcentage du temps total d'un produit multipôle que cela représente. Les trois premières méthodes (de base, rapide condensée et rapide dissociée) sont mathématiquement équivalentes, elles n'introduisent pas d'autre erreur que celle due à la précision finie des machines. La méthode creuse au contraire rajoute une erreur en introduisant un seuillage.

La méthode de base est totalement distancée. Le fait de devoir assembler et manipuler des matrices pleines de grande taille rend cette approche inutilisable, même pour des cas modestes. La méthode creuse est légèrement plus rapide, mais il faut souligner qu'elle n'a pas été testée et optimisée à fond contrairement aux autres. En outre, il faut garder à l'esprit que cette approche montrerait tout son potentiel sur de très grands cas de calcul. Les méthodes dites rapides sont clairement plus ... rapides. La méthode dissociée est arrivée seconde : le fait de ne pas avoir à assembler la matrice ne suffit pas à compenser les deux produits matrice-vecteur mis en œuvre. Enfin, la méthode rapide condensée est la plus performante ici. Elle utilise la formule de Christoffel-Darboux uniquement pour assembler la matrice des Q_l^m. Cette approche est la plus efficace.

2.3.4 Avant ou après ?

On revient sur le point abordé au paragraphe 1.3.1.6.5, et décrit sur la figure 2.10. Les montées et descentes sont composées de deux phases distincts : le changement de niveau et le changement de centre. La question est de savoir dans quel ordre on réalise ces deux changements.

Prenons le cas d'une montée, représenté figure 2.10. Si on réalise les changements de centre en premier, on ne fait ensuite qu'une seule extrapolation. Cette étape étant plus coûteuse que la translation, le calcul est plus rapide dans cet ordre. Mais la translation augmente la largeur de bande de la fonction translatée, donc cette méthode est a priori moins précise. De la même manière, au cours d'une descente, il est plus rapide de faire d'abord les translations puis une

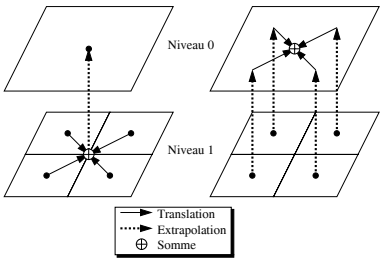


FIG. 2.10 – *Phase de montée*

seule réduction, et il est plus précis de faire d'abord les réductions puis les translations. On voit donc apparaître deux méthodes que nous appellerons méthode rapide et méthode précise.

On continue à utiliser la sphère 255972 et l'airbus 94704. On calcule l'écart entre un produit matrice-vecteur classique et un produit matrice-vecteur multipôle pour chacune des deux méthodes proposées, et pour chacun de ces deux objets. Les résultats obtenus, ainsi que les temps correspondants (en secondes et en pourcentage du total) sont présentés dans le tableau 2.7.

Méthode	Sphère 255972	Airbus 94704
Méthode rapide		
Temps	10,6 s	9,2 s
% du total	11,4%	17,6 %
Ecart	$1{,}29.10^{-3}$	$8{,}79.10^{-3}$
Méthode précise		
Temps	36,7 s	30,2 s
% du total	30,7 %	41,1%
Ecart	$1{,}28.10^{-3}$	$8{,}78.10^{-3}$

TAB. 2.7 – *Impact de l'ordre des opérations dans les montées/descentes*

On constate que la différence de précision entre les deux méthodes est négligeable. En revanche, l'écart en temps est considérable. Lors d'un changement de niveau, la méthode rapide fait une extrapolation (ou une réduction) par parent, contre une par enfant pour la méthode lente. Or, en moyenne, chaque parent a quatre enfants (on ne conserve que les cellules non-vides, cf. figure 2.11). Donc la méthode rapide divise environ par quatre le nombre d'extrapolations et réductions (qui sont la partie coûteuse des montées/descentes). On retrouve cela dans les chiffres du tableau 2.7 : la méthode rapide prend 3,2 à 3,4 fois moins de temps.

2.3.5 Récapitulatif

En théorie, les phases de montée/descente ont la complexité la plus élevée. C'est du moins ce qu'on a constaté à la section 1.3.2.3 consacrée au comptage des opérations. Dans la pratique, le temps consacré à cette partie du calcul est compris entre 10 et 20 %. On a vu qu'il y avait de nombreuses manières de réaliser ces opérations. La plus efficace combine la conservation des matrices, la méthode d'extrapolation/réduction rapide condensée, et la minimisation du nombre de ces extrapolations/réductions. A plus long terme, lorsque l'on abordera des cas pour lesquels la complexité en $O({n_{dl}}^{3/2})$ de cette approche posera problème, il conviendra de

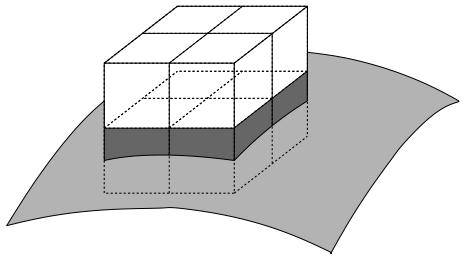


FIG. 2.11 – *Quatre enfants par parent en moyenne*

s'intéresser de plus près à la méthode creuse, ou à l'utilisation de FMM 1D pour les produits par la matrice (2.5). Notre étude théorique avait fixé ce seuil à 10^7 inconnues.

2.4 Traitement des autres opérations

On présente dans cette section le traitement des autres opérations mises en œuvre lors d'un produit multipôle, à savoir les initialisations, les intégrations, et les interactions proches.

2.4.1 Initialisations

L'étape d'initialisation réalise le passage des courants surfaciques $t(x)$ (avec $x \in \Gamma$) aux fonctions de radiations $\mathcal{F}_{\mathcal{C}}(\vec{s})$ (avec $\vec{s} \in \mathcal{S}$). La fonction de radiation associée à une feuille $\mathcal{C}$ centrée en M s'écrit de manière continue :

$$\mathcal{F}_{\mathcal{C}}(\vec{s}) = \int_{x \in \Gamma \cap \mathcal{C}} e^{ik\vec{s}.x\vec{M}} t(x) dx \qquad \vec{s} \in \mathcal{S} \tag{2.6}$$

Le calcul effectif de cette fonction passe par trois discrétisations.

Tout d'abord, la fonction de courant $t(x)$ se discrétise sur notre base d'éléments finis de la manière suivante :

$$t(x) = \sum_{1 \leq i \leq n_{dl}} \lambda_i \varphi_i(x) \qquad x \in \Gamma$$

Comme il a été vu à la section 1.2.2.3.1, l'intégrale sur $\Gamma \cap \mathcal{C}$ est en fait une intégrale sur les degrés de liberté inclus dans $\mathcal{C}$. L'équivalent de (2.6) s'écrit alors :

$$\mathcal{F}_{\mathcal{C}}(\vec{s}) = \int_{x \in \Gamma} e^{ik\vec{s}.x\vec{M}} \sum_{A_i \in \mathcal{C}} \lambda_i \varphi_i(x) dx \qquad \vec{s} \in \mathcal{S} \tag{2.7}$$

Ensuite, il faut discrétiser l'intégrale sur Γ : cela se fait en utilisant des points d'intégrations de Gauss notés G_j associés à des poids d'intégration p_j. Bien sûr, seuls les points G_j pour lesquels $\exists i / A_i \in \mathcal{C}$ et $\varphi_i(G_j) \neq 0$ contribuent à ce calcul. (2.7) devient :

$$\mathcal{F}_{\mathcal{C}}(\vec{s}) = \sum_{G_j} p_j \left[e^{ik\vec{s}.\vec{G_jM}} \sum_{A_i \in \mathcal{C}} \lambda_i \varphi_i(G_j) dx \right] \qquad \vec{s} \in \mathcal{S}$$

Enfin, on ne calcule $\mathcal{F}_{\mathcal{C}}(\vec{s})$ que pour les points $\vec{s}$ appartenant à notre quadrature $(\vec{s_k})$ de $\mathcal{S}$.

Comme à la section 2.3.2.1, on note $\tau_{\vec{r}}$ la matrice de translation définie sur notre quadrature $(\vec{s_k})$ par $\tau_{\vec{r}}(\vec{s_k}) = e^{iks\vec{s_k}.\vec{r}}$. Par ailleurs, on note $t_{\mathcal{C}}(x)$ la « restriction » de $t(x)$ aux degrés de liberté contenus dans $\mathcal{C}$, c'est-à-dire :

$$\vec{t_{\mathcal{C}}}(x) = \sum_{A_i \in \mathcal{C}} \lambda_i \vec{\varphi_i}(x)$$

En pratique, le calcul de $\mathcal{F}_{\mathcal{C}}$ sur la quadrature $(\vec{s_k})$ comporte trois étapes :

1. Pour chaque point G_j, calculer $p_j.\vec{t_{\mathcal{C}}}(G_j)$.
2. Multiplier la matrice $\tau_{\vec{G_jM}}$ par cette quantité.
3. Sommer les contributions de chaque point de Gauss G_j.

Les coefficients $\vec{\varphi_i}(G_j)$ avec $i = 1, \ldots, n_{dl}$ et $j = 1, \ldots, n_g$ (où n_g est le nombre de points de Gauss, typiquement 3 par triangle) sont presque tous nuls : pour un degré de liberté i donné, le support de $\vec{\varphi_i}$ est composé de deux triangles adjacents. Il y a donc seulement 6 points de Gauss pour lesquels $\vec{\varphi_i}(G_j) \neq 0$. On conservera la matrice creuse des $\vec{\varphi_i}(G_j)$ sur disque.

De la même manière, les matrices $\tau_{\vec{G_jM}}$ reviendront identiques à elle-même à chaque itération du produit multipôle lors des phases d'initialisation et d'intégration (comme on va le voir tout de suite). On peut donc aussi envisager de les stocker sur disque. On renvoie à la section 3.4.10 pour les tests correspondants.

On a présenté ici le cas le plus simple d'initialisation, correspondant à une fonction $t(x)$ scalaire. Si cette dernière est vectorielle, comme c'est le cas avec les équations de Maxwell, on fait les mêmes opérations pour chacune des composantes (x,y,z) et avec la divergence. Dans le cas d'une formulation multipôle à deux composantes, on rajoute à la fin du calcul le passage aux coordonnées sphériques tangentielles (équation (1.30)) qui ne présente pas de difficulté particulière.

2.4.2 Intégrations

L'opération d'intégration permet de repasser d'une fonction de radiation $\mathcal{G}_{\mathcal{C}}(\vec{s})$ aux degrés de liberté contenus dans $\mathcal{C}$. De manière plus précise, pour chaque fonction de base φ_i associée à la cellule $\mathcal{C}$ (de centre M), on calcule la contribution des interactions lointaines au produit matrice-vecteur via la formule :

$$\lambda_i^{(far)} = \int_{y \in \Gamma \cap \mathcal{C}} \int_{\vec{s} \in \mathcal{S}} \mathcal{G}_{\mathcal{C}}(\vec{s}) e^{ik\vec{s}.\vec{M}y} \varphi_i(y) d\vec{s} dy \tag{2.8}$$

On a volontairement omis le coefficient $ik/16\pi^2$ pour simplifier. De plus, on a repris ici l'expression la plus simple pour une intégration, on verra plus loin que les intégrations MFIE, CFIE ou à 2 termes se traitent de même.

Le résultat $\lambda_i^{(far)}$ de cette double intégration se rajoute au terme correspondant $\lambda_i^{(close)}$ issu du traitement des interactions proches. La somme des deux constitue la i-ème coordonnée du produit matrice-vecteur recherché. Le calcul de (2.8) nécessite deux discrétisations. D'une part, on discrétise l'intégrale sur Γ en utilisant des points et poids d'intégration de Gauss (G_j,p_j) (les mêmes que pour l'initialisation, bien sûr). D'autre part, l'intégrale sur la sphère unité $\mathcal{S}$ est calculée à l'aide de nos points et poids d'intégration $(\vec{s_k},\omega_k)$. L'expression de $\lambda_i^{(far)}$ devient :

$$\lambda_i^{(far)} = \sum_{G_j} p_j \left[\sum_k \omega_k \mathcal{G}_{\mathcal{C}}(\vec{s_k}) e^{iks\vec{s_k}.M\vec{G}_j} \varphi_i(G_j) \right]$$

On traite cela en trois temps :

1. On modifie la fonction de radiation en remplaçant $\mathcal{G}_{\mathcal{C}}(\vec{s_k})$ par $\omega_k \mathcal{G}_{\mathcal{C}}(\vec{s_k})$.
2. Pour chaque point G_j, on multiplie terme à terme cette fonction modifiée par la matrice de translation $\tau_{M\vec{G}_j}$ et on somme le résultat sur k (ce qui correspond à l'intégrale sur $\mathcal{S}$).
3. Pour chaque point G_j, le terme précédent est multiplié par $p_j \varphi_i(G_j)$, puis sommé sur les points de Gauss G_j.

Évidemment, seuls les quelques points G_j pour lesquels $\varphi_i(G_j) \neq 0$ sont concernés par ce calcul. On pourra ici réutiliser des données ayant déjà servi au moment des initialisations, comme par exemple les valeurs de $\varphi_i(G_j)$, ou bien encore les matrices de translation $\tau_{M\vec{G}_j}$ (qui sont les matrices complexes conjuguées des matrices $\tau_{G_j\vec{M}}$ issues de l'initialisation). Si la fonction de radiation est un vecteur $\vec{\mathcal{G}}_{\mathcal{C}}$, la deuxième étape crée un vecteur qui subit un produit scalaire avec $p_j \vec{\varphi}_i(G_j)$ à la dernière étape.

Regardons le cas d'une intégration plus élaborée comme celle de la CFIE page 34 que l'on reprend ici :

$$\begin{aligned}\lambda_i^{(far)} = \int_{y \in \Gamma \cap \mathcal{C}} \int_{\vec{s} \in \mathcal{S}} e^{ik\vec{s}.\vec{M}y} \Big[\alpha \left(\vec{\mathcal{G}}_{\mathcal{C}}(\vec{s}).\vec{\varphi}_i(y) \right) \\ + (1-\alpha) \left(\vec{s} \wedge \vec{\mathcal{G}}_{\mathcal{C}}(\vec{s}).\vec{\varphi}_i(y) \wedge \vec{\nu}(y) \right) \Big] \, d\vec{s} dy\end{aligned}$$

Le traitement est assez similaire : le premier terme, qui correspond à l'EFIE, est traité comme ci-dessus. Pour le second terme, on remplace $\vec{\mathcal{G}}_{\mathcal{C}}$ par $\vec{s} \wedge \vec{\mathcal{G}}_{\mathcal{C}}$, et $\varphi_i(G_j)$ par $\varphi_i(G_j) \wedge \vec{\nu}(G_j)$, et les trois étapes sont inchangées.

2.4.3 Interactions proches

La matrice des interactions proches (notée ici M) ne prend en compte que les interactions entre degrés de liberté se trouvant dans des feuilles voisines. C'est en effet le seul cas d'interaction qui ne peut pas être traité par la FMM. La matrice correspondante est donc creuse. La section 1.2.2.4.3 donne une approximation du nombre total d'éléments non-nuls dans la matrice creuse des interactions proches : pour une ligne ou une colonne de matrice donnée, le nombre moyen d'éléments non-nuls dépend de la taille des feuilles a et du nombre de points par longueurs d'onde d par la formule $9\sqrt{3}d^2a^2$. Le nombre total de non-zéros dans cette matrice est donc de l'ordre de $n_{dl}9\sqrt{3}d^2a^2$. La constante dépend beaucoup du maillage, mais la dépendance en n_{dl} est systématique.

Il existe deux manières de considérer cette matrice. La première approche consiste à raisonner de manière classique, c'est-à-dire en terme de degrés de liberté. Pour chaque indice de ligne i, on cherche tous les indices de colonne j pour lesquels $M_{i,j}$ sera non-nul. On calcule les termes correspondants, et on sauvegarde ainsi M, de préférence en utilisant un stockage creux de type *compressed sparse row*. On parlera d'approche « creuse ».

Mais il existe une approche plus efficace, qui consiste à raisonner en terme de cellules. Pour chaque cellule ligne $\mathcal{C}_i$, on cherche toutes les cellules colonnes $\mathcal{C}_j$ voisine de $\mathcal{C}_i$. L'interaction

entre $\mathcal{C}_i$ et $\mathcal{C}_j$ est une petite matrice pleine. En stockant directement ces petits blocs pleins, on peut donc gérer la matrice creuse M comme une succession de petites matrices pleines. On y gagne en simplicité, en temps d'assemblage, en taille de stockage, et en vitesse d'exécution pour les produits matrice-vecteur par M. La seule contrainte liée à cette approche est le besoin de réordonner les degrés de liberté dans l'ordre des feuilles où ils se situent, et ce avant et après chaque produit matrice-vecteur par M. Dans ce cas, on parlera d'approche « pleine » (ce qui ne signifie pas que la matrice est stockée pleine, mais qu'elle est composée de petits blocs qui, eux, le sont).

On compare ces deux approches dans le tableau 2.8. On y montre la taille de la matrice des interactions proches, le temps d'assemblage et le temps pour un produit matrice-vecteur (par M seule) pour différents maillages et pour chacune de ces deux approches.

	Sphère 6912	airbus23676	sphere255792
Approche creuse			
Assemblage (s)	106	410	25640
Produit (s)	0,26	0,71	14,7
Taille (Mo)	8,5	32,3	326
Approche pleine			
Assemblage (s)	10	41	407
Produit (s)	0,11	0,39	4
Taille (Mo)	6,7	23,9	253

TAB. 2.8 – *Comparaison des traitements des interactions proches*

Bien sûr, les résultats dépendent énormément de l'implémentation du code, et de l'architecture utilisée. En particulier, notre implémentation de l'approche creuse assemble la matrice M de taille $n_{dl} \times n_{dl}$ par blocs de (environ) 200×200. Pour chaque bloc, on essaie de savoir s'il est vide ou non avant de chercher à l'assembler. Or le nombre de blocs croît comme ${n_{dl}}^2$, donc le temps consacré à la détermination des blocs non-vides aussi. Ce qui fait que la procédure d'assemblage de M a un temps d'exécution proportionnel à ${n_{dl}}^2$ pour n_{dl} grand. L'approche pleine, qui raisonne directement sur les cellules, fait l'économie de ce type de calculs. L'écart monumental sur les temps d'assemblage est donc avant tout la conséquence de ce mode d'assemblage « par blocs » (qui au demeurant devient incontournable pour n_{dl} très grand).

En revanche, l'écart sur les temps de réalisation d'un produit matrice-vecteur est lié au stockage creux ou plein de la matrice. Il n'est pas surprenant qu'un produit entre un bloc plein et un vecteur aille deux fois plus vite qu'un produit entre un bloc creux et un vecteur (pour un même nombre d'éléments non-nuls).

Enfin, l'écart sur les tailles de matrice provient du mode de stockage utilisé : en creux, la compression CSR (pour *compressed sparse row*) utilise un entier plus un flottant par élément non-nul (soit 10 octets en simple précision), contre seulement un flottant (8 octets) par élément non-nul pour l'approche pleine (tout ceci est simplifié).

Il semble donc clair que l'approche pleine est plus rapide, plus compacte. Elle est en outre à nos yeux plus simple et plus naturelle.

2.5 Optimisation de la mémoire

On va s'intéresser maintenant à la gestion de la mémoire dans notre implémentation multipôle. Après avoir démontré à l'aide de quelques exemples la nécessité de gérer finement la mémoire, on présente une première approche permettant d'économiser facilement les ressources du système. On introduit ensuite notre formulation « out-of-core » qui permet de diminuer encore davantage la quantité de mémoire requise par la FMM. Enfin, on illustrera l'efficacité de ces développements sur quelques cas de grande taille.

2.5.1 Limitation

La méthode multipôle est très consommatrice de mémoire. Les données les plus volumineuses à traiter sont les fonctions de radiation. En reprenant les notations de la section 1.2.2.4.2, pour un niveau donné, on a $n_{\mathcal{C}} = n_{dl}/\sqrt{3}d^2a^2$ cellules, avec 2 fonctions de radiation par cellules, discrétisées chacune en $n_{\vec{s}} = 2L^2$ points sur $\mathcal{S}$. Soit $4L^2 n_{dl}/\sqrt{3}d^2a^2$ valeurs complexes par niveau. Avec N donné par (1.73), on voit que la quantité totale de mémoire requise pour stocker toutes les fonctions de radiation croît comme $n_{dl}\log(n_{dl})$. C'est certes inférieur au n_{dl}^2 d'une méthode « classique », mais cela reste important.

Avec L choisi via la formule simplifiée $L = \sqrt{3}ka\lambda = 2\pi\sqrt{3}a$, on trouve que tous les niveaux de l'octree occupent la même place en mémoire, égale à $16\sqrt{3}\pi^2 n_{dl}/d^2$ flottants complexes (16 octets chacun en double précision). En pratique, avec L donné par (3.1), ce sont les niveaux les plus bas qui sont les plus volumineux. Par exemple, pour une sphère à un million d'inconnues, les tailles des différents niveaux sont donnés dans le tableau 2.9. Les tailles indiquées correspondent à un seul type de fonctions de radiation ($\mathcal{F}$ ou $\mathcal{G}$), il faut multiplier par deux pour les avoir toutes. Ces valeurs ont été obtenues avec une formulation FMM à deux composantes (i.e. chaque fonction $\mathcal{F}$ ou $\mathcal{G}$ se décompose en deux fonctions de radiation scalaires). Avec des formulations multipôles à 3 ou 4 termes, les chiffres du tableau 2.9 augmentent de 50 % ou 100 % respectivement.

Niveaux	Taille (Mo)
2	17,6
3	23,0
4	32,2
5	44,3
6	71,0
7	155,3
Total	343,4

TAB. 2.9 – *Taille en mémoire des fonctions de radiation $\mathcal{F}$ ou $\mathcal{G}$ pour une sphère à 1 million d'inconnues*

La mémoire totale nécessaire (en l'absence d'optimisation) pour un tel cas de calcul est donc supérieure à 686,8 Mo (car il y a bien d'autres données à manipuler que les fonctions de radiation). Or, un calcul de cette taille peut être traité en mode multipôle en quelques heures sur une simple station de travail. On voit donc que si l'on n'optimise pas l'usage de la mémoire, elle sera très rapidement notre facteur limitant.

2.5.2 Réordonnancement des sous-listes

Une première amélioration va permettre de diviser quasiment par deux la quantité de mémoire nécessaire. Elle va consister à réordonner judicieusement les différentes étapes du calcul (c.à.d. les sous-listes) afin de libérer[2] le plus tôt possible les fonctions de radiation devenues inutiles. L'idée originale provient de [15].

La pire des méthodes est représentée sur la figure 2.12. Elle consiste à traiter les opérations dans l'ordre suivant :

1. Les initialisations (au niveau des feuilles) ;
2. Les montées (à tous les niveaux) ;
3. Les transferts (à tous les niveaux) ;
4. Les descentes (à tous les niveaux) ;
5. Les intégrations (au niveau des feuilles).

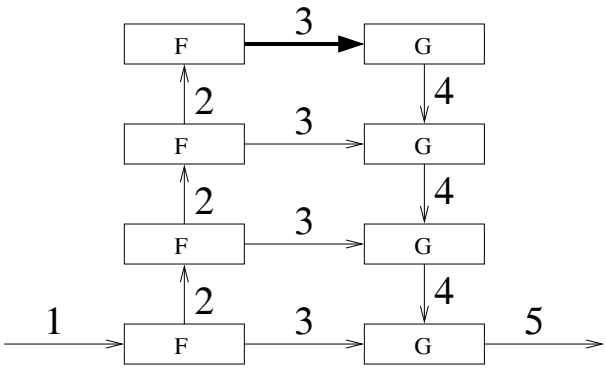


FIG. 2.12 – *Algorithme FMM gourmand*

Si on n'y prend pas garde, à la fin du traitement des transferts (étape 3), on a en mémoire vive la totalité des fonctions de radiation $\mathcal{F}$ et $\mathcal{G}$ pour tous les niveaux de l'arbre, dont on a vu que cela pouvait être assez considérable.

En revanche, si on ordonne les sous-listes de tâches du calcul multipôle comme on l'a fait à la section 2.1.3, et comme on le rappelle sur la figure 2.13, alors on va pouvoir consommer beaucoup moins de mémoire :

- Dès la fin de l'étape 3, on peut libérer les fonctions $\mathcal{F}$ du niveau $N-1$;
- De même après les étapes 5, 7 on peut libérer les fonctions $\mathcal{F}$ des niveaux $N-2$ et $N-3$ respectivement.

C'est donc au moment de l'étape 8 que la consommation de mémoire est maximale, car on a alors les fonctions $\mathcal{G}$ pour tous les niveaux, et les fonctions $\mathcal{F}$ du niveau plafond.

Si on reprend les chiffres du tableau 2.9, on voit que l'on passe de 686,8 Mo à 361 Mo, soit un gain de 47,4 %.

2.5.3 Réécriture out-of-core

On va proposer une méthode permettant de diminuer encore davantage les besoins en terme de mémoire vive. Un peu comme on l'a fait pour la FMM elle-même, nous allons proposer une

2. au sens informatique du terme, c'est-à-dire « désallouer »

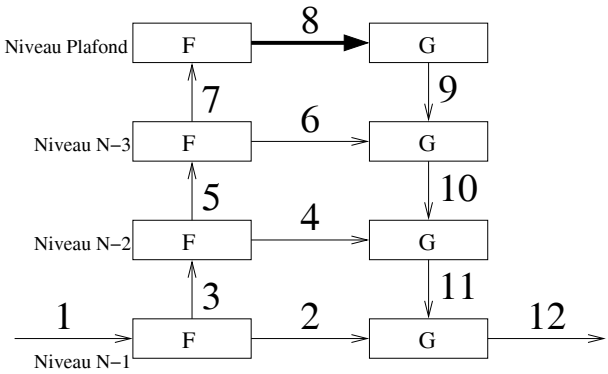


FIG. 2.13 – *Algorithme FMM économique*

première version simple, puis une seconde écriture plus élaborée.

2.5.3.1 Version mono-groupe

L'idée de base est de constater que les différentes sous-listes qui composent le calcul n'utilise jamais plus de deux « blocs » de fonctions de radiation. On appelle « bloc » l'ensemble des fonctions $\mathcal{F}$ ou $\mathcal{G}$ d'un niveau donné. Sur les figures 2.13 et 2.12, chaque rectangle représente un bloc de fonctions de radiation. Les phases d'initialisation et d'intégration utilisent un seul bloc chacune, les montées, descentes et transferts en utilisent deux.

La version mono-groupe de notre FMM out-of-core conserve par défaut sur disque toutes les fonctions de radiation. Avant chaque sous-liste, on charge en mémoire vive le bloc (ou les deux blocs) dont on va avoir besoin. Après avoir traité cette sous-liste, on sauvegarde sur disque les fonctions de radiation modifiées. Par conséquent, on a au plus deux blocs de fonctions de radiation en mémoire au cours du calcul. Le niveau le plus volumineux étant celui des feuilles, la consommation mémoire maximale sera atteinte au moment du transfert entre les fonctions $\mathcal{F}$ et $\mathcal{G}$ du plus bas niveau (i.e. la sous-liste 2). Dans le cas de la sphère à un million d'inconnues, ce maximum représente 310,6 Mo.

2.5.3.2 Version multi-groupe

La version multi-groupe est une extension de la précédente. L'idée est de diviser les « blocs » de fonctions de radiation en groupes, et de traiter les opérations non plus bloc par bloc, mais groupe par groupe. Par exemple, la figure 2.14 représente la façon de traiter un transfert entre un bloc de fonctions $\mathcal{F}$ divisé en trois groupes notés $\mathcal{F}_1$, $\mathcal{F}_2$ et $\mathcal{F}_3$, et un bloc de fonctions $\mathcal{G}$ divisé en trois groupes notés $\mathcal{G}_1$, $\mathcal{G}_2$ et $\mathcal{G}_3$. Chacun de ses groupes va correspondre à un tiers des cellules du niveau de l'octree considéré.

L'opération de transfert symbolisée sur cette figure est ainsi décomposée en 3×3 sous-étapes, chacune opérant sur deux groupes. Les groupes non utilisés sont conservés sur disque. La mémoire nécessaire à cette opération de transfert est donc divisée par 3.

L'application de ce type de division à l'algorithme tout entier tel qu'il est symbolisé sur la figure 2.13 se fait aisément en traitant chaque étape séparément. Afin d'avoir des groupes de même taille à tous les niveaux, on fera varier le nombre de groupes en fonction du niveau. Par exemple, dans le cas de la sphère à un million d'inconnues, avec des tailles de niveau reprises

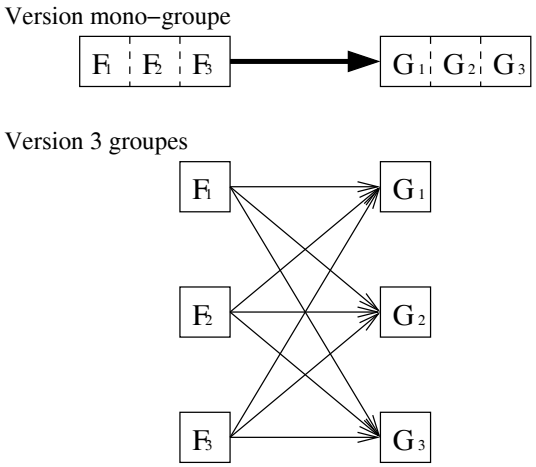


FIG. 2.14 – *Opération de transfert en mode multi-groupe*

dans le tableau 2.10, si on crée 4 groupes au niveau des feuilles, chaque groupe fera 38,8 Mo. Aux autres niveaux, on choisira le plus petit nombre de groupes assurant une taille de groupe inférieure à 38,8 Mo.

Niveaux	Taille totale (Mo)	Nombre de groupes	Taille d'un groupe (Mo)
2	17,6	1	17,6
3	23,0	1	23,0
4	32,2	1	32,2
5	44,3	2	22,1
6	71,0	2	35,5
7	155,3	4	38,8

TAB. 2.10 – *Nombre de groupes aux différents niveaux*

Ainsi, pour ce cas de calcul, les fonctions de radiation consommeront au plus $2 \times 38{,}8 = 77{,}6$ Mo en mémoire vive au cours des produits multipôles.

2.5.3.3 Implémentation

Le paramètre choisit pour gouverner le mode out-of-core est le nombre de groupes au niveau des feuilles noté $\mathcal{N}_{GR}$.

$\mathcal{N}_{GR} = 0$ correspond à un calcul in-core, tel qu'on le voit sur la figure 2.13. $\mathcal{N}_{GR} = 1$ correspond au mode out-of-core mono-groupe de la section 2.5.3.1. Enfin, $\mathcal{N}_{GR} \geq 2$ correspond au mode out-of-core multi-groupe de la section 2.5.3.2.

La rajout de ces améliorations à notre méthode multipôle se fait par la modification des sous-listes de tâches définies au paragraphe 2.1.3 : on va réordonner les tâches existantes, et rajouter de nouvelles tâches.

2.5.3.3.1 Nouvelles tâches élémentaires Dans la suite, la lettre $\mathcal{H}$ remplace soit $\mathcal{F}$, soit $\mathcal{G}$, et sert à désigner de manière générique une famille de fonctions de radiation. On

définit deux nouvelles tâches élémentaires pour charger et sauvegarder sur disque un groupe de fonctions de radiation à un niveau donné :

2.5.3.3.1.1 LireGroupe($\mathcal{H}$,l,n) :

Paramètres :	un type de fonction de radiation ($\mathcal{F}$ ou $\mathcal{G}$),
	un numéro de niveau l,
	un numéro de groupe n.
Action :	charge en mémoire le groupe de fonctions de radiation désigné.

2.5.3.3.1.2 EcrireGroupe($\mathcal{H}$,l,n) :

Paramètres :	un type de fonction de radiation ($\mathcal{F}$ ou $\mathcal{G}$),
	un numéro de niveau l,
	un numéro de groupe n.
Action :	écrit sur disque le groupe de fonctions de radiation désigné.
	libère la zone de mémoire correspondante

2.5.3.3.2 Transformation d'une sous-liste $\mathcal{L}_i$ Le travail est différent selon qu'il s'agit d'une sous-liste travaillant sur un seul bloc de fonctions de radiation (initialisation, intégration) ou sur deux blocs (montée, descente, transfert). On va traiter le cas d'une sous-liste travaillant sur deux blocs situés à des niveaux différents : soit $\mathcal{L}_i$ la sous-liste de montée du niveau l au niveau $l-1$ (créée à la section 2.1.3.2).

On note $\mathcal{N}_{GR}^{(l)}$ et $\mathcal{N}_{GR}^{(l-1)}$ les nombres de groupes aux deux niveaux considérés. La répartition des fonctions de radiation en groupes se fait en pratique en créant des groupes de cellules à chaque niveau de l'octree, donc on peut noter $gr(\mathcal{C})$ le numéro du groupe auquel appartient la cellule $\mathcal{C}$. La liste out-of-core $\mathcal{L}'_i$ est créée par l'algorithme 2.6.

En substance, on fait une double boucle sur les numéros de groupe aux deux niveaux, et on recopie de $\mathcal{L}_i$ à $\mathcal{L}'_i$ toutes les montées travaillant sur ces deux groupes. A chaque changement de groupe (au niveau l ou $l-1$), on rajoute les tâches de chargement ou de sauvegarde. Pour une liste de descente, ce serait le même chose avec des fonctions $\mathcal{G}$. Dans le cas d'un liste de transfert, on aurait un seul niveau, mais deux types de fonctions de radiation à considérer. Remarquons que si les transferts de la liste $\mathcal{L}_i$ sont ordonnés (cf. section 2.2.2.1), les transferts pour deux groupes donnés sont aussi ordonnés dans la liste modifiée $\mathcal{L}'_i$. Autrement dit, cette transformation ne nous fait pas perdre le bénéfice des transformations précédentes de nos sous-listes de tâches.

Pour une liste d'initialisation, il n'y a par nature pas de « LireGroupe », donc cela donne l'algorithme 2.7.

Pour une liste d'intégration, il n'y a pas de tâche « EcrireGroupe ».

Finissons cette section consacrée à l'implémentation du mode out-of-core par une remarque : plutôt que de demander à l'utilisateur le nombre de groupes au niveau des feuilles, il aurait été plus pratique de lui demander la quantité de mémoire vive qu'il souhaitait allouer aux fonctions de radiation. En effet, le physicien qui lance un calcul d'électromagnétisme ne connaît sans doute pas la taille des fonctions de radiation au niveau des feuilles. Néanmoins, le passage de l'un à l'autre de ces paramètres se fait sans difficulté, il est certain que nous réaliserons cette modification très bientôt.

$\mathcal{L}'_i \leftarrow \varnothing$
// Boucle sur les groupes du niveau l :
for $n^{(l)} = 1, \ldots, \mathcal{N}_{GR}^{(l)}$ **do**
  $\mathcal{L}'_i = \mathcal{L}'_i \cup$ LireGroupe($\mathcal{F}$, l, $n^{(l)}$)
  // Boucle sur les groupes du niveau $l-1$:
  for $n^{(l-1)} = 1, \ldots, \mathcal{N}_{GR}^{(l-1)}$ **do**
    $\mathcal{L}'_i = \mathcal{L}'_i \cup$ LireGroupe($\mathcal{F}$, $l-1$, $n^{(l-1)}$)
    // Examen de la sous-liste $\mathcal{L}_i$:
    for all montée($\mathcal{C}$) $\in \mathcal{L}_i$ **do**
      if $\left(gr(\mathcal{C}) = n^{(l)}\right)$ & $\left(gr(p(\mathcal{C})) = n^{(l-1)}\right)$ **then**
        $\mathcal{L}'_i = \mathcal{L}'_i \cup$ montée($\mathcal{C}$)
      end if
    end for
    $\mathcal{L}'_i = \mathcal{L}'_i \cup$ EcrireGroupe($\mathcal{F}$, $l-1$, $n^{(l-1)}$)
  end for
  $\mathcal{L}'_i = \mathcal{L}'_i \cup$ EcrireGroupe($\mathcal{F}$, l, $n^{(l)}$)
end for

Algorithme 2.6: Montée en mode out-of-core

$\mathcal{L}'_i \leftarrow \varnothing$
// Boucle sur les groupes du niveau l :
for $n^{(l)} = 1, \ldots, \mathcal{N}_{GR}^{(l)}$ **do**
  // Examen de la sous-liste $\mathcal{L}_i$:
  for all initialisation($\mathcal{C}$) $\in \mathcal{L}_i$ **do**
    if $\left(gr(\mathcal{C}) = n^{(l)}\right)$ **then**
      $\mathcal{L}'_i = \mathcal{L}'_i \cup$ initialisation($\mathcal{C}$)
    end if
  end for
  $\mathcal{L}'_i = \mathcal{L}'_i \cup$ EcrireGroupe($\mathcal{F}$, l, $n^{(l)}$)
end for

Algorithme 2.7: Initialisation en mode out-of-core

2.5.4 Représentation graphique

Nous reviendrons à la section 3.4.2 sur les gains obtenus en terme de mémoire, et l'impact sur les performances de ces modifications. Nous allons juste illustrer graphiquement l'efficacité et les limites de nos travaux.

On reprend encore le cas de notre sphère maillée avec 1.023.768 inconnues. Grâce à une bibliothèque de suivi des allocations et libérations de mémoire, on peut connaître précisément l'évolution de la quantité de mémoire vive utilisée par le code d'électromagnétisme au fil du calcul. L'utilisation de cette bibliothèque n'étant pas gratuite, les temps de calcul mesurés sont très supérieurs à ceux obtenus auparavant.

On réalise un calcul utilisant un solveur QMR volontairement arrêté après 7 produits multipôles. Dans un premier temps, on travaille en mode in-core. La figure 2.15 donne l'espace mémoire alloué (en méga-octets) en fonction du temps (en secondes). La partie de gauche (pour une date $t \in [0,1000]$) correspond au démarrage du calcul (relecture du maillage, création de l'octree, de la liste de tâches). Le motif qui se répète ensuite 7 fois correspond aux 7 produits multipôles. En dehors des itérations multipôles, le maximum est atteint à t=200 s et se situe à 270 Mo. Il correspond à la phase de création de la liste de tâches.

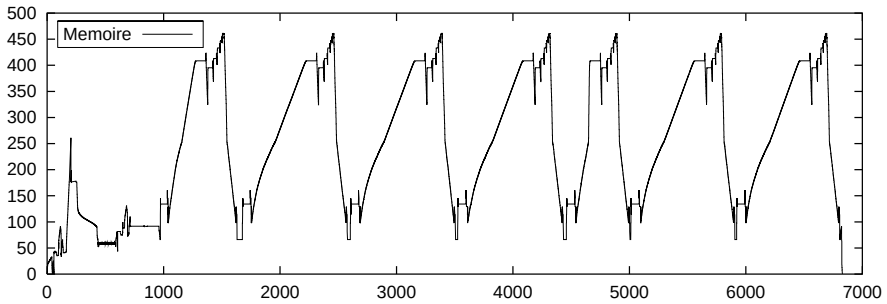


FIG. 2.15 – *Mémoire utilisée au fil d'un calcul in-core*

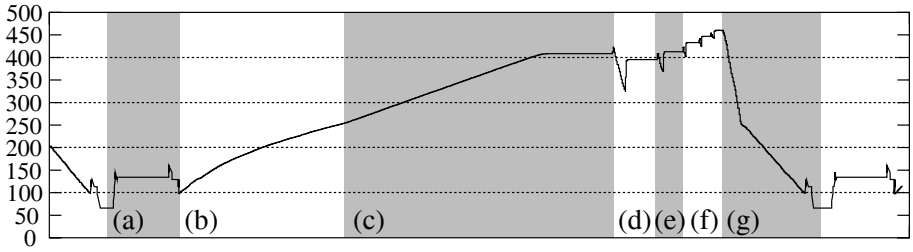


FIG. 2.16 – *Mémoire utilisée au cours d'un produit multipôle*

La figure 2.16 est un zoom sur un seul produit multipôle. Le profil de cette courbe s'interprète de la manière suivante (les lettres (a) à (g) renvoient aux bandes grisées du graphique 2.16) :

(a) Interactions proches : la zone (a) correspond au traitement des interactions proches, la matrice étant stockée sur disque, c'est une phase peu consommatrice de mémoire.

(b) Initialisation : ensuite vient l'initialisation des fonctions de radiation $\mathcal{F}$ des feuilles, qui représentent en tout 155,3 Mo (cf. tableau 2.9).

(c) Transfert au niveau 7 : le seconde partie de cette pente (de 250 à 400 Mo) et le petit plat qui la suit représentent la phase de transfert au niveau des feuilles (qui conduit à allouer toutes les fonctions de radiation $\mathcal{G}$ des feuilles soit encore 155,3 Mo).

(d) Montée et transfert au niveau 6 : la zone (d) correspond à la montée du niveau 7 au niveau 6 (on alloue les $\mathcal{F}$ au niveau 6 et on libère les $\mathcal{F}$ au niveau 7, d'où un gain de 80 Mo environ), puis aux transferts au niveau 6 (on alloue les $\mathcal{G}$ au niveau 6, d'où 71 Mo de hausse).

(e) Montée et transfert au niveau 5 : la zone (e) représente la même chose que la zone (d), mais un niveau plus haut. On retrouve le même profil en plus petit, car le traitement est plus rapide. On libère les fonctions $\mathcal{F}$ au niveau 6, et on alloue les fonctions $\mathcal{F}$ et $\mathcal{G}$ au niveau 5, d'où une hausse de $2 \times 44{,}3 - 71{,}0 = 17{,}6$.

(f) Montée et transfert jusqu'au niveau 2 : la motif des zones (d) et (e) se reproduit de plus en plus petit au fur et à mesure de la montée dans l'arbre. A l'arrivée au niveau 2, on atteint le maximum de consommation de mémoire.

(g) Descentes et intégrations : les phases successives de descentes se traduisent par la libération progressive des fonctions $\mathcal{G}$ des niveaux 2 à 6. L'intégration finale libère les fonctions $\mathcal{G}$ du niveau 7.

Cette analyse nous montre que :

- Le maximum est bien atteint après les transferts au niveau 2, il est de 460 Mo.
- Les fonctions de radiation sont bien les plus importantes consommatrices de mémoire durant le calcul, car sur ces 460 Mo occupés, 360 le sont par des fonctions de radiation, et 100 Mo sont des données autres.

La figure 2.17 montre le même type d'information (i.e. mémoire en fonction du temps) pour des calculs en mode out-of-core avec, de haut en bas, un puis deux puis trois groupes. Le travail sur des blocs de fonctions de radiation se traduit par l'aspect beaucoup plus rectangulaire de la courbe.

Le produit multipôle culmine à environ 410 Mo avec 1 groupe, 250 Mo avec 2 groupes et 200 Mo avec 3 groupes. Cela correspond exactement à nos prévisions théoriques (cf. tableau 2.11).

Nombre de groupes	Taille maximale d'un groupe (Mo)	Données autres	Maximum (=2 groupes+autres)
1	155,3	100	410
2	$155{,}3/2 = 77{,}6$	100	255
3	$155{,}3/3 = 51{,}7$	100	203
$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$155{,}3/n$	100	$2(155{,}3/n) + 100$

TAB. 2.11 – *Consommation d'un produit multipôle out-of-core*

On voit donc que notre méthode marche très bien puisqu'on parvient à faire baisser la mémoire consommée par le produit multipôle à volonté. Bien sûr, assez rapidement le maximum « absolu » n'est plus atteint pendant les itérations multipôles : c'est ce qu'on observe à partir de 2 groupes et au-delà, le maximum s'est déplacé et est désormais atteint au moment de la

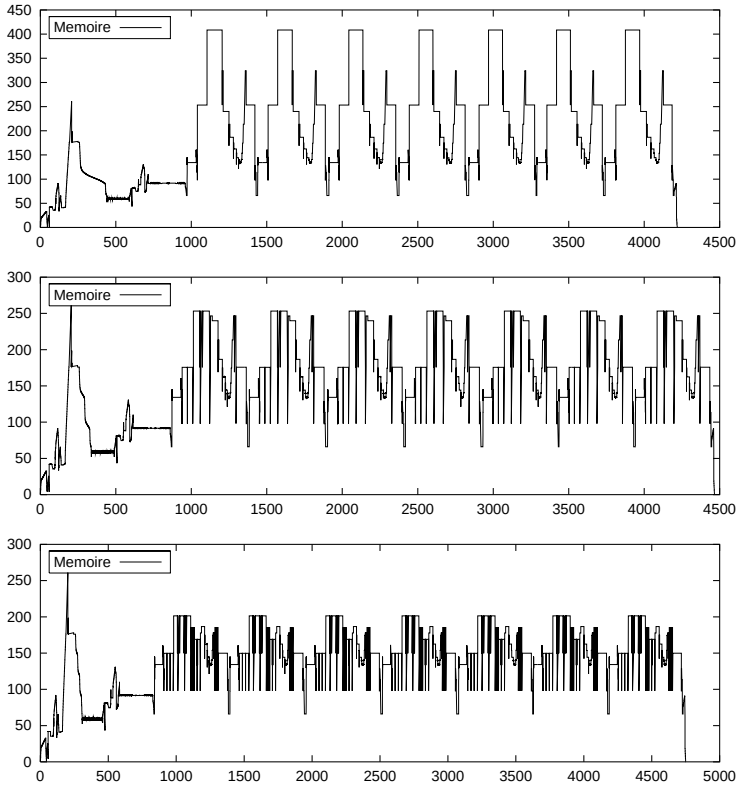


FIG. 2.17 – *Mémoire utilisée au fil d'un calcul out-of-core à 1, 2, et 3 groupes*

création de la liste de tâches. A moins de réécrire cette partie là aussi, il n'est donc pas utile d'utiliser davantage de groupes. Ceci dit, notre objectif est atteint puisque ce qui était notre facteur limitant (la mémoire consommée par les fonctions de radiation) ne nous limite plus (pour les tests de performances, se reporter à la section 3.4.2).

Conclusion

Nous avons présenté les techniques utilisées pour réaliser une implémentation séquentielle de la méthode multipôle rapide. L'utilisation d'une liste de tâches est véritablement la pierre angulaire de ce travail : elle nous a permis de réaliser des optimisations très fines du code de calcul (réordonnancement des transferts, fonctionnement en mode out-of-core), et sera a nouveau mise à contribution lors de la parallélisation de la FMM (décrite au chapitre 4. Nous avons ainsi mis au point un code à la fois flexible et optimisé dont il convient maintenant d'évaluer la réelle performance. C'est l'objet du chapitre à venir.

Chapitre 3

Performances et Applications

Sommaire

Introduction **121**
3.1 Présentation technique **121**
3.2 Présentation des cas tests **122**
3.3 Comparaison avec les méthodes classiques **125**
3.4 Choix des paramètres **128**
3.5 Comparaison par machine **145**
3.6 Scalabilité **147**
3.7 Applications industrielles **154**
Conclusion **161**

Introduction

Dans ce chapitre, nous allons présenter l'implémentation séquentielle de la méthode multipôle que nous avons réalisée dans le code AS_ELFIP de la société EADS. Ce logiciel résout les équations de Maxwell sous leur forme intégrale par des méthodes directes (factorisation $L.D.^tL$) ou itératives (solveur GMRES et TFQMR entre autres).

Après une courte introduction technique au logiciel, nous présentons les trois types d'objets que nous avons utilisé pour nos tests. Nous démontrerons ensuite l'intérêt de la méthode multipôle en la comparant aux méthodes usuelles sur des cas de calculs simples. La section suivante passera en revue tous les paramètres ajustables de notre FMM en tentant de déterminer à chaque fois les meilleurs compromis. Les valeurs ainsi obtenues seront utilisées pour comparer les différentes machines utilisées, puis pour analyser la scalabilité numérique de notre code. Nous terminerons ce chapitre par une présentation de quelques cas de calcul industriels qui démontreront la puissance de cette méthode, même séquentielle.

3.1 Présentation technique

3.1.1 Langages

Le langage utilisé est principalement le C. Cela résulte à la fois d'un choix personnel et d'une contrainte imposée par le partenaire industriel (EADS). La partie « gestion et algo-

rithme » du code est donc rédigée dans ce langage, même si a posteriori il semblerait qu'un langage orienté objet comme le C++ eût été préférable.

Les parties de « calcul pur » du logiciel ont été écrite en FORTRAN 77, pour des raisons de vitesse et de simplicité (en particulier, il n'y a pas de type complexe en C).

3.1.2 Bibliothèques

Afin d'être automatiquement optimisé sur toutes les architectures, on a tenté d'utiliser un maximum de librairies standards. On a utilisé les bibliothèques suivantes :

BLAS, ATLAS, ESSL : sert pour toutes les opérations d'algèbre linéaire de base.

LAPACK : utilisé très ponctuellement (quadrature de Gauss par exemple).

QMRPACK : nous a fourni certains des solveurs itératifs utilisés.

FFTW : pour les transformées de Fourier dans les opérations de montée/descente.

NUMERICAL RECIPES : pour les calculs de fonctions spéciales (Hankel, Bessel) et les polynômes de Legendre.

3.1.3 Plates-formes

Le logiciel a été utilisé avec succès sur un grand nombre d'architectures différentes : x86 (compilateurs Gnu et Portland), Sun solaris, SGI (octane, origin2000, origin3000), IBM (SP2, SP3), DEC alpha, Cray T3E, NEC SX5 (non vectorisé).

3.2 Présentation des cas tests

3.2.1 Sphère

Les sphères utilisées ici sont créées directement par nos soins. Le maillage est homogène, puisque les arêtes font toutes la même taille ±20%. Toutes les sphères ont un rayon égal à 1 mètre, on fait varier la longueur d'onde et la finesse du maillage.

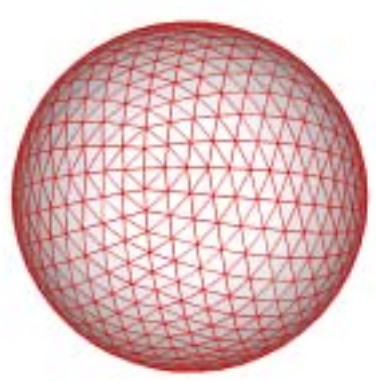


FIG. 3.1 – *Sphère à 2028 inconnues*

L'avantage d'utiliser un objet aussi simple qu'une sphère pour faire nos tests est que l'on connaît la solution exacte du problème. En particulier les séries de Mie [6] nous donnent directement le champ lointain diffracté. On note h_n les fonctions de Hankel sphériques du premier

type, on note j_n leurs parties réelles, et P_n^l les fonctions de Legendre (pour les expressions de ces fonctions, voir [1]). Dans le cas d'une sphère parfaitement conductrice de rayon r illuminée par une onde plane arrivant de la direction ($\theta = 0$,$\phi = 0$) et polarisée verticalement, la surface équivalent radar normalisée émise dans la direction (θ,ϕ) est donnée par $\sigma(\theta,\phi)$:

$$\begin{cases} \psi_n(x) = x j_n(x), \\ \zeta_n(x) = x h_n(x), \\ an = \dfrac{\psi_n(kr)}{\zeta_n(kr)}, \\ bn = \dfrac{\psi'_n(kr)}{\zeta'_n(kr)}, \\ S_1(\theta) = -i \displaystyle\sum_{n=1}^{\infty} (-1)^n \frac{2n+1}{n(n+1)} \left[b_n \frac{\partial P_n^1(\cos\theta)}{\partial\theta} - a_n \frac{P_n^1(\cos\theta)}{\sin\theta} \right], \\ S_2(\theta) = i \displaystyle\sum_{n=1}^{\infty} (-1)^n \frac{2n+1}{n(n+1)} \left[-a_n \frac{\partial P_n^1(\cos\theta)}{\partial\theta} + b_n \frac{P_n^1(\cos\theta)}{\sin\theta} \right], \\ \sigma(\theta,\phi) = \dfrac{4\pi}{k^2} \left[|S_1(\theta)|^2 \cos^2\phi + |S_2(\theta)|^2 \sin^2\phi \right] \end{cases}$$

Autre avantage, les matrices sont bien mieux conditionnées qu'avec des objets plus réalistes, et les solveurs itératifs convergent plus vite. Dernier avantage, si l'on souhaite tester la FMM sur des cas de grande taille, la forme de l'objet ne compte pas, donc une sphère fait très bien l'affaire. Inconvénient majeur : ce type de calcul n'est pas représentatif de ce que les utilisateurs potentiels de cette méthode attendent.

3.2.2 Cetaf

La maquette Cetaf est un cas de calcul standard pour lequel il n'existe pas de solution analytique. Le maillage est représenté sur la figure 3.2. L'objet a une forme d'aile, et comporte une fente. Ses dimensions sont 50 $cm \times 30$ $cm \times 5$ cm.

FIG. 3.2 – *Cetaf à 5391 inconnues*

Le maillage comporte 5391 inconnues et 3594 triangles. Il est moyennement homogène, puisque les arêtes minimales, moyennes, et maximales mesurent respectivement 0,23 cm, 1,2 cm et 1,8 cm. On prendra toujours une longueur d'onde égale à 7,9 fois l'arête moyenne, soit seulement 5,5 fois l'arête la plus longue (ce qui reste toutefois correct).

3.2.3 Avion

Enfin, nous avons à notre disposition un maillage d'avion. Il s'agit d'un Airbus A318 dont les dimensions (un peu irréelles) sont environ 1,8 $m \times$ 1,9 $m \times$ 0,65 m. Il y a un facteur d'homothétie de 18 par rapport à la réalité. Le maillage est représenté sur la figure 3.3.

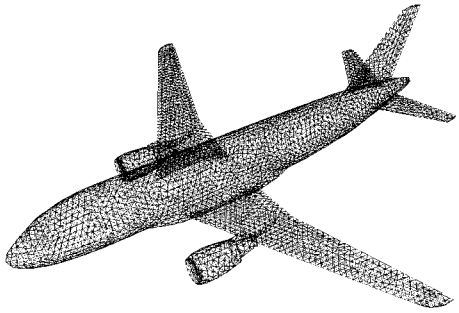


FIG. 3.3 – *Airbus A318 à 23676 inconnues*

Le maillage comporte 23676 inconnues et 15784 triangles. Il est fortement inhomogène, puisque les arêtes minimum/moyenne/maximum mesurent 0,6 mm, 1,7 cm et 7,4 cm. Si on choisit une longueur d'onde égale à 7,5 fois l'arête moyenne, l'arête la plus grande mesurera $\lambda/1{,}7$ ce qui peut être une source d'imprécision.

3.2.4 Raffinement de maillages

Les maillages de sphères étant générés automatiquement, on peut facilement faire varier le nombre d'inconnues. En revanche, les maillages du cetaf et de l'A318 nous ont été donnés par notre partenaire industriel, on ne peut donc pas les faire évoluer facilement. C'est pourquoi nous avons créé un outil de raffinement de maillage qui, partant d'un maillage existant et d'un paramètre entier n, subdivise chaque arête en n morceaux égaux, puis chaque triangle en n^2 petits triangles identiques (à une symétrie près, cf. figure 3.4), avant de sauvegarder ce nouveau maillage sous un format exploitable. Ainsi, en prenant $n = 2,3,4,\ldots$ on obtient des Cetaf à 21564, 48519, 86256 inconnues, et des A318 à 94704, 213084, 378816, ... inconnues.

L'inconvénient de cette technique simple est que le maillage raffiné ne suit plus la géométrie de l'objet. Cela n'est pas gênant dans le cas du cetaf où toutes les surfaces sont planes, c'est en revanche plus embêtant pour l'Airbus A318, pour lequel le maillage raffiné comportera des facettes. Techniquement, pour la FMM cela ne change rien, mais il faut garder à l'esprit que les résultats physiques obtenus risquent d'être déformés.

Par exemple, on a tracé sur la figure 3.5 les SER obtenues avec :

- D'une part une sphère maillée avec 800 triangles puis raffinée avec $n = 10$ pour obtenir 80.000 triangles. C'est donc une sphère « anguleuse ».
- D'autre part une sphère maillée directement avec 80.000 triangles.

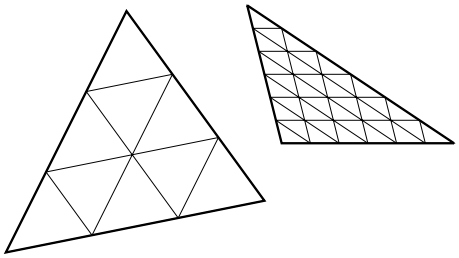


FIG. 3.4 – *Découpage d'un triangle dans les cas $n = 3$ et $n = 6$*

Les maillages sont tous conçus avec 10 points par longueurs d'onde (en moyenne), la sphère anguleuse a donc des facettes dont l'arête moyenne est environ une longueur d'onde. On voit que sur la partie gauche du graphique, la SER de la sphère anguleuse diffère légèrement, alors que les deux courbes coïncident sur la partie droite. Si on avait tracé en sus la solution exacte issue des séries de Mie, on aurait constaté que la courbe de la sphère originale coïncide exactement avec cette dernière. La déformation des résultats issue du raffinement est ici limitée, mais c'est un problème qui ira en augmentant avec la valeur de n. Il est donc préférable d'en être conscient avant de faire des calculs avec des maillages raffinés.

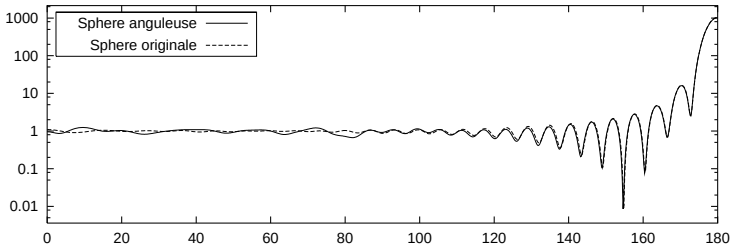


FIG. 3.5 – *SER comparée entre une sphère lisse et une sphère anguleuse*

3.3 Comparaison avec les méthodes classiques

3.3.1 Objectif

Avant de tester en détail la FMM, nous allons dans cette section comparer les différents modes de résolution du système linéaire considéré, à savoir : solveur direct, solveur itératif classique et solveur multipôle (terme désignant un solveur itératif utilisant un produit matrice-vecteur multipôle). Le solveur direct testé réalise une factorisation $L.U$ de la matrice du problème. Le solveur itératif choisi est un GMRES [10] avec résidu normalisé 10^{-2}. On utilise la formulation CFIE. Cette dernière permet aux solveurs itératifs de converger en moins d'itérations, mais pénalise les solveurs directs par sa matrice non-symétrique (les temps de calcul sont multipliés par au moins 2 par rapport à la formulation EFIE).

On va utiliser les trois maillages suivants :

- Sphère à 1452 inconnues ;
- Cetaf à 5391 inconnues ;

– Airbus à 23676 inconnues.

A chaque fois, on a utilisé un seul second membre.

3.3.2 Temps d'exécution

Pour chacun de ces trois cas de calcul, on donne dans le tableau 3.1 le temps d'initialisation et le temps de résolution sur un Pentium III 866 MHz. On note n_{dl} le nombre d'inconnues du problème, N_{iter} le nombre d'itérations du solveur itératif, et N_{rhs} le nombre de seconds membres. On donne dans la dernière colonne la croissance asymptotique théorique des temps d'initialisation et de résolution pour chacun des trois solveurs. Pour le solveur direct, l'initialisation comprend l'assemblage et la factorisation de la matrice, la résolution revient à faire un produit par la matrice factorisée. Pour le solveur itératif classique, l'initialisation consiste à assembler la matrice, la résolution contient N_{iter} produits par cette matrice. Enfin, pour le solveur itératif avec multipôle, l'initialisation consiste à assembler l'arbre et la matrice des interactions proches, la résolution contient N_{iter} produits multipôles.

	Sphère1452	Cetaf5391	Airbus23676	Croissance
Solveur direct :				
initialisation	150,7 s	4867 s	373380 s	${n_{dl}}^3$
résolution	0,4 s	4,2 s	380 s	$N_{rhs}{n_{dl}}^2$
Solveur itératif :				
initialisation	46 s	947 s	13150 s	${n_{dl}}^2$
résolution	3,6 s	259 s	31650 s	$N_{rhs}.N_{iter}.{n_{dl}}^2$
Solveur multipôle :				
initialisation	16,6 s	57 s	272 s	n_{dl}
résolution	3,3s	95 s	833 s	$N_{rhs}.N_{iter}.n_{dl}\log n_{dl}$

TAB. 3.1 – *Temps d'exécution pour différents types de solveurs*

Il est clair que dès que le nombre d'inconnues dépasse 1000, le solveur itératif multipôle est sans conteste le plus rapide. Son principal inconvénient est de dépendre du nombre d'itérations N_{iter}. Ce dernier n'est a priori pas majoré (si ce n'est par n_{dl} en arithmétique exacte), mais en choisissant un bon solveur, un bon préconditionneur et une bonne formulation, on peut en général le rendre inférieur à 100.

3.3.3 Stockage

Le stockage sur disque comporte dans tous les cas le maillage et la matrice. Dans le cas multipôle, il s'agit de la matrice des interactions proches, à laquelle il faut ajouter le stockage des fonctions de transfert. Les données numériques se trouvent dans le tableau 3.2.

	Sphère1452	Cetaf5391	Airbus23676	Croissance
Solveur direct	41 Mb	467 Mb	9067 Mb	${n_{dl}}^2$
Solveur itératif	43 Mb	474 Mb	9094 Mb	${n_{dl}}^2$
Solveur multipôle	7 Mb	28 Mb	137 Mb	$n_{dl}\log n_{dl}$

TAB. 3.2 – *Stockage pour différents types de solveurs*

Les deux solveurs non-multipôles nécessitent de calculer et stocker la matrice pleine de taille $n_{dl} \times n_{dl}$ (divisé par deux si une formulation symétrique type EFIE est utilisée). Au contraire, seules les interactions proches sont stockées dans le cas du solveur multipôle. Par ailleurs, le solveur itératif GMRES (qu'il soit multipôle ou non) a besoin de sauvegarder un certain nombre de vecteurs (dépendant du paramètre de *restart* choisi). C'est ce qui explique l'écart entre solveur direct et solveur itératif.

3.3.4 Précision

Dans chacun des trois cas de calcul, on compare la solution obtenue avec les deux solveurs itératifs à la solution issue du solveur direct (considérée comme référence). Les écarts relatifs figurent dans le tableau 3.3.

	Sphère1452	Cetaf5391	Airbus23676
Solveur itératif	1,7 %	18,7 %	6,6 %
Solveur multipôle	1,7 %	18,3 %	7,8 %

TAB. 3.3 – *Précision pour différents types de solveurs*

Les solveurs itératifs GMRES avaient comme condition d'arrêt un résidu normalisé de 10^{-2}. Sur un cas simple comme une sphère, la matrice est bien conditionnée, et l'écart relatif entre la solution exacte et la solution obtenue avec les solveurs itératifs (multipôle on non) est inférieur à deux fois le résidu demandé.

Sur le cetaf, l'écart sur les courants est nettement plus important. Il faut toutefois souligner trois points : tout d'abord, la méthode multipôle n'est pas en cause car le solveur itératif classique donne le même résultat. Ensuite, en demandant un résidu normalisé de 5.10^{-3}, cet écart baisse jusqu'à 9%, on voit que la convergence peut être améliorée. Elle le sera davantage si on utilise un préconditionneur. Enfin, si on trace le champ lointain diffracté pour chacune des trois solutions, on ne voit aucune différence (figure 3.6). Cela signifie que cet écart de 20% n'est pas gênant pour un calcul de champ lointain.

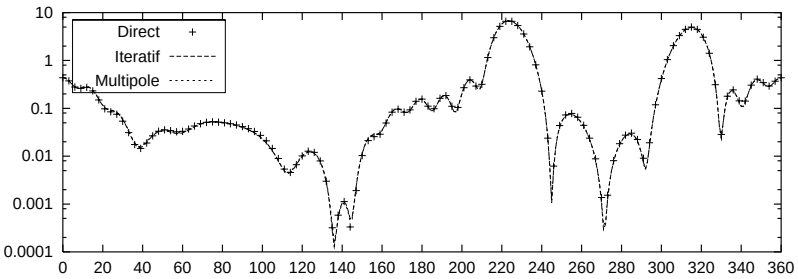


FIG. 3.6 – *Champ lointain du Cetaf pour nos trois solveurs*

Enfin, pour le cas test Airbus23676, les résultats sont à mi-chemin entre la sphère et le cetaf en terme de qualité. Sur la figure 3.7, on voit que globalement le résultat est très satisfaisant, même si on voit quelques écarts ici et là. Mais comme pour le cetaf, la faute en revient plutôt au solveur itératif, car les deux courbes correspondants aux solveurs itératifs (classique et multipôle) sont quasiment confondues. D'ailleurs, en diminuant le résidu normalisé de 10^{-2} à 10^{-4}, les différences en question s'estompent totalement.

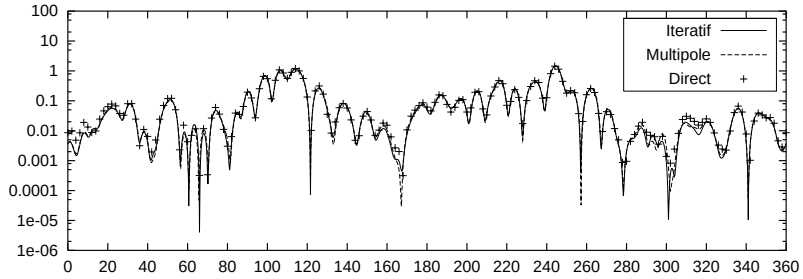


FIG. 3.7 – *Champ lointain de l'Airbus à 23676 inconnues pour nos trois solveurs*

3.3.5 Synthèse

On voit que le solveur multipôle répond a un véritable besoin, car les solveurs classiques atteignent très vite leurs limites, et il y répond bien : les besoins en temps et en mémoire restent très mesurés, et la précision des calculs est équivalente à celle d'un solveur itératif classique.

3.4 Choix des paramètres

Dans cette section, on va s'intéresser à étudier individuellement chacun des paramètres ajustables de la méthode multipôle. Certains de ces paramètres sont liés à la méthode elle-même (nombre de pôles, raffinement), d'autres proviennent d'améliorations spécifiques à notre implémentation (nombre de groupes, précision machine). A chaque fois, les paramètres non-étudiés sont fixés à leurs valeurs par défaut données dans le tableau 3.4. Il est important de noter que les résultats présentés dans deux sections différentes ne peuvent pas toujours être comparés, car ils ont pu être obtenus sur des machines ou avec des versions du logiciel différentes.

Paramètre	Valeur
Nombre de pôles	$C_\varepsilon = 7$
Nombre de groupes	0 (in-core)
Finesse de maillage	10 (pour les sphères)
	7,9 (pour les cetafs)
	7,5 (pour les Airbus)
Formulation	EFIE
Points de Gauss	1 par triangle
Taille de feuille	$\lambda/4$
Niveau Plafond	2
Précision machine	simple
Nombre de composantes	2
Stockage transfert	oui
Stockage translation	oui

TAB. 3.4 – *Valeurs par défaut des paramètres de notre FMM*

3.4.1 Nombre de pôles

3.4.1.1 Introduction

Le nombre de pôles est sans doute le paramètre à la fois le plus important et le plus difficile à déterminer. En fait, il faudrait plutôt dire les nombres de pôles, car il y en a un différent à chaque niveau de l'arbre. Idéalement, on cherche une formule qui donne le nombre de pôles L en fonction de la marge d'erreur tolérée ε et de l'arête des cellules d au niveau considéré (à ne pas confondre avec le paramètre a des sections 1.2.2.4 et 1.3.2.3 qui était l'arête des cellules *en nombre de longueurs d'onde*).

3.4.1.2 Formule donnant le nombre de pôles

En pratique, la dépendance de L en ε est difficile à intégrer dans une formule. Pour notre implémentation, on a repris une expression proposée dans [15], à savoir :

$$L(d) = \sqrt{3}dk + C_\varepsilon . \log_{10}(\sqrt{3}dk + \pi) \tag{3.1}$$

Avec les notations :

- $\sqrt{3}d$ correspond au diamètre des cellules cubiques d'arête d.
- $k = 2\pi/\lambda$ est le nombre d'onde.
- C_ε est un paramètre dépendant de ε.

En fait, c'est plutôt ε qui va dépendre de C_ε. On va en effet tester toutes les valeurs de C_ε comprises entre 0 et 20 par pas de 0,5 et voir quelle est la précision ainsi obtenue. L'approche n'est certes pas très noble, mais elle donne de bons résultats.

Remarquons que dans la formule (3.1), pour les petites boîtes, le terme en logarithme est prépondérant, tandis qu'il devient négligeable pour les grandes boîtes. Le tableau 3.5 montre les nombres de pôles obtenus en pratique pour différentes valeurs de C_ε et pour différentes tailles de boîtes (données en nombre de longueurs d'onde). Ces boîtes correspondent à celles utilisées pour l'Airbus213084. Lorsque C_ε passe de 2 à 8, le nombre de pôles pour les feuilles est multiplié par 2, tandis qu'il augmente d'à peine 4% au niveau le plus haut.

Arête (en λ)	$C_\varepsilon = 2$	$C_\varepsilon = 5$	$C_\varepsilon = 8$
1/4	4	6	8
1/2	7	10	12
1	13	16	20
2	24	28	32
4	46	51	56
8	90	96	102
16	178	185	192

TAB. 3.5 – *Nombre de pôles $L(d)$ en fonction de C_ε et d*

3.4.1.3 Etude de la précision

Les résultats obtenus sont présentés graphiquement, et le tableau 3.7 reprend une partie des résultats numériques. La machine utilisée ici est une IBM SP3 équipée de processeurs

Power3+/375 Mhz. On réalise ces tests sur trois objets : une sphère à 255792 inconnues, un Airbus à 213084 inconnues, et un cetaf à 539100 inconnues.

La figure 3.8 montre la précision du produit multipôle en fonction de C_ε. Ici et dans toute la suite, cette précision est l'écart relatif entre certaines colonnes de la matrice du système et les colonnes correspondantes de la « matrice multipôle » (obtenues en faisant un produit multipôle par un vecteur dont toutes les composantes sont nulles sauf une qui vaut 1). Ces colonnes de tests sont choisies réparties sur l'objet, et sont au nombre de 10.

Pour des valeurs de C_ε petites (inférieures à 2), il n'y a pas suffisamment de pôles aux plus bas niveaux de l'arbre pour rendre compte des phénomènes électromagnétiques. Par conséquent la précision est médiocre. Pour C_ε grand (supérieur à 15), il y a au contraire trop de pôles, et les fonctions de transfert divergent (cf. section 1.2.2.2). Dans ce cas, le calcul devient totalement faux.

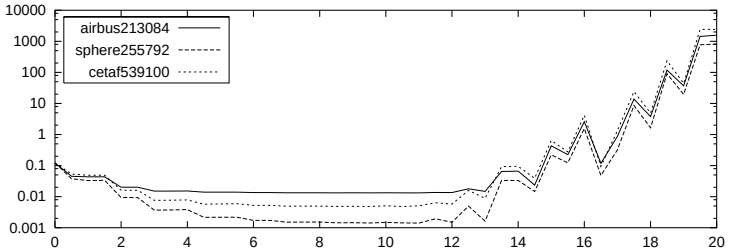


FIG. 3.8 – *Précision en fonction de C_ε*

Entre les deux, la précision stagne à un niveau minimum qui dépend de l'objet : environ $1{,}4.10^{-3}$ pour la sphère, $1{,}3.10^{-2}$ pour l'Airbus, $4{,}8.10^{-3}$ pour le cetaf. Il y a donc un écart de presque 10 entre les deux valeurs extrêmes, on peut avancer l'explication suivante : les trois maillages testés se distinguent par la finesse de leur maillage. La plus grande arête mesure $\lambda/1{,}7$ pour l'Airbus, $\lambda/5{,}4$ pour le cetaf et $\lambda/8{,}1$ pour la sphère. On voit que la FMM est d'autant plus précise que l'arête maximale est petite. Il est certain que le fait d'avoir des triangles « surdimensionnés » par rapport aux feuilles de l'octree est une source d'erreur. La qualité du maillage est donc importante pour la précision de la méthode. Nous reviendrons sur ce point à la section 3.4.3. Retenons qu'on choisira C_ε dans l'intervalle [2,8] puisqu'en deçà la précision est insuffisante, et au-delà elle stagne puis se détériore.

3.4.1.4 Etude du temps d'exécution

La figure 3.9 montre le temps en secondes nécessaire à la réalisation d'un produit multipôle en fonction de la valeur de C_ε. La croissance est régulière et quasi-linéaire dans l'intervalle $C_\varepsilon \in [2,8]$.

La figure 3.10 combine les résultats précédents pour tracer la précision multipôle en fonction du temps d'exécution. La partie droite du graphique, très chaotique, correspond aux nombres de pôles trop grands qui font diverger les fonctions de transfert. La partie de gauche correspond aux nombres de pôles admissibles.

La figure 3.11 est une restriction de la figure 3.10 aux valeurs de C_ε comprises entre 2 et 8 et espacées de 0,5. Les petits carrés correspondent aux points effectivement obtenus. On constate que ces points ne sont pas équirépartis sur la courbe, mais qu'au contraire ils ont

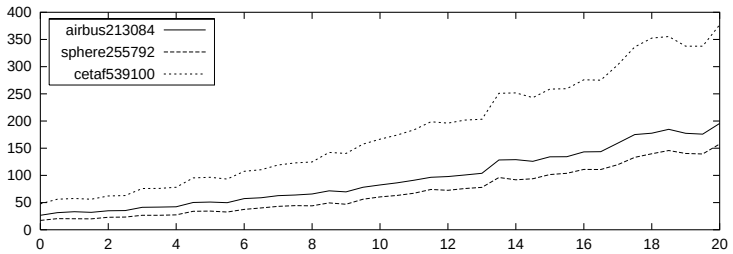


FIG. 3.9 – *Temps en fonction de C_ε sur SP3*

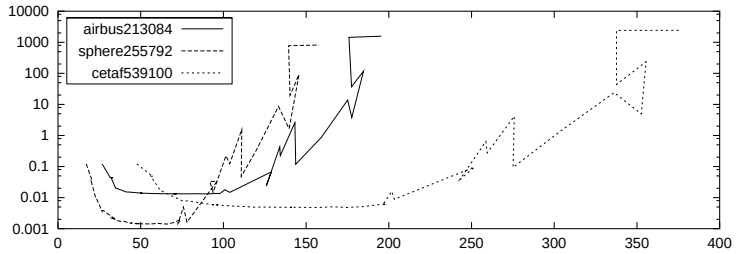


FIG. 3.10 – *Précision en fonction du temps*

tendance à se regrouper pour former des petits paquets. Sur chacune des trois courbes, on trouve 5 paquets regroupant, de la gauche vers la droite, 2, 3, 3, 2, et 3 points. L'explication se trouve dans le tableau 3.6 : on y donne le nombre de pôles utilisés pour les feuilles en fonction de C_ε.

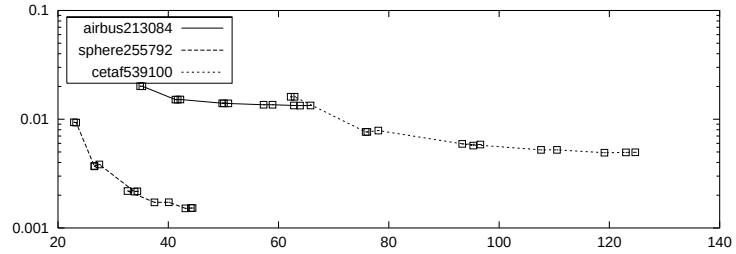


FIG. 3.11 – *Précision en fonction du temps pour $C_\varepsilon \in [2,8]$*

C_ε	Nombre de pôles
2 à 2,5	4
3 à 4	5
4,5 à 5,5	6
6 à 6,5	7
7 à 8	8

TAB. 3.6 – *Nombre de pôles des feuilles en fonction de C_ε*

Les points de la courbe 3.11 se regroupent en fonction du nombre de pôles au plus bas niveau. Par conséquent, on en déduit que le nombre de pôles des feuilles est un facteur primordial à la fois pour la précision du calcul multipôle et pour sa rapidité d'exécution.

3.4.1.5 Synthèse des résultats

Dans l'optique de réaliser un code multipôle capable d'ajuster sa précision, on a choisi de retenir trois valeurs de C_ε donnant respectivement :

- la meilleur précision possible ;
- la plus grande rapidité possible en gardant (si possible) une erreur inférieure à 1 % ;
- un compromis vitesse/précision intermédiaire.

Les valeurs retenues sont données dans le tableau 3.7.

C_ε	Sphère 255792		Cetaf 539100		Airbus 213084	
	écart	temps	écart	temps	écart	temps
2	$9{,}37.10^{-3}$	22,9	$1{,}61.10^{-2}$	62,2	$2{,}01.10^{-2}$	34,9
4	$3{,}83.10^{-3}$	27,5	$7{,}86.10^{-3}$	78,1	$1{,}52.10^{-2}$	42,2
7	$1{,}52.10^{-3}$	43,1	$4{,}92.10^{-3}$	119,1	$1{,}34.10^{-2}$	62,8

TAB. 3.7 – *Valeurs de C_ε choisies*

3.4.2 Nombre de groupes

3.4.2.1 Objectif

On s'intéresse ici aux modes de fonctionnement in-core et out-of-core décrits à la section 2.5. Nous allons illustrer à la fois à travers quelques exemples l'économie réalisable en terme de consommation mémoire, et son impact sur le temps d'exécution d'un produit matrice-vecteur multipôle. On ne parlera pas ici de précision puisque la version out-of-core ne diffère de la version in-core que par l'ordre des opérations, et produit donc un résultat identique (aux erreurs d'arrondis près).

3.4.2.2 Tests

Pour nos tests, on a utilisé un maillage de cetaf à 2.156.400 inconnues sur une machine IBM SP3 (en mode séquentiel bien sûr).

Pour un nombre croissant de groupes, on indique dans le tableau 3.8 la mémoire vive maximale utilisée par le programme au cours d'un calcul standard complet, le temps nécessaire à la réalisation d'un seul produit matrice-vecteur, et enfin la part de ce temps consacrée à la lecture et à l'écriture des groupes de fonctions de radiation. Notons que les temps donnés ici ne sont pas optimaux, car ils sont détériorés par la surveillance des allocations de mémoire.

3.4.2.3 Interprétation

On constate que le gain en mémoire est notable jusqu'à 3 groupes, puis inexistant. Cela signifie simplement que le maximum n'est plus atteint au cours des produits multipôles. Pour le cas à 3 groupes, on consacre 13,5 % du temps d'une itération multipôle au fonctionnement out-of-core, ce qui reste raisonnable compte tenu du gain obtenu sur la mémoire.

Nombre de groupes	Mémoire vive maximale (Mo)	Temps (s)	% I/O
0	1369	579,8	0
1	1070	590,6	5
2	649	636,9	11,9
3	545	658,6	13,5
4	545	702,3	16,7
5	545	729,9	19,5
6	545	748,0	21,1
7	545	778,9	23

TAB. 3.8 – *FMM out-of-core pour le cetaf à 2.156.400 inconnues*

Il faut noter que l'impact sur les performances sera d'autant plus grand que le système de fichiers sera peu performant. Par exemple, pour un calcul sur un Airbus à 1,1 million d'inconnues en mode out-of-core à 5 groupes, un PC équipé d'un disque IDE (entrée de gamme) consacrera 376 secondes par itération aux lectures et écritures du mode out-of-core, contre seulement 200 secondes pour un PC équivalent équipé d'un disque SCSI (haut de gamme). Le temps de calcul ici est de 630 secondes par itération.

Enfin, il est bien évident que le mode out-of-core a aussi un impact sur l'espace disque utilisé. Pour cet Airbus à 1,1 million de degrés de liberté, on utilise 4,26 Go en mode in-core, auxquels il faut rajouter 1,39 Go en mode out-of-core, soit environ 30 % d'augmentation.

3.4.3 Raffinement du maillage

3.4.3.1 Objectif

On va chercher à cerner l'influence de la finesse du maillage sur la précision du produit multipôle, et sur le temps nécessaire à la réalisation d'une itération multipôle. Cette section est le pendant appliqué de l'étude théorique de la section 1.2.2.4.4.2.

On appelle finesse de maillage, et l'on note d, le nombre moyen de points par longueur d'onde. Cela signifie que la longueur *moyenne* des arêtes du maillage est λ/d. Le nombre d'inconnues du problème varie comme d^2. Si on utilise un solveur direct, le temps de résolution croît donc comme d^6, et comme d^4 avec un solveur itératif « classique » (i.e. non-multipôle). Autrement dit, une augmentation de 10 % de la valeur de d se traduit par une augmentation de 77 % (resp. 46 %) du temps de résolution. Par conséquent, on peut obtenir des accélérations très importantes simplement en faisant varier la finesse du maillage. Nous allons voir s'il en est de même avec un solveur itératif multipôle.

3.4.3.2 Expériences numériques

Pour mener cette étude, nous allons utiliser une série de maillages sphériques de même rayon $r = 4\lambda$, mais de finesse de maillage variable. Dans les maillages de sphères utilisés, les longueurs moyennes, minimales et maximales des arêtes sont λ/d, $(\lambda/d)/1{,}23$ et $(\lambda/d) \times 1{,}23$ La sphère n'est certes pas un objet très représentatif, mais l'avantage est qu'on peut en créer facilement des maillages, et avoir un contrôle total sur leurs paramètres. Les figures 3.12 et 3.13 représentent la précision multipôle et le temps d'exécution de la FMM en fonction de d. Le tableau 3.9 reprend numériquement certaines de ces valeurs.

d	Taille d'arête			Nombre	Précision	Temps
	minimum	moyenne	maximum	d'inconnues		d'exécution (s)
3	$\lambda/3{,}7$	$\lambda/3$	$\lambda/2{,}4$	6348	$1{,}07.10^{-1}$	12,7
6	$\lambda/7{,}4$	$\lambda/6$	$\lambda/4{,}9$	25932	$9{,}20.10^{-3}$	17,4
10	$\lambda/12{,}3$	$\lambda/10$	$\lambda/8{,}1$	71148	$1{,}30.10^{-3}$	21,7
13	$\lambda/16$	$\lambda/13$	$\lambda/10{,}5$	122412	$5{,}15.10^{-4}$	26,6
17	$\lambda/21$	$\lambda/17$	$\lambda/13{,}8$	209088	$2{,}83.10^{-4}$	74,0
20	$\lambda/24{,}7$	$\lambda/20$	$\lambda/16{,}2$	288300	$1{,}62.10^{-4}$	105,6

TAB. 3.9 – *Précision de la FMM en fonction de la finesse du maillage*

3.4.3.3 Etude de la précision

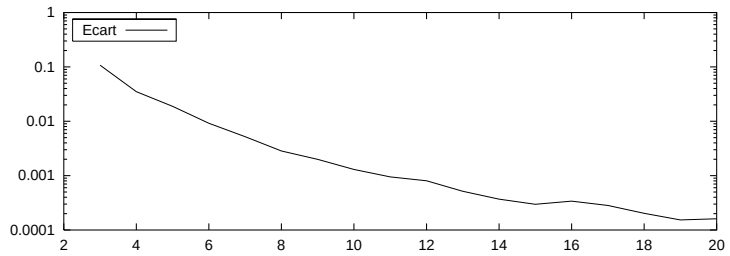


FIG. 3.12 – *Précision en fonction de la finesse du maillage d*

Regardons tout d'abord la figure 3.12. Comme prévu, la précision s'améliore avec la finesse du maillage. Les triangles étant plus petits, il y a moins d'interactions proches traitées en mode multipôle à cause de problèmes de débordement (cf. figure 1.12). A $d = 6$, on passe en dessous de 1 % d'erreur, ce qui est satisfaisant. La courbe s'aplanit ensuite progressivement, et le gain en précision obtenu en prenant des valeurs de d plus grandes est de plus en plus faible.

3.4.3.4 Etude du temps d'exécution

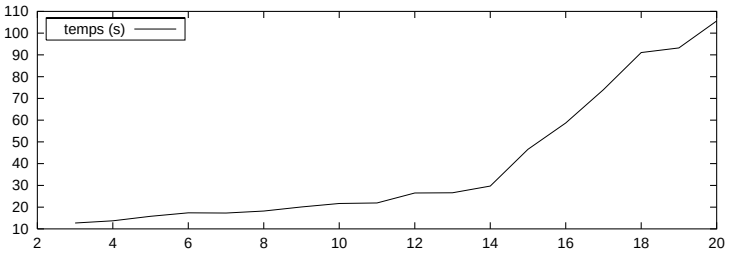


FIG. 3.13 – *Rapidité de la FMM en fonction de la finesse du maillage d*

Analysons maintenant le temps d'exécution sur la courbe 3.13. Le nombre d'inconnues du problème n_{dl} varie comme d^2. En revanche, en terme de longueurs d'onde, l'objet ne change pas, donc l'octree reste le même quel que soit d. Par conséquent, pour ce qui est des temps

d'exécution :

- la partie « interactions proches » va croître comme ${n_{dl}}^2$ (ou encore d^4) ;
- les parties « initialisation » et « intégration » vont croître comme n_{dl} (c.à.d. d^2) ;
- les autres parties vont rester constantes.

Cela se vérifie en pratique. Lorsque d passe de 3 à 10, le nombre d'inconnues est multiplié par plus de 11, mais le temps d'exécution varie d'un facteur 1,7 seulement, car les interactions lointaines sont encore la partie qui consomme le plus de temps. Avec une méthode itérative classique, le temps pour un produit matrice-vecteur aurait été multiplié par 11^2 (au lieu de 1,7). On en tire la conclusion suivante : le sous-raffinement du maillage, en tant que méthode pour gagner du temps, est très peu efficace avec la FMM. On ne parle ici que des itérations multipôles proprement dites car la matrice des interactions proches a bien sûr une taille et un temps d'assemblage qui croissent comme d^4.

3.4.3.5 Conclusion

Au final, les valeurs de d comprises entre 8 et 12 (c'est-à-dire des finesses de maillage comprises entre $\lambda/8$ et $\lambda/12$) offrent un bon compromis entre le temps d'exécution et la précision. Mais les gains que l'on peut escompter restent modestes, et n'ont rien à voir avec ceux obtenus dans le cas d'une méthode directe ou itérative classique. En particulier, le fait d'avoir localement des triangles de grande taille (avec des tailles d'arête en $\lambda/1{,}7$ pour l'airbus par exemple) détériore sensiblement la précision de la FMM sans pour autant accélérer le calcul.

3.4.4 Formulation intégrale en multipôle

3.4.4.1 Objectif

Nous allons nous intéresser ici aux différentes formulations multipôles en électromagnétisme (EFIE, MFIE, CFIE). On ne va pas étudier leurs propriétés de convergence . On va seulement regarder le temps d'exécution et la précision multipôle de ces différentes écritures de la FMM.

3.4.4.2 Tests

Pour ce faire, nous avons utilisé trois maillages différents, et testé sur chacun d'eux la précision et la vitesse de la méthode multipôle en EFIE, MFIE, et CFIE. Les résultats sont présentés dans le tableau 3.10.

Formulation	Sphère 255792		Cetaf 539100		Airbus 213084	
	écart	temps	écart	temps	écart	temps
EFIE	$1{,}51.10^{-3}$	89,0	$4{,}92.10^{-3}$	298,3	$1{,}34.10^{-2}$	163,6
MFIE	$1{,}11.10^{-3}$	104,4	$1{,}23.10^{-2}$	300,9	$2{,}58.10^{-2}$	179,2
CFIE	$8{,}02.10^{-4}$	107,8	$1{,}07.10^{-2}$	306,6	$2{,}27.10^{-2}$	232,4

TAB. 3.10 – *FMM et formulations intégrales en électromagnétisme*

3.4.4.3 Interprétation des résultats

Dans la mesure où seule la phase d'intégration diffère entre les trois formulations testées en mode multipôle, il n'est pas surprenant d'obtenir des temps quasiment identiques. En ce qui concerne la précision, on trouve que la MFIE est un peu moins précise que l'EFIE, et que la CFIE (qui est une combinaison linéaire des deux précédentes) s'intercale.

3.4.5 Points de Gauss

3.4.5.1 Objectif

Dans la méthode multipôle, on utilise des points de Gauss au cours des phases d'initialisation et d'intégration, pour discrétiser les intégrales sur $\Gamma \cup \mathcal{C}$. On compare les formules à 1 et 3 points de Gauss par triangle. Dans le premier cas, le point d'intégration unique est le centre de gravité du triangle. Dans le second cas, les trois points de Gauss ont pour coordonnées barycentriques (2/3,1/6,1/6) (et permutations circulaires de ce triplet). On va étudier l'impact sur le temps et sur la précision FMM du choix de l'une ou l'autre de ces familles de points de Gauss. Soulignons que ce choix est indépendant de celui fait pour le calcul des interactions proches (qui utilise trois points de Gauss pour les interactions entre triangles distants, et 6 ou 7 points de Gauss ainsi que des formules analytiques pour les interactions entre triangles proches ou confondus).

3.4.5.2 Tests

Pour trois maillages différents (sphère à 255792 inconnues, cetaf à 539100 inconnues, avion à 213084 inconnues), on donne dans le tableau 3.11 la précision et la durée des produits multipôles réalisés en utilisant 1 ou 3 points de Gauss par triangle.

Nombre	Sphère 255792		Cetaf 539100		Airbus 213084	
de points	écart	temps	écart	temps	écart	temps
3 points	$5{,}41.10^{-4}$	157,5	$1{,}16.10^{-3}$	371,9	$3{,}57.10^{-3}$	189,2
1 point	$1{,}51.10^{-3}$	83,0	$4{,}92.10^{-3}$	276,3	$1{,}34.10^{-2}$	131,8

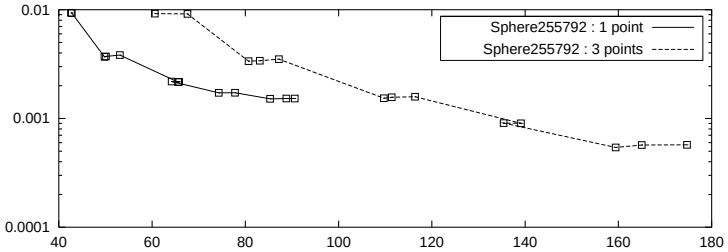
TAB. 3.11 – *Choix du système de points de Gauss*

Le passage de 3 à 1 point de Gauss permet de diviser presque par 3 le temps consacré aux initialisations et intégrations. Avec 3 points de Gauss, ces dernières représentent entre 40 % et 60 % du temps total. Le gain obtenu en passant à 1 point de Gauss s'avère assez conséquent et peut atteindre 50 %.

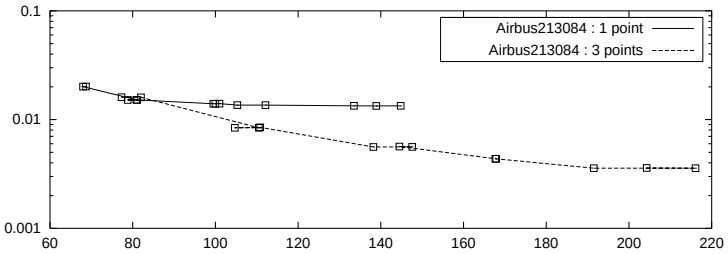
Parallèlement, la précision se détériore d'un facteur 2 à 3. Dans le cas de l'Airbus, l'erreur due à la méthode multipôle passe au dessus du pour-cent. On a vu que la présence de triangles nettement surdimensionnés sur ce maillage conduisait la FMM à traiter comme lointaines des interactions qui étaient en réalité proches. Le fait d'utiliser 3 points de Gauss par triangle permet de calculer plus précisément ces interactions « à problème ». Pour diminuer cette erreur, il serait moins coûteux de raffiner davantage le maillage aux endroits critiques, plutôt que de rajouter des points de Gauss comme on le fait ici. En pratique, dans un contexte industriel, il est toutefois plus simple de s'adapter aux maillages existants, c'est pourquoi cette option « 3 points de Gauss » est utile.

3.4.5.3 Etudes couplées nombre de pôles / nombre de points de Gauss

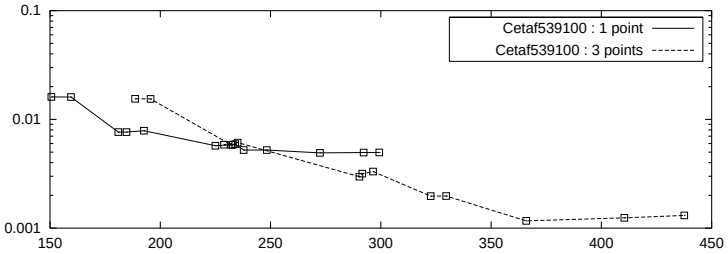
On voit que le choix du système de points de Gauss conduit à faire un compromis entre vitesse et précision, tout comme le choix du nombre de pôles. Il serait incohérent de choisir un grand nombre de pôles associé à un petit nombre de points de Gauss, ou le contraire. Nous allons donc chercher à voir quels sont les choix de paramètres cohérents. Pour ce faire, nous allons reprendre le même type d'étude que pour la figure 3.11 : pour nos maillages de sphère à 255792 inconnues et d'Airbus à 213084 inconnues, pour $C_\varepsilon \in [2,8]$, et pour 1 ou 3 points de Gauss par triangles, on trace la courbe de la précision FMM en fonction de la durée d'une itération. Le résultat est tracé sur la figure 3.14.



(a) Sphère à 255.792 inconnues



(b) Airbus à 213.084 inconnues



(c) Cetaf à 539.100 inconnues

FIG. 3.14 – *Précision et temps d'exécution d'un produit multipôle pour différents nombres de points de Gauss*

Chaque carré sur cette courbe correspond à un couple de paramètre (C_ε, nombre de points de Gauss). Tout point de cette figure qui se trouve à la fois au dessus et à droite d'un autre point est inefficace, car il est à la fois plus lent et moins précis. Naturellement, nous choisiront nos paramètres optimaux parmi les points efficaces. On voit qu'il est possible d'améliorer la précision en passant à 3 points de Gauss par triangle, mais que le surcoût est important. Néanmoins, il faut souligner que ce ralentissement n'affecte que les étapes d'initialisation et d'intégration, qui vont devenir de plus en plus négligeables avec l'augmentation de la taille des objets.

Sur la figure 3.14, la courbe du haut correspond au maillage de sphère à 255792 inconnues. Dans ce cas, le calcul avec 1 point de Gauss est toujours efficace, sauf si on souhaite une grande précision, qui ne peut être obtenue qu'avec 3 points de Gauss. Pour l'Airbus à 213084 inconnues (courbe du milieu), avec 1 point de Gauss la précision plafonne assez vite.

3.4.5.4 Résultats

On a toujours pour objectif de mettre au point une méthode multipôle dont la précision et la vitesse puissent s'ajuster à volonté. Notre étude du nombre de pôles nous a conduit à sélectionner trois valeurs de C_ε (cf. 3.4.1.5). Si on se donne en plus la possibilité d'ajuster le nombre de points de Gauss, on rajoute une liberté qui nous conduit à sélectionner trois nouvelles valeurs.

Les valeurs retenues sont données dans le tableau 3.12.

Précision	C_ε	points	Sphère 255792		Cetaf 539100		Airbus 213084	
		de Gauss	écart	temps	écart	temps	écart	temps
faible	2,0	1	$9{,}37.10^{-3}$	42,6	$1{,}61.10^{-3}$	157,3	$2{,}01.10^{-2}$	68,7
moyenne	7,0	1	$1{,}51.10^{-3}$	83,0	$4{,}92.10^{-3}$	276,3	$1{,}34.10^{-2}$	131,8
élevée	7,0	3	$5{,}41.10^{-4}$	157,5	$1{,}16.10^{-3}$	371,9	$3{,}57.10^{-3}$	189,2

TAB. 3.12 – *Valeurs de C_ε et du nombre de points de Gauss choisies*

3.4.6 Taille des feuilles

3.4.6.1 Présentation

La taille des feuilles est, après le nombre de pôles, le deuxième paramètre essentiel de la méthode multipôle rapide. On note a l'arête des feuilles de l'octree mesurée en nombre de longueurs d'onde. Les feuilles sont donc des boîtes cubiques de côtés $a\lambda$ (en mètre). Lorsque l'on remonte dans l'octree des feuilles vers la racine, à chaque niveau la taille des boîtes est multipliée par deux. Par conséquent, la taille des feuilles détermine la taille des cellules à tous les niveaux, ainsi que le nombre de niveaux. Elle induit aussi le partage entre interactions proches (traitées classiquement) et interactions lointaines (traitées via la FMM).

3.4.6.2 Expériences numériques

Nous allons à nouveau tester la précision et la vitesse de la méthode multipôle pour trois maillages différents, et pour des tailles de feuilles variant de $0{,}1\lambda$ à λ par pas de $0{,}05\lambda$, et pour chacune des combinaisons « pôles/Gauss » du tableau 3.12.

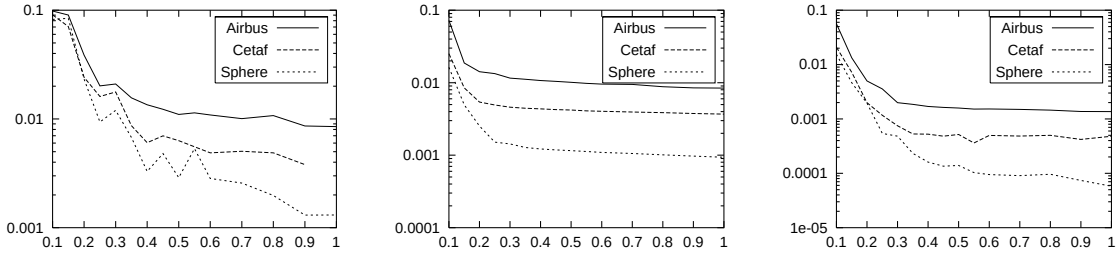
Une partie des résultats obtenus avec le niveau de précision intermédiaire est reprise dans le tableau 3.13.

Taille des feuilles	Sphère 255792		Cetaf 539100		Airbus 213084	
	écart	temps	écart	temps	écart	temps
0,1	$3{,}63.10^{-2}$	114,1	$2{,}51.10^{-2}$	280,9	$7{,}23.10^{-2}$	133,6
0,3	$1{,}43.10^{-3}$	51,5	$4{,}59.10^{-3}$	135,8	$1{,}16.10^{-2}$	71,0
0,5	$1{,}15.10^{-3}$	61,5	$4{,}20.10^{-3}$	143,1	$1{,}01.10^{-2}$	76,5
0,8	$1{,}01.10^{-3}$	129,7	$3{,}86.10^{-3}$	223,8	$8{,}80.10^{-3}$	117,9

TAB. 3.13 – *Etudes de différentes tailles de feuille*

3.4.6.3 Analyse de la précision

Plus les feuilles sont grandes, plus grand est le nombre d'interactions considérées comme proches et traitées classiquement. Du coup, on n'utilisera pas la taille des feuilles comme un paramètre pour améliorer la précision, puisque cela nous conduirait à prendre des feuilles tellement grandes que toutes les interactions seraient proches. En revanche, il convient de voir comment évolue la précision lorsque a se rapproche de zéro.



(a) Précision faible (b) Précision moyenne (c) Précision élevée

FIG. 3.15 – *Précision Multipôle en fonction de la taille des feuilles*

Sur la figure 3.15, on a tracé la précision du produit multipôle en fonction de l'arête des feuilles pour chacun des trois modes de calcul disponibles (précision faible, moyenne ou élevée), et pour nos trois objets (sphère à 255792 inconnues, cetaf à 539100 inconnues, Airbus à 213084 inconnues).

Pour une taille d'arête trop petite, la méthode multipôle diverge. C'est là un fait connu, qui s'explique par la divergence des fonctions de Hankel $h_l^{(1)}(x)$ lorsque x tend vers 0. Or ces fonctions apparaissent dans la définition des fonctions de transferts (équation (1.5)). Par conséquent, on ne peut pas choisir une taille de boîte trop faible. Graphiquement, on observe une nette dégradation de la précision pour une arête inférieure à $0{,}25\lambda$. C'est surtout visible dans le cas de la sphère.

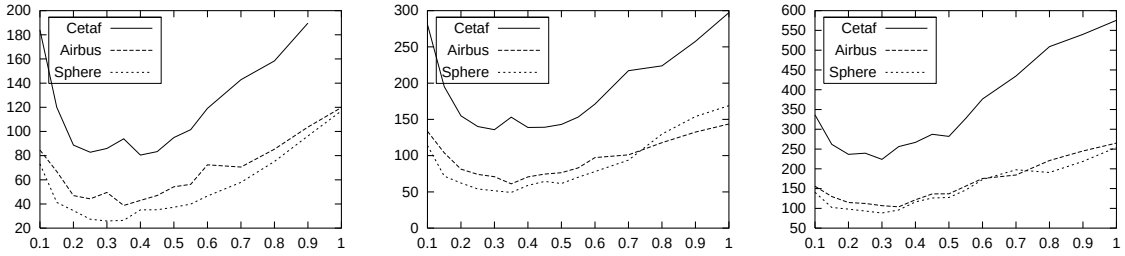
Dans le cas de la précision élevée (graphique de droite), notre objectif est de réaliser le produit multipôle le plus précis possible. La taille de boîte $0{,}35\lambda$ apparaît comme étant le seuil

au-delà duquel il n'y a plus de gain en précision (sauf pour la sphère), mais on privilégie ici les maillages « réalistes ». C'est donc la valeur que nous sélectionnerons.

3.4.6.4 Analyse du temps d'exécution

Cette partie est la mise en œuvre pratique de l'étude de la section 1.3.2.3.5 sur le nombre d'opérations d'un produit multipôle multi-niveau complet. Pour a petit, le nombre de niveaux de l'octree augmente, et le coût des transferts et des montées/descentes à tous les niveaux fait « exploser » la durée d'un calcul FMM comme $\log(1/a)$. Pour a grand, ce sont les interactions proches qui font croître ce temps comme a^2.

On a tracé sur la figure 3.16 la courbe donnant l'évolution du temps en fonction de la taille des boîtes.



(a) Précision faible (b) Précision moyenne (c) Précision élevée

FIG. 3.16 – *Rapidité du produit multipôle en fonction de la taille des feuilles*

Dans le cas de la FMM à précision faible, notre but est de réaliser le produit multipôle le plus rapide possible (tout en conservant une précision de l'ordre de 10^{-2}). La valeur $a = 0{,}25\lambda$ correspond au minimum du temps d'exécution (tout en assurant une précision convenable comme on l'a vu précédemment).

Enfin, pour la FMM à précision intermédiaire, la valeur $a = 0{,}25\lambda$ offre également un bon compromis vitesse/précision. En prenant la même valeur que pour la FMM à précision faible, cela nous permettra d'utiliser la même matrice d'interaction proche.

3.4.6.5 Synthèse

Le tableau 3.14 indique la taille de feuille préconisée en nombre de longueurs d'onde pour chacun des trois niveaux de précision de notre FMM.

Précision de la FMM	Taille de feuille
faible	$0{,}25\lambda$
moyenne	$0{,}25\lambda$
élevée	$0{,}35\lambda$

TAB. 3.14 – *Tailles de feuilles choisies*

Ces valeurs sont nettement supérieures à celle prévue par l'étude simplifiée de la section 1.3.2.3.5 qui était 0,075λ. Comme on l'avait alors signalé, nos simplifications nous avaient conduit à favoriser les petites boîtes en y sous-estimant le nombre de pôles. Néanmoins, on vérifie que la taille de feuille optimale ne dépend pas du nombre de degrés de liberté ou de l'objet considéré.

3.4.7 Niveau plafond

3.4.7.1 Objectif

Nous allons ici vérifier en pratique les prévisions théoriques issues de la section 1.3.2.3.3. On y avait vu que la méthode multipôle la plus rapide était la méthode multi-niveau complète, dans laquelle tous les niveaux de l'arbre étaient parcourus (sauf les niveaux 0 et 1 bien sûr).

3.4.7.2 Tests

Nous reprenons nos trois maillages habituels, et nous regardons les précisions et temps d'exécution obtenus pour chaque niveau plafond possible. Les résultats numériques sont présentés dans le tableau 3.15.

Niveau	Sphère 255792		Cetaf 539100		Airbus 213084	
plafond	écart	temps	écart	temps	écart	temps
2	$1{,}51.10^{-3}$	**83,2**	$4{,}92.10^{-3}$	284,4	$1{,}34.10^{-2}$	134,3
3	$1{,}51.10^{-3}$	102,8	$4{,}92.10^{-3}$	**276,4**	$1{,}34.10^{-2}$	**116,2**
4	$1{,}51.10^{-3}$	237,5	$4{,}91.10^{-3}$	280,5	$1{,}34.10^{-2}$	121,0
5	$1{,}51.10^{-3}$	1113,4	$4{,}91.10^{-3}$	381,8	$1{,}34.10^{-2}$	154,0
6			$4{,}90.10^{-3}$	1188,0	$1{,}34.10^{-2}$	418,1
7					$1{,}34.10^{-2}$	1345,8

TAB. 3.15 – *Choix du niveau plafond*

Au niveau plafond, les transferts se font entre toutes les cellules non-voisines. Si on note N le nombre de cellules au niveau plafond, le nombre de transferts à effectuer à ce niveau croît comme N^2, et la liste de tâches peut atteindre des tailles ingérables (plusieurs centaines de millions de tâches, si $N = 10.000$). C'est pourquoi les algorithmes FMM à 1 et 2 niveaux n'ont pas pu être testés ci-dessus (si on tenait absolument à tester ces méthodes, il suffirait d'augmenter la taille des feuilles, pour diminuer le nombre de cellules à tous les niveaux).

3.4.7.3 Interprétation

Analysons les résultats du tableau 3.15. La première remarque est bien sûr que la précision ne dépend pas du niveau plafond. Comme prévu, la méthode multi-niveau n'introduit pas d'erreur supplémentaire. Cela démontre que notre formule pour choisir le nombre de pôles en fonction de la taille des boîtes est bonne.

En ce qui concerne les temps d'exécution, sur un objet simple et homogène comme une sphère, nos résultats confirment la théorie : l'algorithme complet est bien le plus rapide. Sur des objets plus complexes, plus plats (comme le cetaf) ou filiformes (comme l'Airbus), il est parfois plus intéressant de s'arrêter à un niveau inférieur. Notre étude théorique supposaient que chaque cellule avait en moyenne 4 enfants. C'est toujours vrai quand on se rapproche

des feuilles, ça peut être faux aux plus hauts niveaux, comme le montre le tableau 3.16, qui indique pour chaque maillage le nombre de cellules, le nombre moyen d'enfants par cellules et la taille en Mo pour chaque niveau de l'octree.

Niveau	Sphère 255792			Cetaf 539100			Airbus 213084		
	NbCel	NbEnf	Taille	NbCel	NbEnf	Taille	NbCel	NbEnf	Taille
2	**56**	**4,6**	**5,4**	13	3,2	14,3	14	1,7	15,4
3	256	4,1	8,0	42	3,2	13,2	**24**	**3,8**	**7,5**
4	1040	3,9	11,5	**133**	**5,0**	**12,7**	90	4,2	8,6
5	4016	3,9	17,6	667	3,9	20,8	374	3,6	11,7
6	15512		38,3	2578	4,1	28,4	1349	4,0	14,9
7				10652	4,0	46,8	5385	3,9	23,7
8				42436		104,9	21262		52,6

TAB. 3.16 – *Nombre de cellules et nombre moyen d'enfants par cellule à chaque niveau*

Dans l'algorithme FMM, lors de la remontée vers la racine, l'exploration d'un niveau supplémentaire n'est rentable que si l'augmentation de la taille des fonctions de radiation ($\times 4$ environ) est compensée par une diminution équivalente du nombre de ces fonctions. En pratique, il y a un critère simple permettant de savoir jusqu'où monter dans l'arbre : il faut s'arrêter au niveau le plus petit en mémoire. Il n'y a pas (encore) de justification rigoureuse à cela, mais c'est quelque chose que l'on peut vérifier sur les 3 cas testés ici. Pour la sphère, le cetaf et l'Airbus, cela conduit à choisir comme niveau plafond les niveaux 2, 4 et 3 respectivement (en gras dans le tableau 3.16), ces choix conduisent aux temps d'exécution quasi-minimums à chaque fois.

3.4.8 Précision machine

La plupart des microprocesseurs disposent aujourd'hui de deux types de format de stockage pour les nombres réels – définis par des normes IEEE. Ces formats sont présentés dans le tableau 3.17.

Nom	Taille (octets)	Chiffres significatifs	Exposant maximal
simple	4	≈ 8	≈ 40
double	8	≈ 16	≈ 300

TAB. 3.17 – *Stockage des nombres réels*

Avec 8 chiffres significatifs, le format simple précision paraît amplement suffisant pour des calculs de FMM dont la précision excède rarement 10^{-4}. Nous avons donc choisi de tester deux modes de calcul, s'appuyant respectivement sur des nombres flottants stockés en simple et double précision.

En mode « double précision », tous les calculs sans exception sont traités dans ce format. En mode « simple précision », les calculs analytiques (trigonométrie, fonctions spéciales de Hankel et de Bessel, polynômes de Legendre, fonctions de transfert et de translation, ...) sont toujours réalisés en double précision, puis le résultat final est éventuellement converti en simple précision. Cela permet de limiter les erreurs d'arrondis, surtout dans les calculs de

sommes de séries, ou de formules de récurrence. Les fonctions de radiation sont entièrement gérées en simple précision. Nous allons regarder l'impact de ces modifications sur les temps d'exécution, sur la précision et sur la mémoire vive requise.

Stockage des flottants	simple	double
Cetaf 539100 :		
précision	$4{,}924.10^{-3}$	$4{,}924.10^{-3}$
temps (s)	131,3	183,2
mémoire (Mo)	335	592
Airbus 213084 :		
précision	$1{,}337.10^{-2}$	$1{,}337.10^{-2}$
temps (s)	73,2	100,4
mémoire (Mo)	189	339
Sphère 255792 :		
précision	$1{,}515.10^{-3}$	$1{,}515.10^{-3}$
temps (s)	49,6	78,1
mémoire (Mo)	121	207

TAB. 3.18 – *Impact sur la FMM de l'utilisation de flottants simples ou doubles*

Les résultats obtenus figurent dans le tableau 3.18. On le voit, l'utilisation de la simple précision permet de gagner en vitesse et en consommation mémoire quasiment sans dégrader la précision de la FMM (aucune différence sur les 4 premières décimales). On aurait donc tort de s'en priver. Soulignons toutefois que les tests ci-dessus ont été réalisés sur IBM SP3, et que le gain en rapidité dépendra beaucoup du type de processeurs utilisés et de la vitesse avec laquelle ils traitent les calculs en simple et double précision. Il est tout à fait possible que le gain en temps soit nul sur certains ordinateurs. Par contre, quelle que soit la machine utilisée, le gain en mémoire demeure.

3.4.9 Nombre de composantes

Dans la section 1.2.2, nous avons présentés trois écritures multipôles, utilisant des fonctions de radiation respectivement à 4, 3 et 2 composantes. Dans le premier cas il s'agissait des composantes (x,y,z,div), dans le deuxième cas des composantes (x,y,z), et dans le dernier cas des composantes (θ,ϕ). Ces trois produits multipôles sont mathématiquement équivalents, mais la discrétisation par éléments finis et la FMM elle-même peuvent introduire des différences.

Regardons tout d'abord la précision. Le passage de 4 à 3 composantes s'accompagne d'un léger gain en précision. Rappelons que la quatrième composante (la divergence) est supprimée via une intégration par partie (cf. section 1.2.2.1.1). La discrétisation de l'intégrale, selon qu'elle est réalisée avant ou après cette transformation, conduit à des résultats numériques différents. On pouvait donc s'attendre à obtenir des scores de précision qui diffèrent sensiblement. Nous n'avons pas encore d'explication au fait que cette différence se fasse dans le sens d'une plus grande précision sur les trois cas testés. En revanche, le passage de 3 à 2 composantes se fait sans modification de la précision sur les trois premiers chiffres significatifs. Le passage des coordonnées cartésiennes (x,y,z) aux coordonnées sphériques tangentielles (θ,ϕ) se fait sans dégradation des données. En outre, comme souligné au paragraphe 1.3.1.6.4, il n'est pas nécessaire de modifier le nombre de points de quadrature de $\mathcal{S}$ pour accompagner l'augmentation de la largeur de bande des fonctions manipulées.

Nombre de composantes	4	3	2
Cetaf 539100 :			
précision	$7{,}300.10^{-3}$	$4{,}924.10^{-3}$	$4{,}924.10^{-3}$
temps (s)	277.1	219.4	166.64
mémoire (Mo)	591	463	335
Airbus 213084 :			
précision	$2{,}192.10^{-2}$	$1{,}337.10^{-2}$	$1{,}337.10^{-2}$
temps (s)	157,8	123,3	91,9
mémoire (Mo)	339	264	189
Sphère 255792 :			
précision	$2{,}265.10^{-3}$	$1{,}512.10^{-3}$	$1{,}515.10^{-3}$
temps (s)	105,7	84,6	63,0
mémoire (Mo)	207	164	121

TAB. 3.19 – *Performances de la méthode multipôle en fonction du nombre de composantes*

Regardons maintenant les temps d'exécution. Les phases de montées, de descentes et de transferts vont avoir un temps d'exécution proportionnel au nombre de composantes utilisées. Les phases d'interaction proche sont inchangées. Enfin, pour les phases d'initialisation et d'intégration, le calcul à 2 composantes est le plus lent à cause du passage aux coordonnées sphériques. Globalement, le gain est très intéressant, et le temps nécessaire à la réalisation du produit multipôle est presque divisé par 2 en passant de 4 à 2 composantes. Il en est de même pour la mémoire vive utilisée, qui dans le même temps diminue de 40%.

3.4.10 Stockage des matrices de transfert et de translation

Nous allons ici vérifier l'intérêt de conserver certaines données entre deux produits multipôles consécutifs. D'une part, on se propose de conserver les matrices de transfert, comme évoqué à la section 2.2.2.3. D'autre part, on envisage de stocker les matrices de translation utilisées dans les phases d'initialisation et d'intégration, comme proposé au paragraphe 2.4. Ces modifications n'ont aucun impact sur la précision du calcul. Nous allons juste voir si le surcoût lié au stockage sur disque est compensé par le gain en terme de temps de calcul. Cela dépendra de la puissance relative du processeur et du système disque.

Pour ce faire, nous allons réaliser notre test sur deux machines : la première est un PC équipé d'un processeur basique (Pentium III) et d'un disque très performant (SCSI), la seconde est une IBM équipé de processeurs très performants (power 3) et d'un système disque moyen (en théorie, il devrait être excellent, mais en pratique il nous a été impossible d'en tirer de bonnes performances).

Le tableau 3.20 présente les résultats obtenus sur la première machine (PC Pentium III + SCSI). Le stockage des fonctions de transfert entraîne un gain d'autant plus important que le nombre de niveaux de l'octree est grand (7 pour la sphère, 9 pour l'Airbus et le cetaf). On a vu que le temps de calcul d'une fonction de transfert était multiplié par 8 à chaque fois que l'on montait d'un niveau dans l'arbre. Le gain lié au stockage de ces fonctions grandira donc avec la taille des cas de calcul traités. Pour les fonctions de translation, la quantité de données à sauvegarder est conséquente, mais le bénéfice est néanmoins net. Sur ce type de machine, on aura donc tout intérêt à utiliser à la fois le stockage des fonctions de transfert et de translation.

Stockage	aucun	transfert	translation
Cetaf 539100 :			
temps (s)	190,0	185,3	173,6
disque (Mo)	1990	1997	2655
Airbus 213084 :			
temps (s)	108,0	101,5	96,6
disque (Mo)	1058	1066	1328
Sphère 255792 :			
temps (s)	76,3	74,6	64,9
disque (Mo)	1534	1536	1832

TAB. 3.20 – *Stockage des fonctions de transfert et de translation (Pentium III, SCSI)*

Le tableau 3.21 présente les résultats obtenus sur la seconde machine (IBM SP3). On n'y donne que les temps d'exécution, puisque l'utilisation de l'espace disque est (quasiment) la même. On voit que selon les cas, le gain en temps peut être intéressant (cetaf), presque inexistant (Airbus) voire même négatif (sphère). Sur une machine parallèle, où les accès aux moyens de stockage sont souvent partagés entre tous les utilisateurs, il semble préférable de ne pas conserver les fonctions de translation. En revanche, il est certain que sur des très grands cas de calcul, le stockage des fonctions de transfert sera à terme rentable.

Stockage	aucun	transfert	translation
Cetaf 539100 :			
temps (s)	145,4	140,6	134,9
Airbus 213084 :			
temps (s)	80,4	77,5	78,5
Sphère 255792 :			
temps (s)	54,5	55,3	56,0

TAB. 3.21 – *Stockage des fonctions de transfert et de translation (IBM SP3)*

3.5 Comparaison par machine

Dans cette section, nous allons comparer les performances d'un grand nombre de processeurs en utilisant notre code multipôle comme outil de mesure. Nous essaierons d'en déduire les caractéristiques qui font qu'une machine est adaptée ou non au calcul multipôle. Nous regarderons ensuite le gain obtenu par l'utilisation de bibliothèques optimisées éventuellement machine-dépendantes comme FFTW, ESSL, ATLAS, COMPLIB.

3.5.1 Comparaison matérielle

Le tableau 3.22 présente les différentes machines testées, en soulignant en particulier le type et la fréquence du microprocesseur utilisé, le type de mémoire et de disque, ainsi que le système d'exploitation (OS) utilisé. Les premières lignes correspondent à des stations de travail, les dernières à des calculateurs parallèles situés au CINES et testés séquentiellement.

A chaque fois, le compilateur du constructeur a été utilisé, sauf pour les machines à base de Pentium, pour lesquelles les deux compilateurs Gnu et Portland Group ont été testés.

Marque et processeur	Mémoire	Disque	OS
Compaq Pentium III 866 MHz	1 Go RDRAM	36 Go SCSI Ultra160	Linux 2.2
Assembleur Pentium III 933 MHz	512 Mo SDRAM	18 Go IDE	Linux 2.2
Sun UltraSPARC-IIi 360 MHz	1 Go -	9 Go IDE	SunOS 5.6
DEC Alpha ev67 500 MHz	1 Go -	9 Go IDE	OSF1 O 4.0
IBM SP3 Power3 375 MHz	1 Go/proc -	- -	AIX 4
SGI Origin 3800 MIPS R14000 400MHz	512 Mo/proc -	- -	IRIX 6.5 (32 bits)

TAB. 3.22 – *Présentation des machines testées*

Le cas test utilisé est la sphère à 255.792 inconnues. Pour chaque configuration testée, on indique dans le tableau 3.23 d'une part le temps (en minutes) nécessaire au démarrage des itérations (construction des structures de données et assemblage de la matrice des interactions proches) et d'autre part le temps (en secondes) requis pour faire un seul produit matrice-vecteur multipôle (en choisissant les paramètres par défaut du tableau 3.4).

Marque	Démarrage (min)	Itération (s)
Compaq + GNU	12,8	61,1
Assembleur + GNU	15,8	86,2
Compaq + PORTLAND	**11,0**	60,5
Assembleur + PORTLAND	14,0	77,2
Sun	45,8	225,1
DEC	12,4	**42,4**
IBM SP3	18,2	57,9
SGI Origin 3800	11,8	70,6

TAB. 3.23 – *Performances des machines de test sur la sphère 255792*

A l'exception de la station SUN, les temps de démarrage sont assez proches. Dans cette partie du calcul, il y a beaucoup d'accès mémoire et disque désordonnés. Les machines ayant une mémoire très rapide et un disque très performant dominent les débats : il s'agit du PC compaq et de l'Origin. A contrario, la SP3 est handicapée par des accès disques assez piteux. Pour ce qui est des temps d'itération, ce sont les machines ayant un processeur puissant et une mémoire vive rapide qui obtiennent les meilleurs scores, en particulier la DEC alpha.

Il est à noter que le cas de calcul utilisé est relativement petit, et ne permet pas à la puissance des calculateurs de s'exprimer pleinement. Le tableau 3.24 donne le même type de résultats que précédemment sur un cas comportant 4 fois plus d'inconnues. On constate que l'IBM SP3 est le plus rapide pour les itérations multipôles, et l'Origin pour le lancement du calcul (assemblage de la matrice des interactions proches en particulier). Ceci dit, notre PC

compaq est loin d'être ridicule face à ces deux « mastodontes » (numériques), qui prendront bien sûr leur revanche en parallèle.

Marque	Démarrage (min)	Itération (s)
Compaq + GNU	56,7	346,8
IBM SP3	62,0	233,6
SGI Origin 3800	37,8	465,5

TAB. 3.24 – *Performances sur la sphère 1.023.768*

Dans l'absolu, la meilleur performance est obtenue sur station DEC alpha. Mais il apparaît que la station de travail Compaq, équipée d'un système disque performant, offre ici la meilleure homogénéité. On voit en particulier l'importance d'avoir une mémoire vive rapide (RDRAM ou DDR-RAM) et un disque performant (SCSI). Comme souligné à la section 3.4.2, cela est encore plus vrai dans le cas d'un calcul en mode out-of-core.

3.5.2 Comparaison logicielle

On va s'intéresser dans cette section à l'accélération obtenue en utilisant des bibliothèques optimisées multiplateformes ou liées à une machine en particulier. Les bibliothèques en question sont au nombre de quatre :

FFTW : La bibliothèque FFTW a été développée au MIT par Matteo Frigo et Steven G. Johnson. C'est un ensemble de routines permettant de réaliser des transformées de Fourier rapides de manière extrêmement rapide sur n'importe quelle machine. Avant d'utiliser FFTW, on utilisait la FFT proposée par Numerical Recipes [30]. Cette dernière est moins rapide, et ne marche que pour des vecteurs dont la taille est une puissance de 2. On est donc obligé d'arrondir le nombre de directions ϕ de nos discrétisations de $\mathcal{S}$ à la puissance de 2 immédiatement supérieure, ce qui ralentit toutes les étapes du calcul. Comme on le voit dans le tableau 3.25, l'usage de FFTW conduit à un gain en temps considérable.

ATLAS : C'est une implémentation des routines basiques d'algèbre linéaire qui s'adapte automatiquement au type de processeur utilisé. ATLAS est un projet ambitieux, encore en développement, qui pourrait à terme fédérer toutes les librairies BLAS constructeur.

COMPLIB : C'est la bibliothèque mathématique de Silicon Graphics, qui contient notamment BLAS et LAPACK.

ESSL : c'est l'équivalent de COMPLIB chez IBM.

ATLAS, COMPLIB et ESSL sont trois implémentations améliorées des librairies d'algèbre linéaire standards BLAS et LAPACK utilisées chaque fois que possible dans notre code. Le gain qu'elles apportent est nettement moins important que dans le cas de FFTW.

3.6 Scalabilité

3.6.1 Introduction

Nous allons étudier la scalabilité numérique de notre code multipôle, c'est-à-dire son comportement lorsque le nombre d'inconnues du problème n_{dl} augmente, toutes choses égales par ailleurs. Pour des raisons de simplicité à générer des maillages, nous allons utiliser des sphères

Bibliothèque	Machine	Temps (en s)	
		Avec	Sans
FFTW	PC	61,6	89,5
ATLAS	PC	59.3	59.7
COMPLIB	Origin	70,6	72,0
ESSL	SP3	53,0	54.2

TAB. 3.25 – *Performances des librairies spécialisées : durée en secondes d'un produit FMM*

pour cette étude. Cela facilite la convergence de la méthode numérique, mais du point de vue de la méthode multipôle cela ne change rien que l'objet soit une sphère ou quelque chose de plus complexe.

On va s'intéresser en particulier à la manière dont évoluent les temps d'exécution et les besoins en mémoire (vive et disque) afin de vérifier si la croissance théorique en $n_{dl} \log n_{dl}$ tant espérée est bien au rendez-vous. Les prévisions théoriques se trouvent à la section 1.3.2.3.4, et les mesures qui suivent ont été réalisées sur le PC compaq (cf. tableau 3.22).

Nous allons utiliser pour ces tests des sphères de diamètre variant de λ à 32λ par pas de λ, maillée avec 10 points par longueur d'onde. Le nombre d'inconnues varie de 972 à 1.160.652. Au-delà, le système commence à avoir la mémoire vive saturée, et les performances chutent pour des raisons « indépendantes de notre volonté ». En utilisant l'option out-of-core, nous aurions pu traiter des cas plus importants, mais nous ne l'avons pas fait afin d'avoir des temps d'exécution comparables entre eux.

3.6.2 Temps

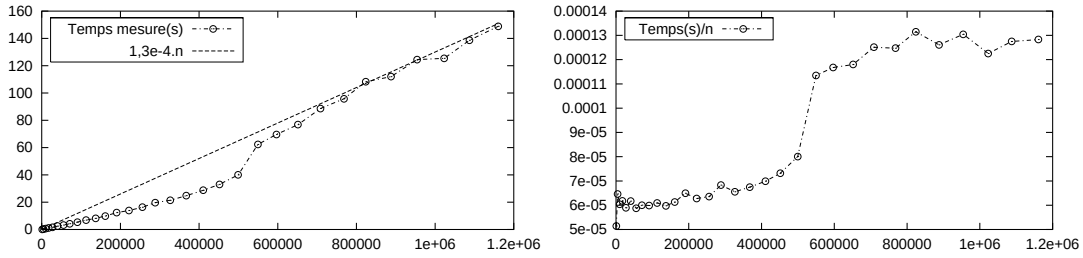
3.6.2.1 Initialisations, intégrations, interactions proches

On étudie le temps moyen par itération consacré à ces trois types d'opération : intégrations, initialisations, interactions proches. On a choisi de les regrouper car pour ces trois parties, on avait prévu une croissance linéaire en n_{dl} :

$$\begin{cases} f_{init} = n_{dl}\left[288\pi^2 a_f^2 + \frac{240\pi^2}{\sqrt{3}d^2}\right], \\ f_{integ} = n_{dl} 24\pi^2 \left[\frac{6\sqrt{3}}{d^2} + 66 a_f^2\right], \\ f_{proche} = n_{dl} 18\sqrt{3} d^2 a_f^2 \end{cases}$$

La figure 3.17 donne le temps en secondes par itération pour ces trois types de calculs en fonction du nombre de degrés de liberté n_{dl}. La courbe de gauche donne le temps en seconde, et afin de vérifier que l'on a bien une croissance linéaire en n_{dl}, on a tracé à droite la pente, c'est-à-dire le temps divisé par n_{dl}.

La courbe de droite est composée de deux portions horizontales, une basse en dessous de 500.000 degrés de liberté, et une haute au dessus. Cela provient de l'utilisation par le système d'exploitation (Linux, ici) de la mémoire vive libre de la machine (1 Go) comme mémoire cache pour le disque. En dessous de 500.000 inconnues, l'OS parvient à conserver en RAM la plupart des données normalement stockées sur disque, ce qui permet d'avoir des accès hyper rapides à ces fichiers. Progressivement, lorsque la taille des problèmes traités augmente, la mémoire vive disponible diminue, la taille des fichiers augmente, et ce mécanisme de cache devient de moins en moins efficace. C'est pourquoi sur la figure 3.17b, on observe une première partie



(a) Temps (b) Temps divisé par n_{dl}

FIG. 3.17 – *Durée cumulée des initialisations + intégrations + interactions proches*

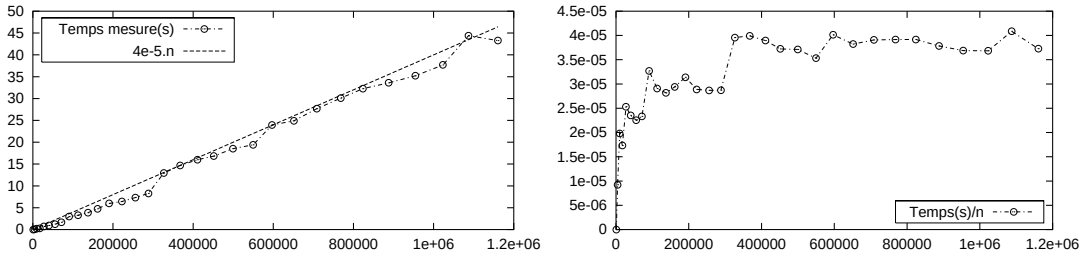
avec une pente faible correspondant à un calcul « dans le cache », puis une seconde partie avec une pente élevée pour un calcul « sur le disque ». Graphiquement, on trouve $t \approx 1{,}3.10^{-4} n_{dl}$ (pour n_{dl} grand).

3.6.2.2 Montées et descentes

Pour ce type d'opération, la croissance prévue est :

$$f_{mont/desc} = 2\pi^2 \frac{n_{dl}}{d^2} \left[52\sqrt{3}(N-3) + 96\pi a_f (2^{N-3} - 1) \right]$$

où N est le nombre de niveaux de l'octree : $N \approx \log_2(\sqrt{n_{dl}})$. On a donc la somme d'un terme en $n_{dl}^{3/2}$ correspondant aux extrapolations/réductions, et un terme en $n_{dl} \log n_{dl}$ issu des translations. La figure 3.18a représente le temps par itération consacré aux montées/descentes en fonction de n_{dl}, la figure 3.18b représente ce même temps divisé par n_{dl}.



(a) Temps (b) Temps divisé par n_{dl}

FIG. 3.18 – *Durée des montées et descentes*

On observe à nouveau une croissance linéaire du temps en fonction de n_{dl} caractérisée par l'aspect « plat » de la pente sur la courbe 3.18b. Cela est un peu surprenant, on a l'habitude d'observer des résultats moins bons que ceux prévus par la théorie, mais ici c'est le contraire qui se produit : l'étude théorique prévoyait un temps t de la forme $t = a.n_{dl} \log n_{dl} + b.n_{dl}^{3/2}$ et on observe $t = 4.10^{-5} n_{dl}$.

Pour expliquer cela, nous avançons une hypothèse différente pour le terme en $n_{dl} \log n_{dl}$ et celui en $n_{dl}^{3/2}$. Pour ce dernier, l'étude de la section 1.3.2.3.6 nous a montré qu'il n'était à prendre en compte que pour n_{dl} très grand (au-delà de 10^7). Autrement dit, même s'il est là, la constante b qui le précède est trop petite pour que l'on puisse le remarquer graphiquement.

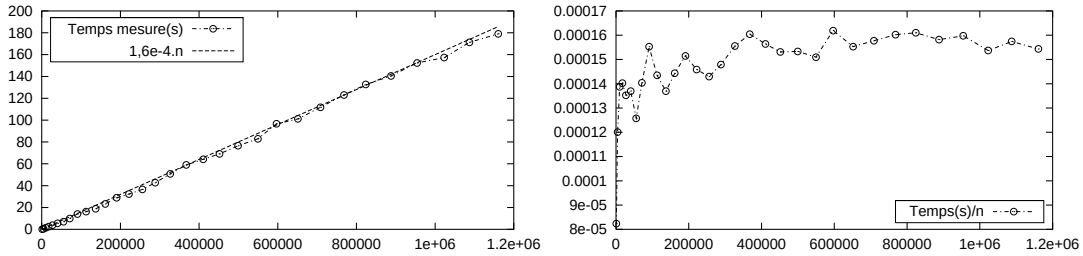
De son côté, le terme en $n_{dl} \log n_{dl}$ représente le temps consacré aux translations des fonctions de radiation. A chaque niveau, ce temps est théoriquement le même (proportionnel à n_{dl}, cf. section 1.3.2.3.2). On le multiplie par le nombre de niveaux (proportionnel à $\log n_{dl}$) pour obtenir le coût total : $n_{dl} \log n_{dl}$. Or en pratique, le coût des translations à un niveau donné n'est pas du tout constant d'un niveau à l'autre. Ce coût est proportionnel à la taille du niveau considéré, or ces tailles décroissent lorsqu'on se rapproche de la racine (cf. tableau 3.16). Conséquence de cette décroissance, le coût total des translations apparaît plus proche de n_{dl} que de $n_{dl} \log n_{dl}$.

3.6.2.3 Transferts

Le coût total théorique des transferts est donné par :

$$f_{transfert} = 864\sqrt{3}\pi^2 \frac{n_{dl}}{d^2}(N-2)$$

toujours avec $N \approx \log_2(\sqrt{n_{dl}})$. On doit donc observer une croissance en $n_{dl} \log n_{dl}$. La courbe 3.19a trace le temps par itérations consacré aux transferts en fonction de n_{dl}, la courbe 3.19b trace ce temps divisé par n_{dl}.



(a) Temps (b) Temps divisé par n_{dl}

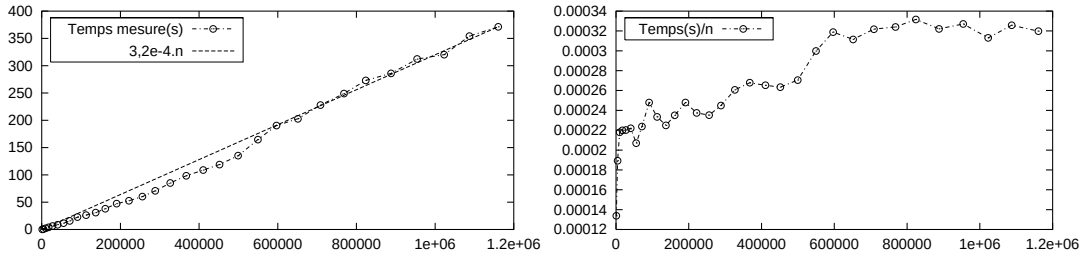
FIG. 3.19 – *Durée des transferts*

Comme pour les montées/descentes, la croissance observée est en n_{dl} et non en $n_{dl} \log n_{dl}$ comme attendu. La raison est la même : la décroissance de la taille des niveaux quand on se rapproche de la racine. On observe $t = 1{,}6.10^{-4} n_{dl}$.

3.6.2.4 Temps cumulé

On trace maintenant (courbe 3.20) le temps total nécessaire à notre station de travail pour réaliser un produit multipôle à un seul second membre.

Cette courbe étant la somme des trois précédentes, on retrouve bien sûr la croissance linéaire avec, pour n_{dl}, la valeur approchée $t = 3{,}2.10^{-4} n_{dl}$.

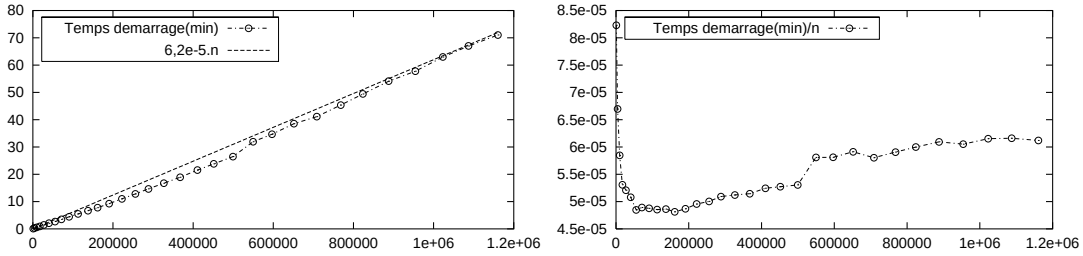


(a) Temps (b) Temps divisé par n_{dl}

FIG. 3.20 – *Durée totale d'un produit multipôle*

3.6.2.5 Démarrage

Le démarrage du calcul comporte deux grandes parties : d'une part la création de l'octree, d'autre part l'assemblage de la matrice des interactions proches. On devrait observer un temps en $n_{dl} \log n_{dl}$ pour la première partie, et en n_{dl} pour la seconde. La courbe 3.21 représente le temps de démarrage (en minutes) pour des sphères comportant n_{dl} inconnues.



(a) Temps (b) Temps divisé par n_{dl}

FIG. 3.21 – *Démarrage d'un calcul multipôle*

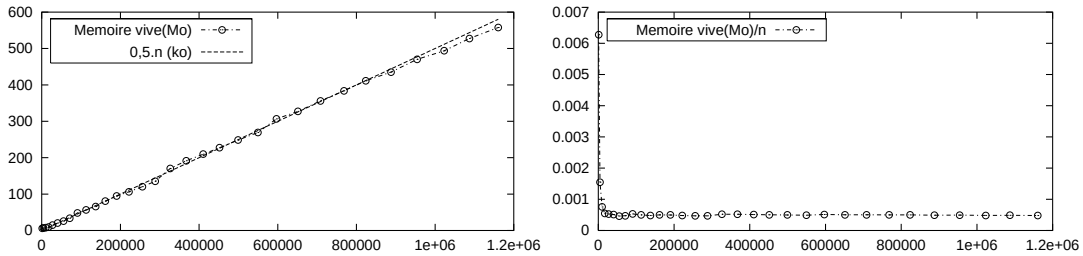
Ici encore, et toujours pour les mêmes raisons, il n'y a pas de croissance en $n_{dl} \log n_{dl}$. Sur le graphique 3.21b, pour n_{dl} petit, on observe comme sur la courbe 3.17b un palier inférieur correspondant aux calculs pour lesquels le cache-disque est efficace, et un palier supérieur lorsque le cache-disque sature. Pour n_{dl} grand, on observe $t = 6{,}2.10^{-5} n_{dl}$ (en minutes).

3.6.3 Mémoire

3.6.3.1 Mémoire vive

On trace (courbe 3.22) la consommation maximale en mémoire vive de notre code multipôle dans le cadre d'un calcul in-core sur des sphères de taille croissante. Comme on l'a vu à la section 2.5.4, le maximum est atteint au milieu des produits multipôles, et comporte d'une

part des données liées au maillage (taille en n_{dl}) et d'autre part les fonctions de radiation (taille cumulée théorique en $n_{dl} \log n_{dl}$).



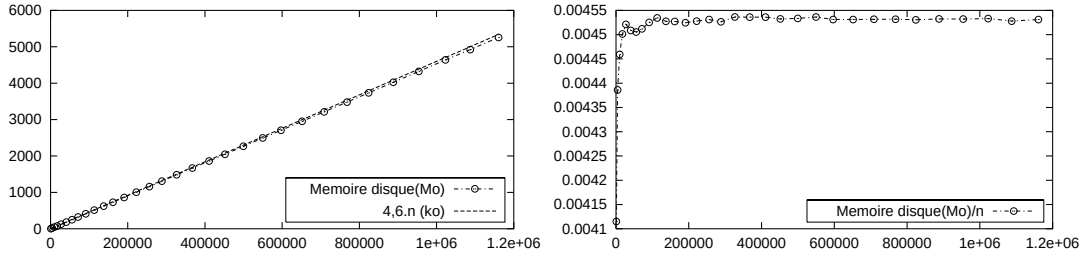
(a) Mémoire vive (b) Mémoire vive divisée par n_{dl}

FIG. 3.22 – *Mémoire vive utilisée par un calcul multipôle*

Encore une fois, pas de croissance en $n_{dl} \log n_{dl}$ à se mettre sous la dent, toujours pour la même raison. C'est à rapprocher de la figure 2.16 (de la section 2.5.4) représentant la mémoire vive consommée au cours d'une itération multipôle dans le cas d'une sphère à 1.023.168 inconnues. On y voit que le maximum de mémoire utilisée est déjà presque atteint dès la fin des transferts au niveau des feuilles. Autrement dit, la taille de l'octree entier est presque égale à deux fois la taille du niveau des feuilles (qui croît comme n_{dl}). Du coup, la taille de l'octree croît elle aussi comme n_{dl} (et non $n_{dl} \log n_{dl}$).

Graphiquement, on trouve la consommation en mémoire vive $m_v = 0{,}5 n_{dl}$ (en kilo-octet).

3.6.3.2 Mémoire disque



(a) Espace disque (b) Espace disque divisé par n_{dl}

FIG. 3.23 – *Espace disque utilisé par un calcul multipôle*

La courbe 3.23 donne la taille totale utilisée sur disque par le code multipôle. Cela comprend principalement la matrice des interactions proches, les données liées au maillage, et les vecteurs de travail des solveurs. Ici, pour une fois, la théorie et la pratique se confondent : la croissance est linéaire en n_{dl} comme prévu. On mesure sur le graphique la consommation en espace disque $d = 4{,}6 n_{dl}$ (en kilo-octet).

3.6.4 Interprétation

La tableau 3.26 récapitule les résultats obtenus dans cette section consacrée à la scalabilité numérique de la méthode multipôle.

Mémoire :	
Vive	0,5 ko par dl
Disque	4,6 ko par dl
Temps :	
Itérations (s)	$3{,}2.10^{-4}.n_{dl}$
Démarrage (min)	$6{,}2.10^{-5}n_{dl}$

TAB. 3.26 – *Scalabilité de la FMM : récapitulatif*

L'information essentielle est que rien ne croît en $n_{dl} \log n_{dl}$ ici : ni la mémoire, ni le temps. Nous allons tâcher d'expliquer pourquoi.

Les études théoriques sont basées sur une formule donnant le nombre de pôles simplifiée :

$$L = 2\pi\sqrt{3}a \tag{3.2}$$

alors que la formule effectivement implémentée s'écrit :

$$L = 2\pi\sqrt{3}a + C_\varepsilon \log_{10}(2\pi\sqrt{3}a + \pi) \tag{3.3}$$

Par exemple, dans le cas d'une sphère à 1,8 millions d'inconnues, le tableau 3.27 donne pour chaque niveau de l'octree le nombre de pôles estimé par la formule (3.2), le nombre utilisé par le programme (formule (3.3)avec $C_\varepsilon = 7$), et le ratio entre ces deux grandeurs. A un niveau donné, la charge de travail et la quantité de mémoire consommée sont proportionnelles à la taille des fonctions de radiation, donc au carré du nombre de pôle. C'est pourquoi la dernière colonne donne ce même ratio élevé au carré. Par exemple, au niveau 5 la charge de travail réelle sera 2,18 fois plus importante que celle estimée.

Niveau	Nombre de pôles		ratio	ratio^2
	estimé	réel		
2	174	189	× 1,09	× 1,18
3	87	100	× 1,15	× 1,32
4	43	55	× 1,28	× 1,64
5	21	31	× 1,48	× 2,18
6	10	18	× 1,8	× 3,24
7	5	11	× 2,2	× 4,84
8	2	8	× 4	× 16

TAB. 3.27 – *Erreur sur le nombre de pôles*

On voit que l'erreur relative commise est beaucoup plus importante aux niveaux bas de l'arbre. Par conséquent, le comptage du nombre d'opérations des sections 1.2.2.4 (FMM mono-niveau) et 1.3.2.3 (FMM multi-niveau) sous-estime beaucoup le poids relatif des niveaux bas, et a contrario estime presque correctement le poids des niveaux hauts. Or, l'estimation de croissance en $\mathcal{O}(n_{dl} \log n_{dl})$ provient du fait que le nombre de niveau est en $\mathcal{O}(\log n_{dl})$, et que la charge de travail à chaque niveau est supposée être la même, en $\mathcal{O}(n_{dl})$. On vient de voir

qu'au contraire, cette charge de travail diminuait à mesure que l'on montait dans l'arbre : elle est bien en $\mathcal{O}(n_{dl})$ au niveau des feuilles, mais décroît ensuite. Aussi, on ne doit pas être surpris d'obtenir une complexité inférieure à notre prévision théorique.

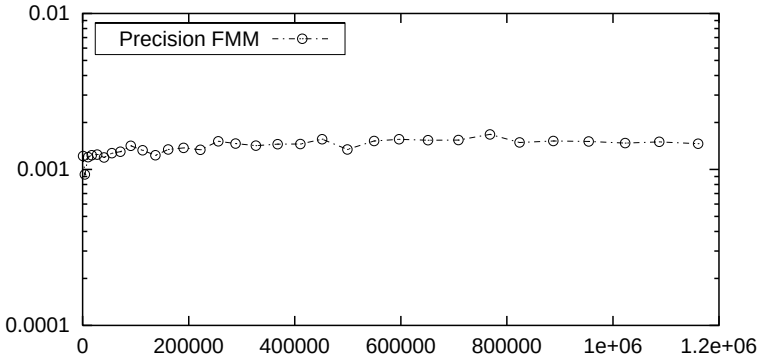


FIG. 3.24 – *Précision du produit multipôle*

Les niveaux les plus hauts étant plus petits que prévu, il convient alors de se poser la question : y a t-il suffisamment de pôles aux niveaux élevés ? Leur taille est-elle suffisante pour préserver toute l'information, et par voie de conséquence la précision du calcul ? La figure 3.24 répond à cette interrogation. On y a tracé la précision FMM de tous les calculs réalisés pour cette étude. On voit clairement que l'augmentation du nombre de degrés de liberté, et du nombre de niveaux de l'octree se fait sans dégradation de la précision. Par conséquent, la formule utilisée pour donner le nombre de pôles est satisfaisante.

Terminons cette section par une mise en garde. Les résultats résumés dans le tableau 3.26 ont été obtenus sur des sphères comportant jusqu'à 1,2 millions d'inconnues. Il ne permettent pas de savoir ce qui se passera au-delà. Il ne faut donc pas en déduire que la FMM est un algorithme en $O(n_{dl})$. Par ailleurs, les constantes trouvées dépendent énormément de la forme de l'objet, de la finesse du maillage et, naturellement, de la machine utilisée (un Pentium III 866 MHz ici).

3.7 Applications industrielles

Nous allons terminer ce chapitre consacré aux performances de notre implémentation multipôle séquentielle par quelques exemples d'applications quasi industrielles, c'est-à-dire mettant en œuvre un calcul complet (résolution + post-traitement + visualisation) sur un objet complexe (i.e. non-sphérique).

3.7.1 Cetaf

La maquette CETAF n'est certes pas un objet réel, mais c'est un objet réaliste, en ce sens qu'il posera les mêmes difficultés de résolution que n'importe quel avion, voiture, antenne ou autre. Le fait qu'aucune arête ni aucune face ne soit parallèle à une autre rend le calcul encore plus délicat. Les éléments du calcul sont présentés dans le tableau 3.28. On y présente juste les objectifs du calcul : tracer une SER bistatique d'un CETAF de diamètre 70λ à l'aide d'une station de travail DEC.

Objet :	
Nom	CETAF
Nombre d'inconnues	1.056.636
Diamètre	70λ
Calcul :	
Second membre	1 onde incidente
Post-traitement	SER, 1800 directions
Machine	DEC Alpha 667 MHz

TAB. 3.28 – *Présentation du calcul : Cetaf à 1 million d'inconnues*

Méthode numérique :	
Formulation	CFIE
Solveur	GMRES(restart=20)
Résidu normalisé	10^{-2}
FMM :	
Out of core	2 groupes
Niveau plafond	4

TAB. 3.29 – *Méthode de résolution*

Les méthodes mises en œuvre pour accomplir cet objectif sont présentées dans le tableau 3.29. Les paramètres non mentionnés de la FMM sont ceux par défaut (cf. tableau 3.4). La valeur du résidu normalisé demandé peut paraître élevée, mais c'est en général amplement suffisant pour obtenir une courbe de SER très précise.

La convergence a été obtenue en 43 itérations, ce qui est convenable pour un objet complexe présentant une fente, surtout en l'absence de préconditionnement. L'historique de cette convergence se trouve sur la figure 3.25. Les autres paramètres, et en particulier les temps d'exécution, figurent dans le tableau 3.30.

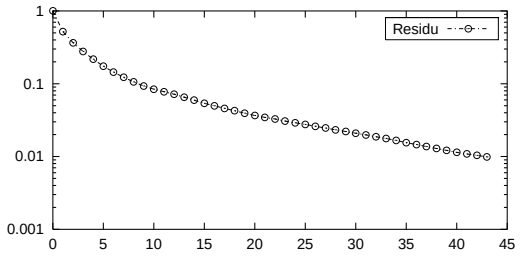


FIG. 3.25 – *Convergence du GMRES(20) pour le cetaf 1 million*

Les temps apparaissant dans le tableau 3.30 ont été grandement tirés vers le haut par la lenteur du disque dur de cette station. Une machine aussi puissante couplée à un disque dur à la hauteur aurait résolu ce problème en deux fois moins de temps.

Le figure 3.26 représente le calcul effectué : le cetaf se trouve dans le plan (xOy) et l'onde incidente arrive dans le plan (xOz) depuis la direction ($\theta = 45$,$\phi = 180$). On regarde le champ diffracté dans toutes les directions du plan (xOz) : $\phi = 180$ et θ varie de 0 à 360. On étudie

Temps :	
Démarrage	140 min
Produit multipôle	573 s (par itération)
Champ lointain	89 min
Total	11,5 h
Mémoire :	
RAM	500 Mo
Disque	5194 Mo

TAB. 3.30 – *Cetaf 1 million : performances*

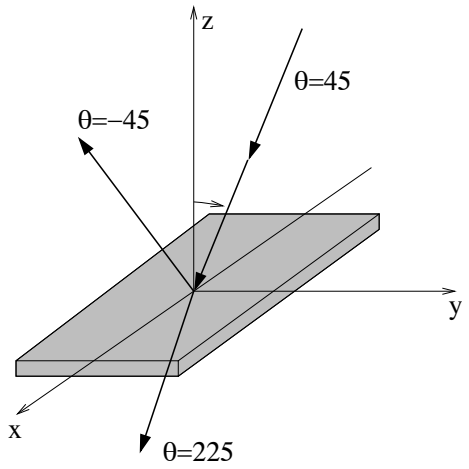


FIG. 3.26 – *Définitions du cas-test*

les ondes incidentes et diffractées en polarisation verticale. On devrait observer deux pics :

- Le premier à $\theta = -45$ correspond à l'onde réfléchie par le plan du cetaf ;
- Le second à $\theta = 180 + 45$ correspond à la zone d'ombre, dans laquelle le champ total est nul. Par conséquent, le champ diffracté y est l'opposé du champ incident, d'où un pic.

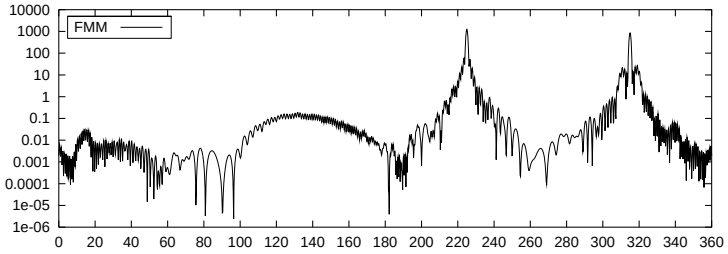


FIG. 3.27 – *SER du cetaf 1 million*

La courbe 3.27 est le résultat recherché. On y voit bien les deux pics attendus. Pour faire une comparaison plus complète, il suffirait de faire le même calcul avec une code de GTD (théorie géométrique de la diffraction) et de comparer les résultats. N'ayant pas un tel code sous la main, je rappellerai simplement que nous avons réalisés moult tests de précision multipôle précédemment qui nous donnent toute confiance dans ces résultats.

3.7.2 Airbus A318

On se propose de calculer les courants surfaciques apparaissant sur un avion de ligne illuminé par une onde électromagnétique à 380 MHz (cf. table 3.31). L'envergure de l'appareil est de 43 longueurs d'onde. Les paramètres du calcul sont similaires à ceux du tableau 3.29, à ceci près que le calcul multipôle est désormais in-core.

Objet :	
Nom	Airbus A318
Nombre d'inconnues	213.084
Envergure	43λ
Calcul :	
Second membre	1 onde incidente
Post-traitement	Courant
Machine	PC 866 MHz

TAB. 3.31 – *Présentation du calcul : A318*

On donne dans le tableau 3.32 les performances obtenues pour ce calcul. La convergence a été obtenue en 81 itérations.

La figure 3.28 est une illustration des visualisations que l'on peut éditer à l'issue d'un tel calcul. On y a tracé, en échelle logarithmique, la norme des courants surfaciques. On peut voir qu'ils se concentrent sur les bords d'attaque de l'empennage.

En l'occurrence, malheureusement, il ne s'agit pas du maillage à 213.084 inconnues, mais d'un maillage plus petit à 23.676 inconnues, illuminé par une onde plane de fréquence 125 MHz. On touche là un des problèmes liés à l'utilisation de la méthode multipôle : le saut

Temps :	
Démarrage	45 min
Produit multipôle	105 s (par itération)
Total	3,3 h
Mémoire :	
RAM	159 Mo
Disque	1156 Mo

TAB. 3.32 – *A318 : performances*

FIG. 3.28 – *Courant surfacique sur un Airbus*

qu'elle permet de réaliser en terme de nombre d'inconnues est tel qu'il impose de réviser tous les modules de prétraitement et de post-traitement. Ici, pour notre calcul à plus de 200.000 inconnues, le logiciel calculant les courants surfaciques a tourné pendant 5 jours sur une station Dec Alpha, et le logiciel graphique n'a rien pu tracer par manque de mémoire. C'est pourquoi nous nous sommes replié sur un maillage à 23.676 inconnues.

3.7.3 Chasseur furtif à 1 GHz

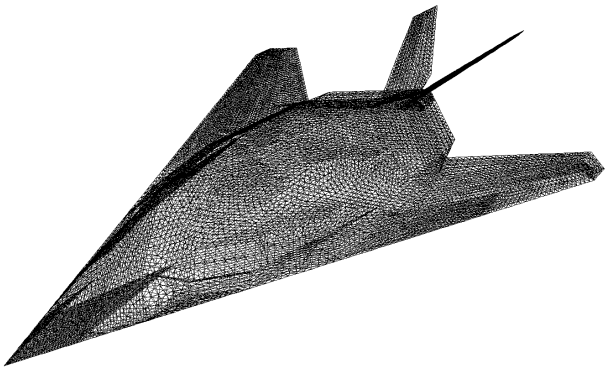


FIG. 3.29 – *Pseudo F117*

Dans ce dernier cas test, nous allons regarder l'action d'une onde plane à 1 GHz sur un avion de chasse type F117 (cf. figure 3.29). L'objet représenté n'est bien sûr pas un authentique F117, mais une simple imitation (d'où le nom de « pseudo F117 »). Traiter ainsi un avion entier à 1 Ghz était il y a peu le symbole du calcul irréalisable, à moins d'utiliser des calculateurs gigantesques. Avec l'avènement de la méthode multipôle, nous allons voir que ce type de calcul peut être entièrement fait sur une seule station de travail.

La principale contrainte dans le choix de la machine utilisée est de devoir gérer des fichiers de plus de deux gigaoctets (limite supérieure pour de nombreux systèmes d'exploitation ou système de fichiers). En effet, la matrice des interactions proches pèse à elle seule près de 3,5 Go (soit environ 450 éléments par lignes). Nous avons donc utilisé un processeur de IBM SP3, mais des temps équivalents voire meilleurs auraient pu être obtenus sur une SGI octane2, ou une DEC alpha.

Objet :	
Nom	pseudo F117
Nombre d'inconnues	981.000
Diamètre	63λ
Calcul :	
Second membre	1 onde incidente
Post-traitement	SER, 1800 directions
Machine	SP3 (1 processeur)

TAB. 3.33 – *Présentation du calcul : Pseudo F117*

Le cas de calcul est présenté dans le tableau 3.33. L'avion est illuminé par une onde radar arrivant de face avec une inclinaison de 5 degrés en dessous de l'horizontale, et on regarde le

champ diffracté par l'engin dans ce même plan horizontal.

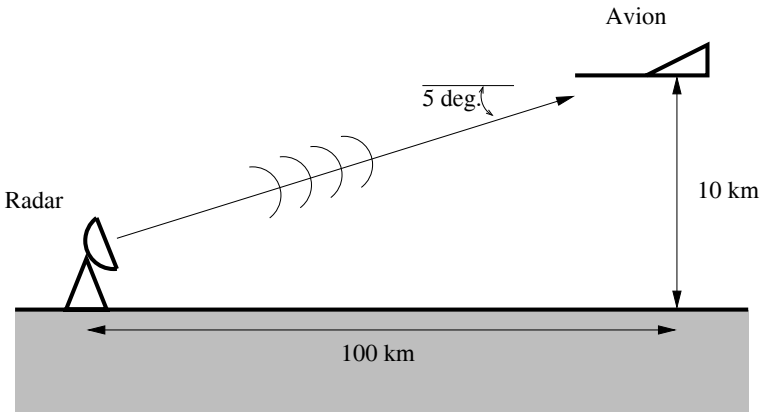


FIG. 3.30 – *F117 : cas traité*

Cela correspond à la configuration représentée sur le schéma 3.30 : l'avion volant à 10 km d'altitude est illuminé par un radar se trouvant face à lui à une distance de 100 km.

Temps :	
Démarrage	297 min
Produit multipôle	425,5 s (par itération)
Champ lointain	104 min
Total	19 h
Mémoire :	
RAM	208 Mo
Disque	6 Go

TAB. 3.34 – *F117 : résultat*

On donne dans le tableau 3.34 les performances obtenues pour ce calcul. Nous avons utilisés ici le mode out-of-core avec 4 groupes, et une FMM à 10 niveaux, le niveau plafond étant le niveau 5. Pour le reste, comme pour les deux autres cas de calcul, on a résolu la CFIE avec un GMRES (restart=20, résidu normalisé=10^{-2}). La convergence à 1 pour-cent n'a pas été obtenue au bout de 100 itérations, le résidu normalisé final est donc de 2,5 % seulement. Bien sûr, un préconditionneur améliorerait les choses ici.

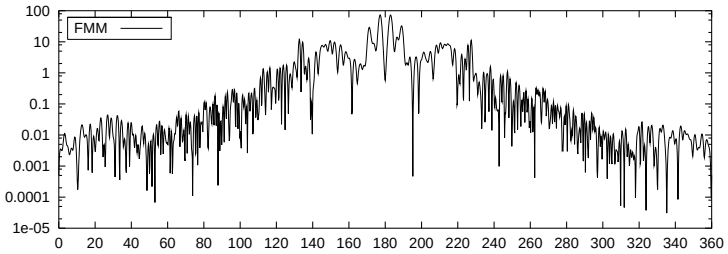


FIG. 3.31 – *SER du pseudo-F117*

La courbe de SER issue de ce calcul est tracée figure 3.31. La direction $\phi = 0$ correspond à l'avant de l'appareil, et $\phi = 180$ pointe vers l'arrière de l'avion. On retrouve bien sûr sur la courbe la symétrie du problème. Le maximum d'énergie est envoyé vers l'arrière de l'engin, ce qui est bien l'objectif recherché.

Conclusion

Nous avons présenté dans ce chapitre une étude se voulant exhaustive de notre code multipôle rapide. Dans un premier temps, nous avons montré que la FMM pouvait avantageusement se substituer aux méthodes dites classiques pour résoudre certains problèmes. Ensuite, nous avons présenté l'ensemble des paramètres ajustables de la méthode multipôle rapide. En particulier, nous avons montré l'intérêt de chercher à optimiser la taille des feuilles, le nombre de pôles, le nombre de composantes, la précision machine. Souvent, il est possible d'obtenir sans perte de précision des gains conséquents en terme de temps ou de mémoire. Il faut souligner que les valeurs numériques obtenues ici dépendent beaucoup de la machine et du programme utilisés. Au-delà de son intérêt intrinsèque, cette étude a permis de souligner la richesse fonctionnelle de la méthode. La possibilité de faire cohabiter dans un même solveur itératif des produits multipôles de précision et de rapidité variables offre des perspectives intéressantes. Enfin, nous avons présenté quelques cas de calcul de nature industrielle réalisés sur stations de travail. Ces calculs auraient été irréalisables sans méthode multipôle, et ce même sur des machines parallèles.

Au final, il apparaît que l'utilisation de la méthode multipôle rapide dans un code d'électromagnétisme n'est pas une simple amélioration. C'est au contraire une « révolution » qui permet de traiter sur de simples stations de travail en quelques heures des calculs qui auraient pris plusieurs jours sur des supercalculateurs d'autrefois (en informatique, autrefois signifie « il y a trois mois ») et ce sans détérioration sensible du résultat. Naturellement, on souhaiterait voir ce que cette méthode est capable de faire sur machines parallèles. C'est précisément l'objet de la seconde partie de ce document.

Deuxième partie

La Méthode Multipôle Rapide : Version parallèle

Chapitre 4

Implémentation parallèle

Sommaire

Introduction **165**
4.1 Méthode de parallélisation **165**
4.2 Répartition des cellules d'un octree **166**
4.3 Nouvelles tâches et nouvelles sous-listes **177**
4.4 Autres aspects de la parallélisation **189**
Conclusion **191**

Introduction

Comme on a pu le voir aux chapitres précédents, la méthode multipôle rapide – même séquentielle – représente un considérable bond en avant en termes de performance par rapport aux méthodes non-multipôles directes ou itératives. La FMM permet de traiter des cas de calculs tout simplement inaccessibles par le passé. Cependant, pour pouvoir envisager de traiter des cas encore plus importants, avec un plus grand nombre de seconds membres, ou encore des problèmes pour lesquels les matrices sont mal conditionnées, l'utilisation de machines parallèles pour accélérer davantage les produits matrice-vecteur multipôles semble incontournable. En outre, dans la mesure où la FMM est conduite à s'intégrer dans des codes le plus souvent parallélisés, une FMM elle-même parallèle permettrait de conserver le bénéfice du travail déjà effectué. Dans ce chapitre, nous nous proposons donc d'exposer les techniques utilisées pour réaliser la parallélisation de la FMM. Après quelques généralités sur le sujet, nous présenterons la méthode utilisée pour distribuer les octrees, puis nous montrerons comment répartir le calcul d'un produit multipôle. La dernière section est consacrée à la parallélisation du reste du code.

4.1 Méthode de parallélisation

4.1.1 Options techniques

Lorsqu'il s'agit de paralléliser un code de calcul, plusieurs options s'offrent à nous. A chacune de ces approches correspond un type de programmation, un type de machine, des avantages et des inconvénients.

La première possibilité consisterait à utiliser des langages permettant une parallélisation automatique (comme le *high performance fortran* de Portland Group) ou des bibliothèques de calcul scientifique déjà parallélisées (telles que PBLAS et ScaLAPACK disponibles via NETLIB). Ce serait de loin l'approche la plus simple si elle était envisageable. Malheureusement, les algorithmes que nous cherchons à paralléliser sont beaucoup trop complexes pour pouvoir tirer partie de ces méthodes de haut niveau. Celles-ci sont adaptées aux traitements simples, comme des produits matrice-vecteur ou des produits scalaires. Elles ne conviennent pas (du moins pas encore) à des calculs élaborés comme la méthode multipôle rapide.

Une autre approche possible serait d'utiliser un compilateur compatible avec la norme openMP. Cette dernière définie des directives de compilation permettant de générer un exécutable multitâche, capable de fonctionner en parallèle sur toute machine à mémoire partagée. La programmation est alors relativement simple. Par contre, le code obtenu ne peut pas fonctionner en parallèle sur les réseaux de stations, les fermes de PC (très à la mode) et les machines à mémoire distribuée. De plus, cette approche étant encore récente (initiée en octobre 1997), tous les compilateurs ne sont pas compatibles (et en particulier, pas les compilateurs GNU).

La troisième solution (que nous avons choisie) consiste à effectuer une parallélisation par échange de messages à l'aide d'une bibliothèque standard comme MPI ou PVM. Dans ce cas, le développeur doit écrire quasi explicitement toutes les commandes aboutissant au partage des données et du travail entre les processeurs. C'est certes assez fastidieux, mais parfois incontournable. L'avantage principal de cette approche est de produire un code pouvant tourner sur tout type de machine parallèle, à mémoire distribuée ou partagée, homogène ou hétérogène (c'est-à-dire constituée de nœud différents). Nous avons choisi d'utiliser MPI, qui est un standard de fait pour ce type d'application. C'est une norme datant de mai 1994 qui est fiable, simple, performante, et de plus en plus répandue. Il en existe même des implémentations libres (MPICH, MPI-LAN).

4.1.2 Principe de la méthode

Nous allons présenter les grandes lignes de la méthode utilisée pour paralléliser le calcul multipôle rapide. Le travail se décompose en trois phases :

1. *Séparer les données* : on a vu dans le cas séquentiel que l'occupation de la mémoire vive était principalement due aux fonctions de radiation. Nous allons donc répartir l'octree et ses cellules entre les processeurs.
2. *Créer les listes de messages* : en tenant compte des données disponibles sur chaque nœuds et de celles dont on a besoin pour continuer le calcul, nous créons la liste des messages à échanger.
3. *Optimiser ces listes* : afin de diminuer le temps consacré aux communications, nous réordonnons les envois des messages.

Chacune de ces trois étapes va être détaillée dans les sections qui suivent.

4.2 Répartition des cellules d'un octree

4.2.1 Simplification du problème

Le point de départ de notre parallélisation est le principe suivant : chaque cellule de l'octree est affectée à un unique processeur. Nous dirons qu'une cellule est « sur » un processeur. Ce

dernier aura la charge de calculer les deux fonctions de radiation associées à cette cellule. Si des fonctions de radiation hébergées par un autre processeur sont requises pour ce calcul, elles devront avoir été envoyées au préalable. Nous verrons à la section 4.3.3 que pour certaines étapes de transfert nous ferons entorse à ce principe, mais cela ne sera qu'à titre provisoire.

La parallélisation d'un algorithme a en général un coût non nul, qui empêche un calcul sur p processeurs d'aller effectivement p fois plus vite qu'un calcul monoprocesseur. Ce coût a deux origines :

1. Les communications : il s'agit du temps consacré à envoyer et recevoir des données entre processeurs.
2. Les barrières : les processeurs qui finissent leur travail en premier doivent attendre les retardataires (au cours d'une opération qui s'appelle barrière), ce qui induit une perte de temps.

Dans cette première section, nous allons étudier la distribution d'un octree entre plusieurs processeurs. Cette répartition aura une influence directe :

- D'une part sur la charge de travail qui sera attribuée à chaque processeur (et la perte de temps liée aux barrières) ;
- D'autre part sur la quantité de données échangées au cours d'un produit multipôle (et la perte de temps liée aux communications).

On le voit, une FMM parallèle efficace passe obligatoirement par une distribution réussie de l'octree.

A priori, le problème à résoudre est très complexe. Nous allons faire une première hypothèse simplificatrice : on décide de distribuer chaque niveau de l'octree indépendamment des autres. C'est un choix qui est dicté par la manière dont fonctionne l'algorithme multipôle, niveau par niveau. On aurait aussi pu affecter les niveaux bas à certains processeurs, et les niveaux hauts à d'autres, mais c'eût été sans doute beaucoup moins naturel (et efficace). Nous laissons à d'autres le soin d'explorer cette voie.

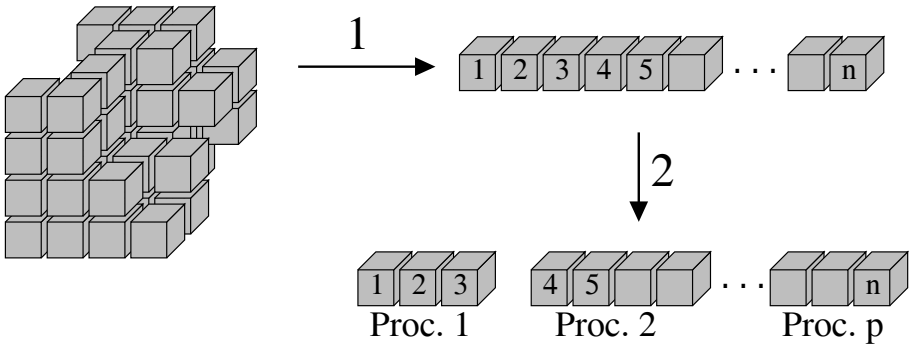


FIG. 4.1 – *Parallélisation d'un niveau d'octree en deux étapes*

Pour répartir un niveau donné entre p processeurs, on procède en deux temps (représenté sur la figure 4.1) :

1. Dans un premier temps, on numérote les cellules de ce niveau de 1 à N. On passe ainsi d'un problème tridimensionnel à un problème monodimensionnel.
2. Dans un deuxième temps, on répartit les cellules entre les processeurs en affectant les cellules 1 à N_1 au processeur 1, $N_1 + 1$ à N_2 au processeur 2, ..., $N_{p-1} + 1$ à $N_p = N$

au processeur p.

La seconde partie de ce travail déterminera la répartition du calcul FMM entre les processeurs, tandis que la première induira la quantité de données échangées au cours de chaque produit multipôle. En procédant ainsi, on peut optimiser séparément les coûts liés aux barrières et aux communications.

4.2.2 Numérotation d'un niveau

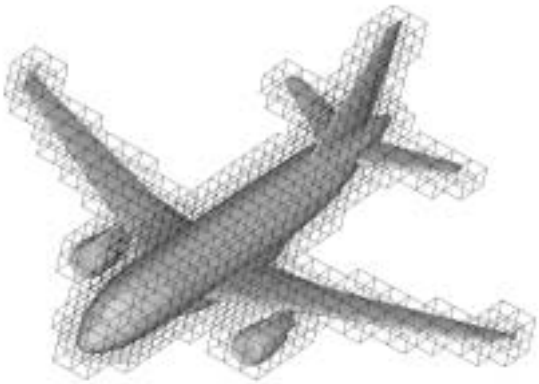


FIG. 4.2 – *Cinquième niveau de l'octree associé à un airbus A318*

Pour fixer les idées, regardons le cas représenté sur la figure 4.2. On y a représenté un maillage d'airbus A318 entouré par le cinquième niveau de l'octree construit sur cet objet. A ce stade du découpage, l'arbre comporte 584 cellules. Rappelons que seules les cellules ayant une intersection non-vide avec la frontière de l'objet sont conservées. L'objectif est de numéroter ces cellules de 1 à 584. Cette numérotation va avoir une influence directe sur la quantité de données échangées entre les processeurs au cours du calcul multipôle. Si deux cellules géométriquement proches obtiennent des numéros éloignés, elles risquent d'être gérées par deux processeurs différents et leur (éventuelle) interaction devra alors être précédée de l'envoi d'un message.

Rappelons les différentes opérations de calcul mettant en jeux plusieurs cellules :

- Les interactions proches : elles se produisent entre feuilles voisines.
- Les montées et descentes : elles mettent en jeu deux cellules parent et enfant.
- Les transferts : ils se font entre cellules banlieues (i.e. des cellules non-voisines dont les parents respectifs sont voisins).

Les initialisations et intégrations ne concernent qu'une seule cellule à la fois, et ne génèrent donc pas de communications. Au final, on voit que si une cellule $\mathcal{C}$ est sur un processeur, on aura intérêt à mettre sur ce même processeur le parent, les enfants, les voisins et les banlieues de $\mathcal{C}$. Bien sûr, sauf dans les cas d'objets composés de plusieurs composantes connexes suffisamment éloignées, la parallélisation idéale (sans communication) est impossible, et il va falloir faire des compromis. Nous allons proposer trois numérotations pour tenter de répondre à ce problème.

4.2.2.1 Numérotation cartésienne

La numérotation cartésienne consiste à ordonner les cellules à chaque niveau de l'octree en fonction des coordonnées x puis y puis z de leur centre. On numérote ensuite les cellules

dans l'ordre ainsi obtenu. Une version 2D de cette numérotation est représentée sur la figure 4.3. L'avantage de cette méthode est sa simplicité. L'inconvénient est qu'elle produit une numérotation discontinue symbolisée par les lignes pointillées sur la figure 4.3.

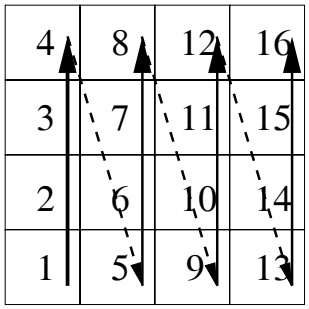


FIG. 4.3 – *Numérotation cartésienne en deux dimensions*

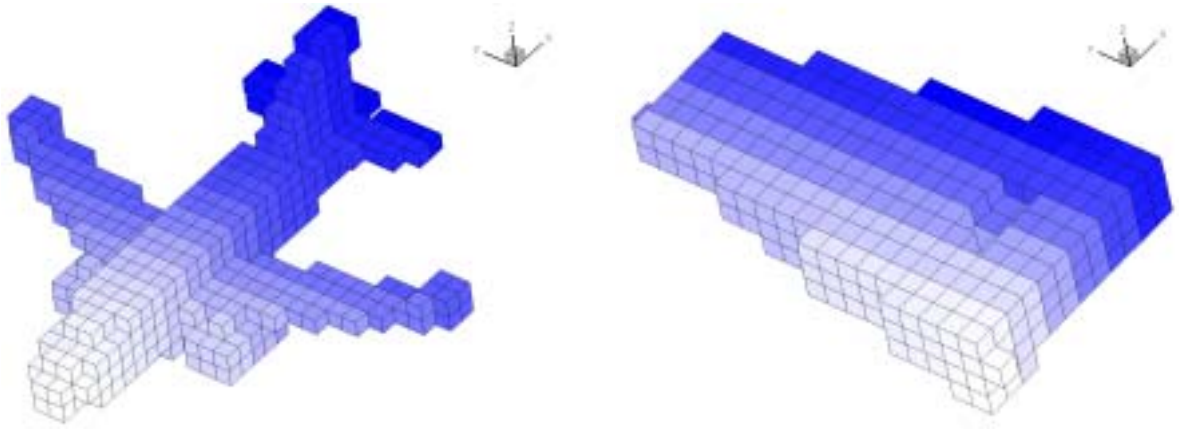
Sur des cas réels tridimensionnels, le résultat de cette numérotation est tracé pour un airbus A318 et un Cetaf sur les figures 4.4a et 4.4b. Sur ces graphiques, les cellules du niveau 5 de l'octree sont tracées sous formes de cubes de couleur (qui masquent les maillages des objets). La cellule portant le numéro 0 est représentée en blanc, la dernière cellule numérotée est teinte en bleu (si vous avez les couleurs, en sombre sinon), la coloration variant progressivement avec la numérotation. On voit clairement la numérotation progresser (et la couleur s'assombrir) selon l'axe des x. Dans le cas d'un objet élancé comme le cetaf, l'axe prioritaire de numérotation (Ox ici) peut influer fortement sur le résultat. C'est pourquoi dans nos tests à venir nous regarderons trois numérotations cartésiennes résultants de tris de cellules selon (x,y,z), (y,z,x) ou (z,x,y). En toute rigueur, on devrait prendre en compte trois autres ordres de tri obtenus en permutant les deux dernières directions ((x,z,y), (y,x,z) et(z,y,x)). En pratique, on peut les négliger car seule la première direction de numérotation influence réellement le résultat final.

Sur les images 4.6, une bonne numérotation va être caractérisée par le fait que des cellules géométriquement proches dans l'espace ont des couleurs proches. Cette caractéristique est déjà très présente sur les figures 4.6a et 4.6b.

4.2.2.2 Numérotation hiérarchique

La numérotation hiérarchique déduit la numérotation du niveau n de celle du niveau supérieur $n-1$. Une représentation de cette idée en 2D est tracée sur la figure 4.5 : à gauche se trouve le niveau $n-1$. A droite, au niveau n, on numérote d'abord les enfants de la cellule 1 du niveau précédent, puis ceux de la cellule 2, et ainsi de suite. Le « motif » de numérotation $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ du niveau $n-1$ est reproduit quatre fois à l'identique au niveau n (huit fois en 3D). L'avantage de cette méthode est qu'au final les enfants ont de grandes chances de se retrouver sur le même processeur que leur parent. Malheureusement, cette numérotation aussi est discontinue.

Sur la figure 4.6, on a représenté le résultat de cette numérotation sur les mêmes cas réels que précédemment. Sur l'airbus, on voit nettement un découpage en quatre parties (avant/arrière et droite/gauche) qui résulte du découpage effectué au niveau 1 de l'octree. Notons qu'en regardant l'appareil de profil, on pourrait aussi distinguer un partage dessus/dessous.



(a) Airbus A318 (b) Cetaf

FIG. 4.4 – *Numérotation cartésienne sur des cas complexes*

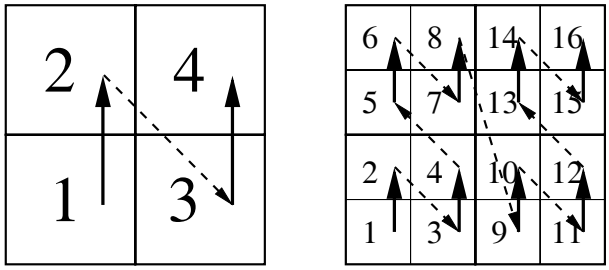
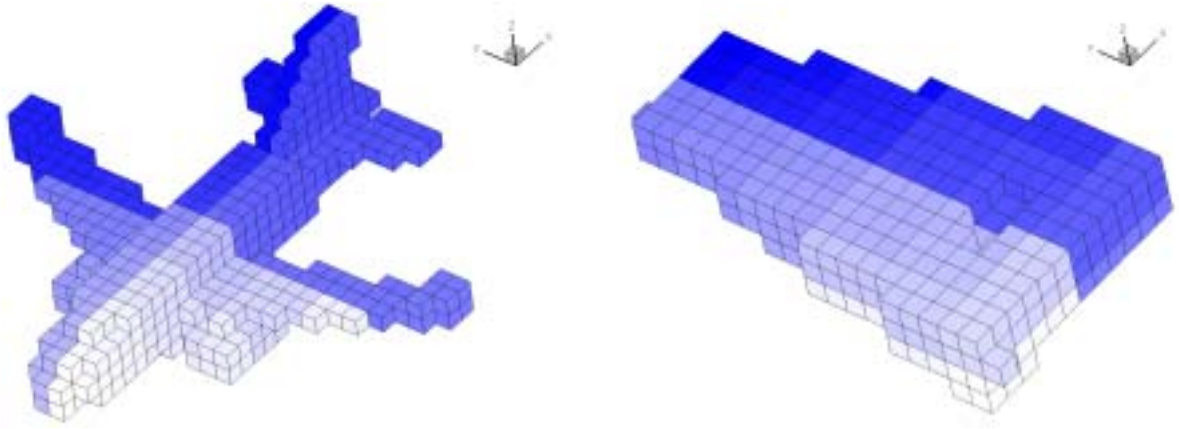


FIG. 4.5 – *Numérotation hiérarchique en deux dimensions*



(a) Airbus A318 (b) Cetaf

FIG. 4.6 – *Numérotation hiérarchique sur des cas complexes*

4.2.2.3 Numérotation Peano

On désigne aujourd'hui par courbe de Peano toute courbe plane continue dont la trajectoire a une aire non nulle, une courbe continue de l'espace dont la trajectoire a un volume non nul ou plus généralement une courbe continue de R^n dont la trajectoire a une mesure n-dimensionnelle non nulle [1]. On va utiliser cette construction mathématique pour numéroter les cellules de notre octree de manière à n'avoir aucune discontinuité. Pour alléger les figures, nous ne noterons plus explicitement les numéros de chaque cellule. La figure 4.7 présente les numérotations des niveaux 1, 2 et 3 de l'octree.

Le motif de numérotation $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ du niveau $n-1$ a un point de départ ou point d'entrée (noté « e » sur la figure) et une arrivée ou point de sortie (noté « s »). Les points d'entrée/sortie des enfants au niveau n sont obtenus à partir de ceux du parent au niveau précédent en suivant les flèches. Le point d'entrée de chaque enfant coïncide avec le point de sortie de l'enfant précédent (ces points sont notés « e/s »). Ainsi, le motif de numérotation $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ du niveau $n-1$ est reproduit au niveau n à l'identique dans les enfants des cellules 2 et 3, et avec une rotation de $\pm\pi/2$ radians dans les enfants des cellules 1 et 4. En reliant entre eux les 4 motifs ainsi reproduits, on obtient une numérotation continue du niveau n (figure 4.7 au centre). En réitérant ce processus, on peut numéroter les niveaux suivants (image de droite). Notons qu'il s'agit d'un type particulier de numérotation hiérarchique.

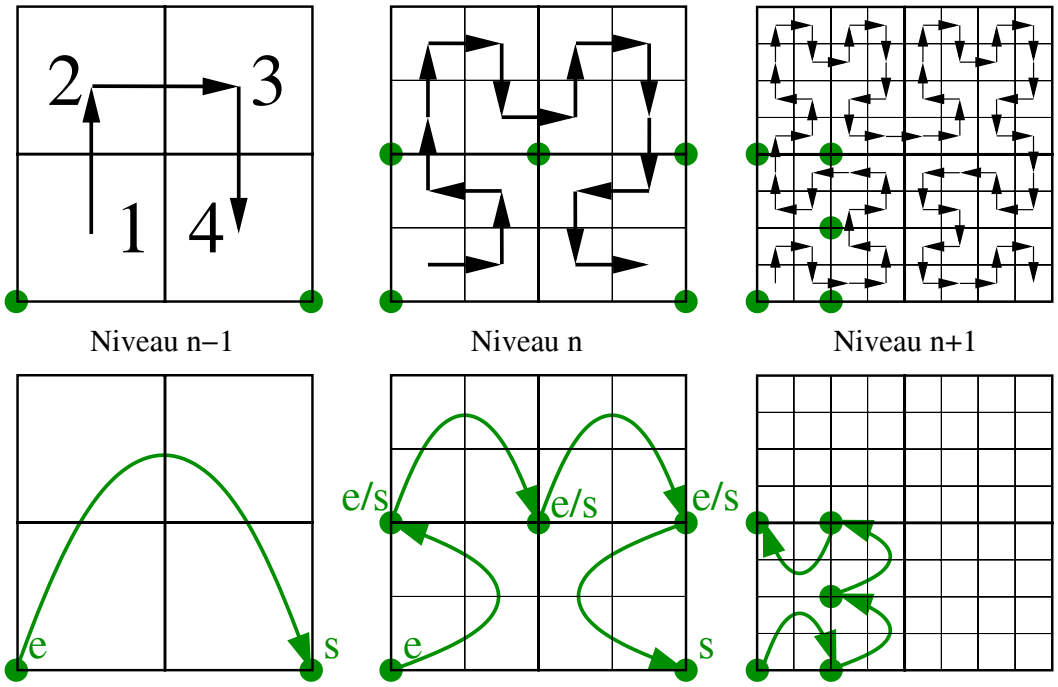


FIG. 4.7 – *Numérotation hiérarchique de type Peano en deux dimensions*

La figure 4.8 utilise le même procédé pour numéroter un cube. Là encore, un motif de base (cube de gauche) est reproduit 8 fois (cube de droite) via des rotations de $\pm\pi/2$ autour des axes du repère. Le concept de points d'entrée et de sortie persiste, et permet de savoir comment reproduire le motif de numérotation du niveau précédent.

1. définition tirée de http://perso.club-internet.fr/rferreol/encyclopedie/introduction.shtml

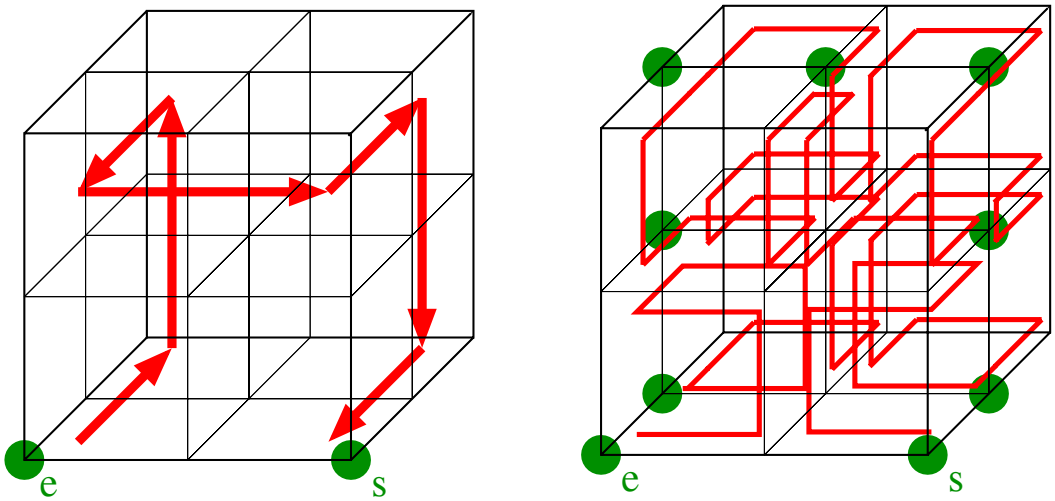
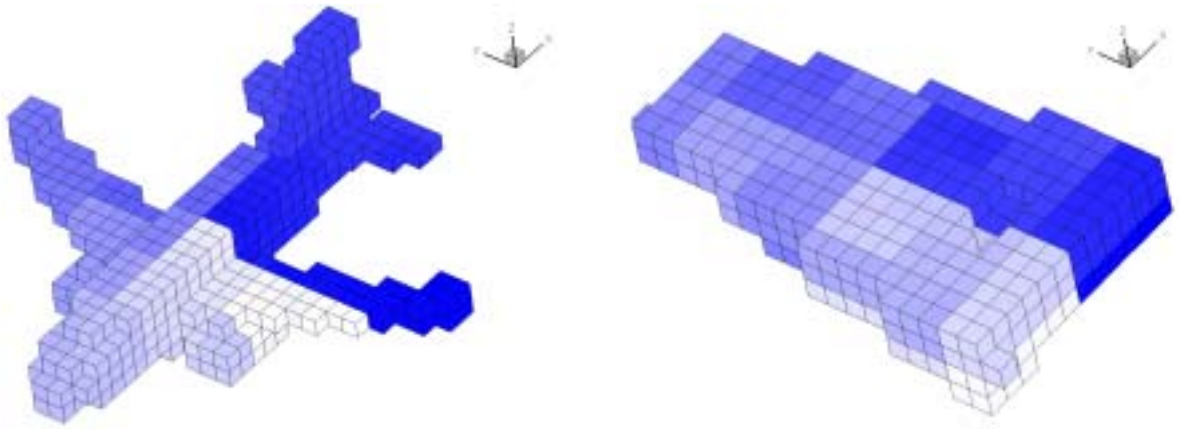


FIG. 4.8 – *Numérotation hiérarchique de type Peano en trois dimensions*

Le résultat de cette numérotation sur un airbus et un cetaf sont tracés sur la figure 4.9. Les avantages de cette numérotation sont d'une part qu'elle est hiérarchique (les enfants ont plus de chances de se retrouver sur le même processeur que leur parent), d'autre part qu'elle est continue (du moins dans le cas d'un cube, puisque la suppression des cellules vides dans les cas réels conduit de toute façon à des discontinuités). L'inconvénient majeur est la complexité de mise au point, surtout dans le cas tridimensionnel.



(a) Airbus A318 (b) Cetaf

FIG. 4.9 – *Numérotation Peano sur des cas complexes*

4.2.2.4 Comparaison de ces numérotations

Nous allons tester ces cinq algorithmes de numérotations :

1. Peano ;
2. Hiérarchique ;
3. Cartésienne selon (Ox) ;
4. Cartésienne selon (Oy) ;
5. Cartésienne selon (Oz).

Pour différents maillages, et différents nombres de processeurs, on fait afficher par le programme la quantité totale de données qui seront échangées par le code FMM au cours d'un produit matrice-vecteur multipôle. Cette quantité est ensuite divisée par le nombre de processeurs. Elle s'exprime en méga-octets. Elle ne dépend pas de la machine de test, car les nombres réels sont stockés sur 4 octets sur toutes les architectures utilisées (IBM SP3, SGI, PC x86).

Nous allons utiliser les quatre maillages présentés en détail à la section 5.2. Il s'agit donc d'un airbus à 1,2 millions d'inconnues, une sphère à 2,6 millions d'inconnues, un pseudo F117 à 3,9 millions d'inconnues et un cetaf à 4,9 millions d'inconnues. La machine utilisée est la ferme de PC de l'INRIA. Le nombre de processeurs a pris les valeurs suivantes : 2, 4, 8, 12, 16, 24 ,32.

Maillage Taille de l'octree (en Mo)	Airbus1M 706	sphère2M 975	furtif4M 1129	cetaf5M 2332
Peano	70	65	87	186
Hiérarchique	71	64	86	178
Cartésienne (Ox)	70	98	158	213
Cartésienne (Oy)	90	98	99	184
Cartésienne (Oz)	107	98	85	279

TAB. 4.1 – *Quantité de données échangées (en Mo) en fonction de la numérotation*

Les résultats numériques pour les tests à 16 processeurs sont donnés dans le tableau 4.1 à titre indicatif. On y a également rappelé la taille totale de l'octree (hors parallélisation) pour chaque cas de calcul. Les résultats complets sont présentés graphiquement : la taille totale des messages (divisée par le nombre de processeurs et exprimée en méga-octets) est tracée en fonction du nombre de processeurs, à raison d'une figure par objet, et d'une courbe par algorithme de numérotation.

Pour le cas Airbus1M, les résultats se trouvent sur la figure 4.10. Les numérotations cartésiennes selon (Oy) et (Oz) sont les moins performantes. Les trois autres sont équivalentes, avec un léger avantage pour l'algorithme cartésien selon (Ox) (qui est l'axe de l'avion) pour les petits nombres de processeurs (12 et moins).

Pour le cas sphere2M, les résultats se trouvent sur la figure 4.11. Par symétrie, les trois numérotations cartésiennes donnent les mêmes résultats, assez mauvais. Les algorithmes hiérarchiques et Peano l'emportent nettement, avec des résultats équivalents.

Pour le cas furtif4M, les résultats se trouvent sur la figure 4.12. Pour 12 processeurs et moins, la numérotation cartésienne selon (Oz) (c'est-à-dire dans l'axe de l'appareil) est la meilleure. Au-delà de 16 processeurs, les numérotations Peano et hiérarchiques sont légèrement plus efficaces. Il est intéressant de constater que la numérotation cartésienne selon (Ox), i.e. dans une direction perpendiculaire au plan de l'avion, donne des résultats uniformément

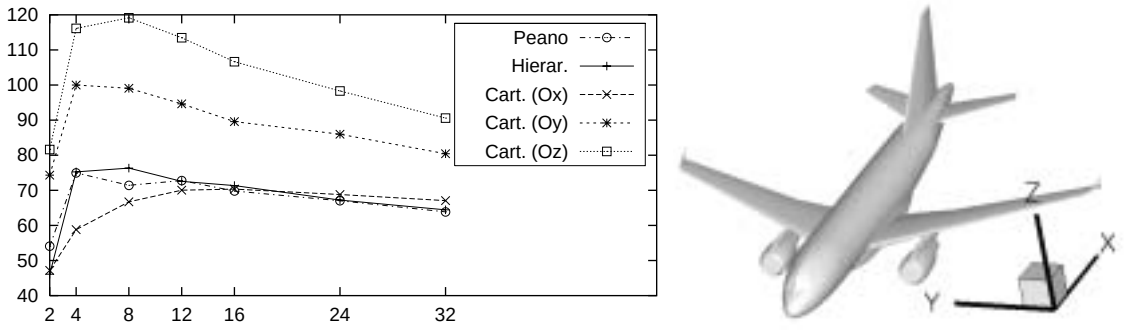


FIG. 4.10 – *Choix de la numérotation : airbus à 1,2 millions d'inconnues*

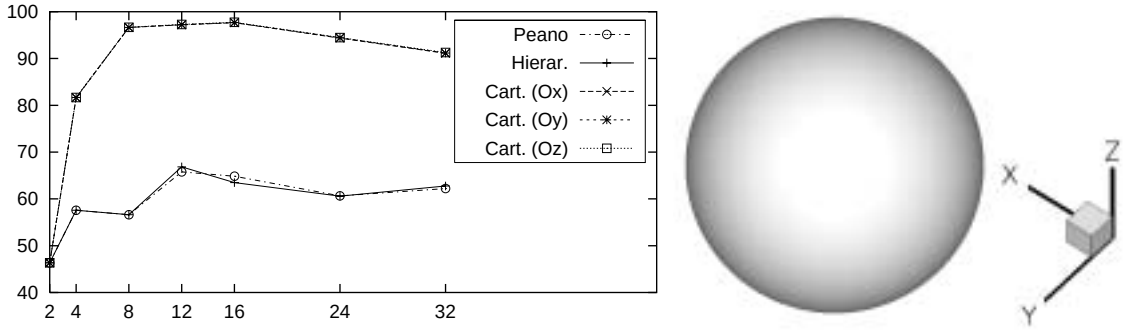


FIG. 4.11 – *Choix de la numérotation : sphère à 2,6 millions d'inconnues*

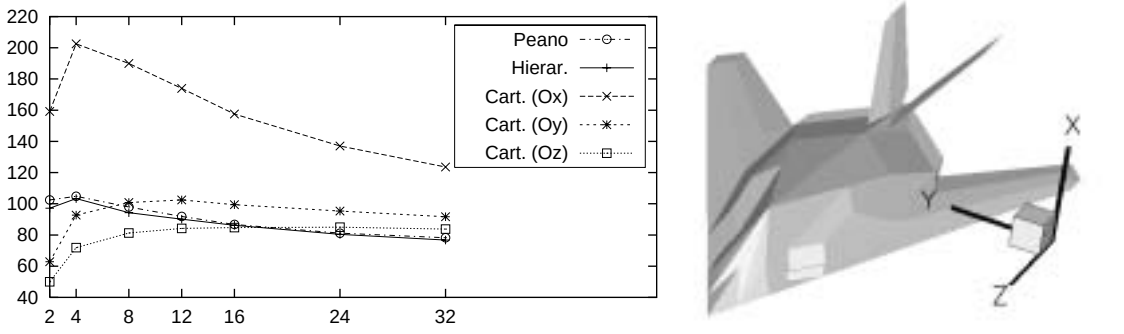


FIG. 4.12 – *Choix de la numérotation : F117 à 3,9 millions d'inconnues*

mauvais (ce qui est assez similaire à ce qu'on observe sur l'Airbus1M pour la numérotation cartésienne selon (Oz)).

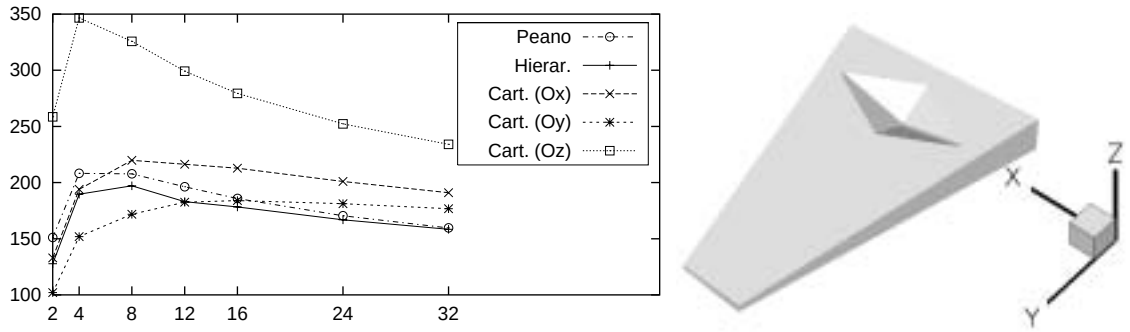


FIG. 4.13 – *Choix de la numérotation : cetaf à 4,9 millions d'inconnues*

Enfin, pour le cas cetaf5M, les résultats se trouvent sur la figure 4.13. Comme dans le cas du pseudo F117, la numérotation cartésienne selon un axe perpendiculaire au plan de l'objet – (Oz) ici – donne les plus mauvais scores. Pour 8 processeurs et moins, la numérotation cartésienne selon (Oy) – i.e. selon la plus grande direction de l'objet – est la meilleure. Avec 12 processeurs et plus, les numérotations de type hiérarchique et Peano dominent.

4.2.2.5 Conclusion

Pour les numérotations cartésiennes, le bilan est mitigé. Si l'objet a une forme élancée dans une certaine direction, alors pour de petits nombres de processeurs la numérotation selon cette axe donne de très bons résultats. Au contraire, dans le cas d'un objet « plan », il est déconseillé de numéroter selon un axe perpendiculaire à ce plan. Globalement, la numérotation hiérarchique est la meilleure. Ses résultats sont uniformément bons, surtout pour les grands nombres de processeurs (qui seront bien sûr les cas les plus critiques). Enfin, la numérotation Peano ne se distingue pas suffisamment de la précédente pour justifier sa complexité accrue. En particulier, le fait qu'elle soit continue semble n'apporter aucun gain (mais il est vrai que de toute façon elle devient forcément discontinue au moment de supprimer les cubes vides).

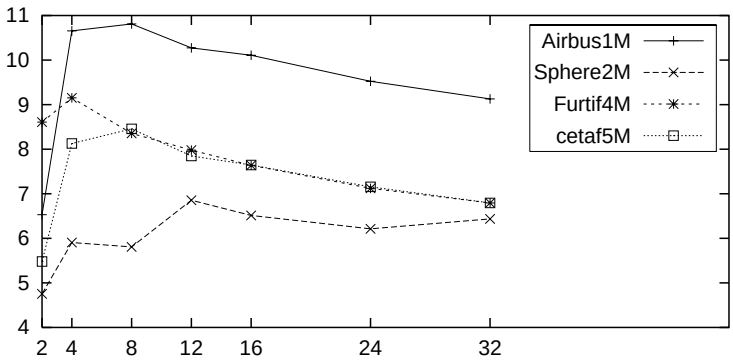


FIG. 4.14 – *Numérotation hiérarchique : quantité de données échangées*

La numérotation hiérarchique étant donc adoptée, nous allons regarder pour finir quelle est la quantité de données échangées par processeur en pourcentage de la taille totale de l'octree (issue du tableau 4.1). Ce pourcentage est tracé sur la figure 4.14 en fonction du nombre de processeurs. On voit que selon le maillage et le nombre de processeurs, chaque processeur envoie (et reçoit) l'équivalent de 5 à 11 % de la taille totale de l'octree. Ce pourcentage est d'autant plus grand que l'objet est fin et élancé.

4.2.3 Répartition

Les cellules étant maintenant numérotées à chaque niveau de l'arbre, on souhaite maintenant les répartir entre les processeurs conformément au processus illustré sur la figure 4.1. Un processeur qui gère une cellule $\mathcal{C}$ a en charge le calcul des deux fonctions de radiations $\mathcal{F}_\mathcal{C}$ et $\mathcal{G}_\mathcal{C}$ définies pour cette cellule. Notons que cela résulte d'un choix de notre part et n'est pas du tout obligatoire. Partant de là, on peut facilement calculer pour chaque cellule la quantité de travail associée.

- Pour calculer $\mathcal{F}_\mathcal{C}$: Si $\mathcal{C}$ est une feuille, une *initialisation*. Sinon, autant de *montées* que $\mathcal{C}$ a d'enfants.
- Pour calculer $\mathcal{G}_\mathcal{C}$: Si $\mathcal{C}$ est au niveau plafond, autant de *transferts* qu'il y a de non-voisins. Sinon, une *descente* puis autant de *transferts* qu'il y a de banlieues. Enfin, si $\mathcal{C}$ est une feuille, une *intégration*.

Le calcul effectif de la charge de travail que cela représente peut être fait de deux manières. La première méthode consiste à compter le nombre d'additions et de multiplications pour chaque opération élémentaire (comme à la section 1.3.2.3). La seconde approche consiste à mesurer le temps CPU effectivement pris par chacune de ces opérations au travers de petits tests préalables. La première possibilité est plus simple, la seconde est plus précise et plus réaliste. Une fois que chaque cellule dispose d'une charge, la répartition des cellules entre les processeurs se fait aussi équitablement que possible, à chaque niveau. Notons que cela assure un bonne répartition du temps de calcul, mais pas forcément des communications.

Mais tout cela reste théorique, puisqu'à ce jour dans notre logiciel les cellules se voient toutes attribuer la même charge égale à 1. La distribution de l'arbre donne donc un nombre quasiment égal de cellules à chaque processeur. Pour autant, comme nous le verrons plus loin, cela ne nuit pas trop aux performances et le déséquilibre de la charge est limité. Gardons simplement à l'esprit que nous avons là une petite marge de progression.

En revanche, il est une optimisation très simple à mettre en œuvre qui consiste à placer d'office les enfants d'une cellule sur le même processeur que le parent dès lors que ce parent se trouve en dessous d'un niveau donné appelé « niveau de séparation ». L'avantage est de supprimer totalement les communications pendant les phases de montées/descentes à ce niveau. On y gagne donc en temps de communication mais – surtout – on supprime une synchronisation imposée à tous les processeurs. L'inconvénient est que la répartition des cellules entre les processeurs au niveau du parent impose la répartition au niveau des enfants. Du coup on risque de détériorer la répartition de la charge. En pratique, si le niveau de séparation est proche du niveau des feuilles (2 ou 3 niveaux d'écart), on gagne quelques pour-cents sur un produit matrice-vecteur multipôle. Ce paramètre fera l'objet d'une étude à la section 5.3.2.

4.3 Nouvelles tâches et nouvelles sous-listes

En reprenant les définitions et la syntaxe de la section 2.1, nous allons maintenant présenter les tâches et les listes de tâches liées à la parallélisation de notre méthode multipôle rapide. Dans un premier temps, nous regarderons la parallélisation par cellule, basée sur la distribution de l'octree détaillée précédemment. Puis nous détaillerons la méthode utilisée pour optimiser les communications qui en découlent. Enfin nous introduirons une modification à cette approche appelée parallélisation par directions qui s'appliquera essentiellement aux plus hauts niveaux de l'arbre.

4.3.1 Parallélisation par cellule

4.3.1.1 Nouvelle tâche

Dans un premier temps, on s'en tient strictement à notre principe initial : un processeur qui gère une cellule $\mathcal{C}$ réalise tous les calculs nécessaires à l'obtention des fonctions de radiation $\mathcal{F}_\mathcal{C}$ et $\mathcal{G}_\mathcal{C}$. On notera désormais $n_\mathcal{C}$ le numéro du processeur gérant la cellule $\mathcal{C}$. On introduit une nouvelle tâche définie de la manière suivante :

PackF($\mathcal{T}_\mathcal{C}$,n_{dest})

Paramètres :	Une cellule $\mathcal{C}$, un type de fonction de radiation ($\mathcal{T} = \mathcal{F}$ ou $\mathcal{G}$), un numéro de processeur destinataire n_{dest}.
Action :	Prépare l'envoi de la fonction $\mathcal{T}_\mathcal{C}$ au processeur n_{dest}.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{T}_\mathcal{C}}$ sur le processeur destinataire n_{dest}.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{T}_\mathcal{C}}$ sur le processeur source $n_\mathcal{C}$.

Seuls les processeurs $n_\mathcal{C}$ et n_{dest} sont concernés par cette tâche. Cette tâche en elle-même ne réalise pas de communications. Elle prépare juste l'envoi d'une fonction de radiation $\mathcal{T}_\mathcal{C}$ vers un processeur n_{dest}. Nous verrons à la section suivante 4.3.2 comment sont effectivement réalisés ces échanges. Notons que c'est en comptant le nombre de `packF` réalisés que l'on a obtenu les quantités de données échangées tracées sur les figures 4.10 à 4.13.

4.3.1.2 Nouvelles sous-listes

A l'instar du cas séquentiel et de la section 2.1.3, nous allons présenter la création des sous-listes de communication. Les phases d'initialisation et d'intégration ne sont pas concernées : elles ne nécessitent pas de communication puisqu'elles n'agissent que sur une seule fonction de radiation. Pour les autres phases, à chaque sous-liste de calcul $\mathcal{L}_i$ avec $1 \leq i \leq 3(N - N_{plaf})$ on va associer une sous-liste de communication notée $\mathcal{L}'_i$. Au cours d'un produit multipôle, ces sous-listes sont exécutées dans l'ordre suivant :

$$\mathcal{L}'_1,\ \mathcal{L}_1,\ \mathcal{L}'_2,\ \mathcal{L}_2,\ \ldots,\ \mathcal{L}'_{3(N-N_{plaf})},\ \mathcal{L}_{3(N-N_{plaf})}$$

Chaque phase de communication $\mathcal{L}'_i$ précède la phase de calcul associée $\mathcal{L}_i$. Nous allons détailler dans la suite le contenu des nouvelles listes $\mathcal{L}'_i$, mais précisons tout d'abord le cas des listes $\mathcal{L}_i$.

4.3.1.2.1 Sous-listes des calculs Les sous-listes créées à la section 2.1.3 vont devoir être modifiées pour tenir compte du parallélisme. En l'occurrence, il suffit de vérifier que la fonction de radiation que l'on s'apprête à calculer est bien locale. Par exemple, l'algorithme 4.1 crée la sous-liste d'initialisation.

```
L_1 ← ∅
for all C cellule du niveau N − 1 do
  if (C est locale) then
    L_1 = L_1 ∪ initialisation(C)
  end if
end for
```

Algorithme 4.1: Initialisation en parallèle

4.3.1.2.2 Sous-listes des communications avant transfert Pour $l = N-1, \ldots, N_{plaf}$, la sous-liste des transferts au niveau l porte le numéro $i = 2(N-l)$. Dans l'algorithme 4.2, on a distingué le niveau plafond (où les transferts se font entre cellules non-voisines) et les autres niveaux (où les transferts se font entre cellules banlieues).

```
L'_i ← ∅
if (l = N_plaf) then
  for all C, C' cellules du niveau l do
    if (C est locale) and (C' n'est pas locale) then
      if (C non-voisine de C') then
        L'_i = L'_i ∪ packF(F_C, n_C')
      end if
    end if
  end for
else
  for all C, C' cellules du niveau l do
    if (C est locale) and (C' n'est pas locale) then
      if (C banlieue de C') then
        L'_i = L'_i ∪ packF(F_C, n_C')
      end if
    end if
  end for
end if
```

Algorithme 4.2: Communications avant transfert

Pour chaque transfert d'une cellule $\mathcal{C}$ sur une cellule $\mathcal{C}'$, si la première est locale et pas la seconde, on envoie auparavant la fonction de radiation $\mathcal{F}_{\mathcal{C}}$ au processeur $n_{\mathcal{C}'}$.

4.3.1.2.3 Sous-listes des communications avant montée Pour $l = N-1, \ldots, N_{plaf}+1$, la sous-liste des montées du niveau l au niveau $l-1$ porte le numéro $i = 2(N-l)+1$. Elle est construite par l'algorithme 4.3.

$\mathcal{L}'_i \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau l **do**
  if ($\mathcal{C}$ est locale) **and** ($p(\mathcal{C})$ n'est pas locale) **then**
    $\mathcal{L}'_i = \mathcal{L}'_i \cup$ `packF`$(\mathcal{F}_{\mathcal{C}}, n_{p(\mathcal{C})})$
  end if
end for

Algorithme 4.3: Communications avant montée

Autrement dit, si une cellule est locale mais que son parent ne l'est pas, il faut envoyer $\mathcal{F}_{\mathcal{C}}$ au processeur qui gère le parent. Il se peut que cet envoi ait déjà été fait pour la phase de transfert précédente.

4.3.1.2.4 Sous-listes des communications avant descente Pour $l = N_{plaf}+1, \ldots, N-1$, la sous-liste des descentes du niveau $l-1$ au niveau l porte le numéro $i = 2(N-N_{plaf})+l-N_{plaf}$. Elle se construit en suivant l'algorithme 4.4.

$\mathcal{L}'_i \leftarrow \varnothing$
for all $\mathcal{C}$ cellule du niveau l **do**
  if ($\mathcal{C}$ n'est pas locale) **and** ($p(\mathcal{C})$ est locale) **then**
    $\mathcal{L}'_i = \mathcal{L}'_i \cup$ `packF`$(\mathcal{G}_{p(\mathcal{C})}, n_{\mathcal{C}})$
  end if
end for

Algorithme 4.4: Communications avant descente

Si le parent de $\mathcal{C}$ est local, mais pas $\mathcal{C}$, il faut envoyer $\mathcal{G}_{p(\mathcal{C})}$ au processeur qui gère $\mathcal{C}$ pour que ce dernier puisse calculer $\mathcal{G}_{\mathcal{C}}$.

4.3.2 Optimisation des échanges

4.3.2.1 Présentation

Dans la section précédente, nous avons vu comment créer une liste contenant la définition de tous les envois à réaliser. Nous allons maintenant montrer comment réaliser toutes ces communications le plus efficacement possible. Techniquement, rien ne nous empêche de poster simultanément tous les messages. Mais pour un calcul de grande taille, l'engorgement du réseau qui en résulterait conduirait à un effondrement des performances globales du calcul (voire même à un arrêt du programme). Cette saturation du système proviendrait d'une part de la trop grande taille des messages échangés, et d'autre part du nombre trop élevé de messages qui circuleraient simultanément sur le réseau (un message par couple de processeurs émetteur/destinataire). Pour éviter cela, nous avons choisi de limiter à la fois la taille des

messages et le nombre de messages pouvant circuler simultanément. Nous allons montrer dans les sections suivantes de quelle manière nous avons procédé.

4.3.2.2 Limitation de la taille des messages

Nous avons donc décidé de plafonner la taille des messages : par défaut, cette taille maximale est de 50.000 nombres réels (soit 400 kilo-octets en double précision). Mais en pratique, les messages envoyés contiennent des fonctions de radiation, donc leur taille ne peut être fixée avec précision (sauf si on choisissait de couper certaines fonctions de radiation en deux, ce qui n'a pas été fait). Pour simplifier, on se contentera de créer des messages en exécutant les tâches `packF`, et en s'arrêtant dès que la taille du message dépasse le seuil fixé. Ainsi, si on note $s_{\mathcal{F}}$ la taille des fonctions de radiation à un niveau donné, le nombre maximal n_{max} de fonctions contenues dans un seul message sera donné par :

$$(n_{max} - 1)s_{\mathcal{F}} < 50.000 \leq n_{max}.s_{\mathcal{F}}$$

Bien sûr, aux plus hauts niveaux de l'arbre les fonctions de radiation peuvent peser plusieurs méga-octets chacune. Mais il nous a paru plus simple de ne pas procéder au découpage avant envoi des fonctions de radiation, d'autant que pour les cas de calcul les plus importants, les niveaux élevés bénéficieront d'un traitement spécifique (cf. 4.3.3).

4.3.2.3 Ecriture des tours de communications

Une fois les messages redéfinis, et toujours dans l'optique d'éviter l'engorgement du réseau, nous allons réorganiser la phase d'échange effectif des messages. Les communications se feront tour par tour, sachant que à chaque tour de communication un processeur peut envoyer *au plus un* message et recevoir *au plus un* message (ces deux échanges ne se font pas forcément avec le même interlocuteur). De cette manière, le nombre de messages circulant simultanément à un instant donné est inférieur ou égal au nombre de processeurs. Le temps nécessaire à l'ensemble des processeurs pour réaliser un tour de communication étant a priori invariable, notre objectif va être de minimiser le nombre de tours nécessaires pour accomplir chaque sous-liste de communication $\mathcal{L}'_i$.

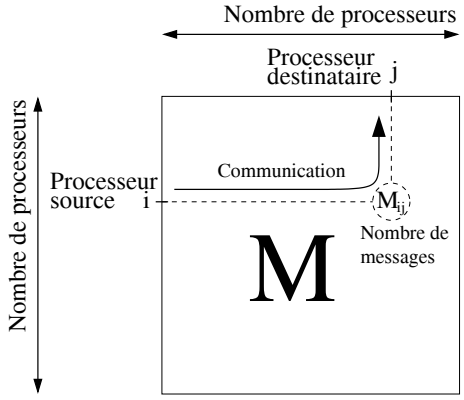


FIG. 4.15 – *Matrice des messages M*

Pour une sous-liste de communication $\mathcal{L}'$ donnée, chaque processeur parcourt cette sous-liste (composée uniquement de tâches `packF`) et détermine le nombre de messages qu'il va

envoyer à chacun des autres processeurs. Après regroupement en parallèle, on obtient la matrice des messages M définie ainsi : M_{ij} est le nombre de messages que le processeur i va devoir envoyer au processeur j (cf. figure 4.15). La somme des éléments d'une ligne i donne le nombre total de messages que le processeur i va envoyer, et la somme sur une colonne j donne le nombre de messages que le processeur numéro j va recevoir. Le nombre de tours nécessaires est forcément supérieur à chacune de ces sommes en ligne et en colonne. Notons enfin que cette matrice est à diagonale nulle, et qu'elle n'est pas symétrique (même si elle peut s'en approcher, notamment pour les sous-listes $\mathcal{L}'$ précédant les phases de transfert).

Pour démarrer un premier tour de communication, la question qui se pose alors est la suivante : dans la matrice M, quel message faire circuler en premier ? Pour y répondre, nous allons utiliser un système de notation : nous allons attribuer à chaque case de la matrice M un score, et le couple (i,j) obtenant la meilleur note déterminera le premier message envoyé. Pour compléter ce tour de communication, on « neutralise » dans la matrice M la ligne i et la colonne j, signifiant ainsi que le processeur i ne peut plus envoyer de message dans ce tour, et que le processeur j ne peut plus en recevoir. En utilisant à nouveau le même système de notation dans le reste de M, on détermine le second message envoyé. En répétant ainsi ce processus jusqu'à ce que M soit entièrement neutralisée, on écrit le premier tour de communication. Il suffit ensuite de mettre à jour les valeurs contenues dans M, et de recommencer le même algorithme pour écrire tous les autres tours de communication. On arrête lorsque M est uniformément nulle, ce qui signifie que tous les messages ont été placés dans des tours de communication. Pour un processeur donné, le résultat de cet algorithme est un couple de liste (SRC_k) et $(DEST_k)$ avec $1 \leq k \leq N_{tour}$ indiquant qu'au k-ième tour, il nous faudra envoyer un message au processeur $DEST_k$ tout en recevant un message du processeur SRC_k.

Le tout est alors de trouver le système de notation qui donne le meilleur résultat, c'est-à-dire celui qui réalise toutes les communications dans le plus petit nombre de tours N_{tour}. Pour ce faire, nous avons testé 18 formules de notation que nous allons maintenant présenter. On note s_{ij} le score attribué au couple (i,j).

1. On donne la priorité au couple de processeurs (i,j) ayant le plus grand nombre de messages à s'échanger : $s_{ij} = M_{ij}$
2. On donne la priorité au processeur j ayant le plus grand nombre de messages à recevoir : $s_{ij} = \sum_{i_0} M_{i_0 j}$ (avec choix aléatoire en cas d'égalité)
3. On donne la priorité au processeur i ayant le plus grand nombre de messages à envoyer : $s_{ij} = \sum_{j_0} M_{i j_0}$ (avec choix aléatoire en cas d'égalité)
4. On donne la priorité au processeur j ayant le plus grand nombre de messages à recevoir, avec (en cas d'égalité) une préférence pour le processeur i ayant le plus grand nombre de messages à envoyer à j : $s_{ij} = M_{ij} + 10.\sum_{i_0} M_{i_0 j}$ (le coefficient 10 est arbitraire, il a pour seul but de fixer des priorités entre les deux critères de notation).
5. On donne la priorité au processeur i ayant le plus grand nombre de messages à envoyer, avec (en cas d'égalité) une préférence pour le processeur j ayant le plus grand nombre de messages à recevoir de i : $s_{ij} = M_{ij} + 10.\sum_{j_0} M_{i j_0}$
6. On donne la priorité au processeur ayant le plus grand nombre de messages à envoyer OU à recevoir : $s_{ij} = max(\sum_{i_0} M_{i_0 j}, \sum_{j_0} M_{i j_0})$
7. On donne la priorité au processeur ayant le plus grand nombre de messages à envoyer ET à recevoir : $s_{ij} = \sum_{i_0} M_{i_0 j} + \sum_{j_0} M_{i j_0}$
8. On donne la priorité au processeur ayant le plus grand nombre de messages à envoyer OU à recevoir, avec (en cas d'égalité) une préférence pour le couple de processeurs (i,j) ayant

le plus grand nombre de messages à s'échanger : $s_{ij} = M_{ij} + 10.max(\sum_{i_0} M_{i_0 j}, \sum_{j_0} M_{i j_0})$

9. On donne la priorité au processeur ayant le plus grand nombre de messages à envoyer *ET* à recevoir, avec (en cas d'égalité) une préférence pour le couple de processeurs (i,j) ayant le plus grand nombre de messages à s'échanger : $s_{ij} = M_{ij} + 10.(\sum_{i_0} M_{i_0 j} + \sum_{j_0} M_{i j_0})$

On a ainsi défini 9 systèmes de notations. On en définit 9 autres numérotés de 10 à 18 en prenant les 9 opposés. L'idée est dans ce cas de favoriser les processeurs ayant peu de messages à envoyer afin de les libérer plus rapidement. Pour chacun de ces 18 barèmes, on crée virtuellement les listes (SRC_k) et $(DEST_k)$, et on détermine le nombre de tours N_{tour}. Ceci fait, on garde le meilleur barème (celui qui donne la plus petite valeur de N_{tour}) et on crée effectivement les tours de communication (SRC_k) et $(DEST_k)$.

A titre d'illustration, regardons le cas d'un airbus à 1,16 millions d'inconnues, sur 12 processeurs. Dans ce cas, l'octree comporte 10 niveaux et 116.392 feuilles. Le niveau plafond est le niveau 4, et on va s'intéresser à la phase de communication qui précède les transferts au niveau 5. La matrice des messages M correspondante est donnée par le tableau 4.2. On voit qu'il y a en tout 714 messages qui doivent circuler, et que le plus grand nombre de messages envoyés ou reçus est détenu par le processeur 6 qui envoie 86 messages. Dès lors, le nombre de tours de communication sera forcément supérieur ou égal à 86.

	0	1	2	3	4	5	6	7	8	9	10	11	total
0	0	17	5	0	0	6	6	0	0	0	12	15	**61**
1	14	0	16	15	9	12	0	0	0	0	1	1	68
2	3	15	0	17	13	0	0	0	0	0	0	0	48
3	0	15	20	0	16	7	0	0	0	0	0	0	58
4	0	2	2	20	0	18	5	8	0	0	0	0	55
5	12	12	0	8	19	0	12	7	0	0	6	6	82
6	6	0	0	0	10	20	0	18	3	3	13	13	**86**
7	0	0	0	0	4	3	20	0	13	15	13	0	68
8	0	0	0	0	0	0	1	14	0	15	8	0	38
9	0	0	0	0	0	0	1	13	16	0	11	0	41
10	2	2	0	0	0	1	14	18	11	13	0	9	70
11	15	6	0	0	0	5	5	0	0	0	8	0	39
total	**52**	69	43	60	71	72	64	78	43	46	72	44	714

TAB. 4.2 – *Matrice des message : airbus sur 12 processeurs*

Le tableau 4.3 donne le nombre de tours de communication obtenus avec chaque barème. Les 9 premiers systèmes de notations donnent tous le score optimal de 86 : si on regarde d'autres cas de calcul, on observe souvent le même comportement. Afin de départager les ex-æquos, nous allons rajouter un critère de tri.

Si on se réfère au tableau 4.2, on voit que le processeur 0 doit envoyer 61 messages, et en recevoir 52. Idéalement, il peut finir ses communications après 61 tours, et reprendre tout de suite le calcul. En pratique, avec le barème 0, il aura fini ses envois au tour 64 et ses réceptions au tour 84. Ce processeur perd donc 23 tours : si tout s'était passé de manière optimal pour ce processeur, il aurait fini ses communications 23 tours plus tôt. Bien évidemment, plus ce nombre est petit et mieux c'est. On indique dans le tableau 4.3 pour chaque barème le *nombre total* de tours perdus par l'ensemble des processeurs, ainsi que le *nombre maximum* de tours perdus par l'un des processeurs. La meilleur méthode est celle qui donne le plus petit nombre

de tours et qui, en cas d'égalité, minimise le nombre de tours perdus. Ici, c'est le barème 6 qui s'impose.

Barème	N_{tour}	Tours perdus		Barème	N_{tour}	Tours perdus	
		Total	Max.			Total	Max.
1	**86**	214	39	10	92	155	27
2	**86**	146	34	11	98	135	28
3	**86**	203	40	12	97	157	28
4	**86**	214	39	13	111	113	33
5	**86**	218	39	14	92	80	21
6	**86**	**119**	**28**	15	94	133	25
7	**86**	197	40	16	99	67	21
8	**86**	215	39	17	99	99	17
9	**86**	198	40	18	97	64	19

TAB. 4.3 – *Nombre de tours de communication*

Globalement, les barèmes opposés sont meilleurs lorsqu'il s'agit de minimiser les nombres de tours perdus : c'est logique dans la mesure où, comme on l'a vu, ces notations favorisent les processeurs ayant peu de messages afin de les libérer plus tôt. Ainsi les barèmes 16 et 18 donnent les plus petits nombres de tours perdus malgré des valeurs de N_{tour} élevées. Cela signifie que la plupart des processeurs réalisent un nombre de tours proche de leur optimum, et que quelques processeurs (ceux qui ont le plus de messages) réalisent au contraire un nombre de tours loin de leur optimum, ce qui au final pénalise tout le monde. En effet, cela aura pour effet de créer un déphasage en temps entre les processeurs les plus rapides et les plus lents. Nous avons au contraire choisi de favoriser la synchronisation entre les processeurs.

Arrivé à ce point de l'algorithme, chaque processeur connaît le nombre de tours de communication N_{tour} ainsi que la liste des interlocuteurs en émission ($DEST_k$) et en réception (SRC_k) pour $k = 1, \ldots, N_{tour}$. Si $DEST_k$ ou SRC_k vaut -1, cela signifie simplement que le processeur considéré n'envoie ou ne reçoit rien au tour numéro k. Il ne reste donc plus qu'à réécrire les listes de tâches `packF` pour prendre en compte ce mécanisme de communication par tour.

4.3.2.4 Transformation des sous-listes

En pratique, la commande MPI utilisée pour réaliser simultanément l'envoi et la réception d'un message est `MPI_Sendrecv`. La tâche élémentaire correspondante est définie de la manière suivante :

SendRecv(n_{src},n_{dest})

Paramètres :	un numéro de processeur source n_{src}.
	un numéro de processeur destinataire n_{dest}.
Action :	envoi un message au processeur n_{dest}
	et prépare la réception d'un message provenant de n_{src}.

La tâche `packF`($\mathcal{T}_{\mathcal{C}}$,$n_{dest}$) copie la fonction de radiation $\mathcal{T}_{\mathcal{C}}$ dans un buffer, et la tâche `SendRecv`(n_{src},n_{dest}) envoie effectivement ce buffer au processeur n_{dest} tout en réalisant la réception d'un autre message en provenance de n_{src}. Par conséquent, une sous-liste de communication $\mathcal{L}'$ se réécrit en $\mathcal{L}''$ de la manière décrite par l'algorithme 4.5.

$\mathcal{L}'' \leftarrow \varnothing$
// *Boucle sur les tours de communication*
for $k = 1, \ldots, N_{tour}$ **do**
  // *Préparation du message à envoyer*
  $Taille \leftarrow 0$
  while $Taille < TailleMax$ **do**
    // *Boucle sur les tâches de* $\mathcal{L}'$
    for all $\texttt{packF}(\mathcal{T}_{\mathcal{C}}, n_{dest}) \in \mathcal{L}'$ **do**
      if $(n_{dest} = DEST_k)$ **then**
        // *On transfère l'opération packF de* $\mathcal{L}'$ *à* $\mathcal{L}''$
        $\mathcal{L}' = \mathcal{L}' \setminus \texttt{packF}(\mathcal{T}_{\mathcal{C}}, n_{dest})$
        $\mathcal{L}'' = \mathcal{L}'' \cup \texttt{packF}(\mathcal{T}_{\mathcal{C}}, n_{dest})$
        // *On incrémente la taille du message*
        Taille =Taille$+s_{\mathcal{T}}$
      end if
    end for
  end while
  // *Envoi effectif du message*
  $\mathcal{L}'' = \mathcal{L}'' \cup \texttt{SendRecv}(SRC_k, DEST_k)$
end for

Algorithme 4.5: Transformation des communications

Pour chaque tour, on observe deux étapes : au cours de la première étape, on crée un message en transferant les `packF` destiné à $DEST_k$ de $\mathcal{L}'$ à $\mathcal{L}''$ et en s'arrêtant dès que la taille du message dépasse $TailleMax$ (50.000 par défaut). La deuxième étape consiste à réaliser l'envoi effectif du message en rajoutant une tâche `SendRecv`. A la fin, la liste $\mathcal{L}'$ est vide.

Il y a un point qui est ici passé sous silence : l'exploitation des messages reçus. A chaque tour, chaque processeur sait *qui* va lui envoyer un message, mais il ne connaît pas le contenu du message. Le plus simple est d'associer à chaque fonction de radiation un en-tête contenant le niveau et l'index de la cellule $\mathcal{C}$, et le type de fonction $\mathcal{F}$ ou $\mathcal{G}$. Si cet en-tête est envoyé en même temps que la fonction, le processeur destinataire saura directement le contenu du message reçu, et pourra le placer de manière adéquate dans son octree. De cette manière, chaque processeur doit seulement se soucier de ce qu'il envoie, et non de ce qu'il reçoit.

4.3.3 Parallélisation par direction

4.3.3.1 Préambule

Nous allons terminer cette section consacrée à la parallélisation effective de la méthode multipôle rapide par la présentation d'une autre approche particulièrement utile pour les calculs de très grande taille (au-delà de 10 millions d'inconnues) ou pour de grands nombres de processeurs (au-delà de 50). Nous verrons de manière plus précise dans le chapitre suivant l'impact de ces modifications sur les performances, mais nous attirons dès maintenant l'attention du lecteur sur le fait que ces développements ne servent réellement que pour des cas de calcul d'une certaine importance.

4.3.3.2 Limitations de la méthode

Au niveau plafond, toutes les interactions qui n'ont pas encore été prises en compte aux niveaux inférieurs doivent l'être. Aussi les transferts se font ils entre toutes les cellules non-voisines. Si un processeur gère une cellule $\mathcal{C}$, il va devoir réaliser le transfert sur $\mathcal{C}$ de toutes les cellules non-voisines, c'est-à-dire la quasi totalité du niveau, à quelques exceptions près. Si on travaille en parallèle, cela a principalement deux conséquences :

- D'une part, chaque processeur va devoir héberger localement la presque totalité du niveau plafond ;
- D'autre part, chaque processeur va devoir envoyer les cellules qu'il gère en local à quasiment tous les autres processeurs.

En terme de stockage de l'octree, le niveau plafond sera presque entièrement présent sur chaque processeur au lieu d'être partagé. Cela aura pour effet de rendre quasiment obligatoire l'utilisation du mode out-of-core, avec une chute de performance à la clé. En terme de parallélisme, chaque processeur va recevoir tout le niveau plafond, ce qui implique que la quantité de données échangées va croître proportionnellement au nombre de processeurs. Cela aura bien sûr un effet désastreux sur l'efficacité parallèle.

Pour éviter ce genre de problème, une méthode astucieuse (proposée notamment dans [48]) consiste à paralléliser les plus hauts niveaux de l'arbre non plus par cellule, mais par direction sur la sphère unité. Nous allons expliciter cela dans la suite de cette section.

4.3.3.3 Nouvelle distribution des données

L'idée de départ est très simple : on doit paralléliser les fonctions de radiation $\mathcal{F}(\vec{k})$. Aux niveaux proches des feuilles, où l'on a beaucoup de fonctions $\mathcal{F}$ et peu de directions $\vec{k}$, on distribue les fonctions entre les processeurs, qui gèrent alors toutes les directions. Au niveaux élevés, où l'on a peu de fonctions $\mathcal{F}$ et beaucoup de directions $\vec{k}$, on distribue les directions $\vec{k}$ entre les processeurs, qui gèrent alors toutes les fonctions $\mathcal{F}$ du niveau concerné.

Direction	$\vec{k}_1$	$\vec{k}_2$	$\dots$	$\vec{k}_n$
$\mathcal{F}_1$	$\mathcal{F}_1(\vec{k}_1)$	$\mathcal{F}_1(\vec{k}_2)$	$\dots$	$\mathcal{F}_1(\vec{k}_n)$
$\mathcal{F}_2$	$\mathcal{F}_2(\vec{k}_1)$	$\mathcal{F}_2(\vec{k}_2)$	$\dots$	$\mathcal{F}_2(\vec{k}_n)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\mathcal{F}_m$	$\mathcal{F}_m(\vec{k}_1)$	$\mathcal{F}_m(\vec{k}_2)$	$\dots$	$\mathcal{F}_m(\vec{k}_n)$

TAB. 4.4 – *Représentation simplifiée des fonctions de radiation*

Si on représente les valeurs des fonctions de radiations $\mathcal{F}_i(\vec{k}_j)$ (avec $1 \leq i \leq m$ et $1 \leq j \leq n$) comme dans le tableau 4.4, on voit que la parallélisation par cellule est un découpage selon les lignes, ce qui est naturel si $m \gg n$ (plus de fonctions que de directions : niveaux bas de l'octree). A l'inverse, la parallélisation par direction correspond à un découpage en colonne, ce qui est préférable si $n \gg m$ (plus de directions que de fonctions : niveaux hauts de l'octree).

Regardons comment se comportent ces deux types de parallélisation vis-à-vis des tâches élémentaires qui composent la FMM. Les tâches d'initialisation, de montée, de descente et d'intégration travaillent à chaque fois sur – au plus – deux fonctions de radiation, mais utilisent toutes les valeurs de ces dernières. La parallélisation par cellule est alors très préférable. Par exemple, les montées/descentes sont constituées d'une FFT directe sur les lignes de la fonction de radiation de départ, d'un produit matrice-vecteur, et d'une FFT inverse. On imagine mal réaliser un tel calcul si le contenu de la fonction de radiation est distribué sur N processeurs. En revanche, la parallélisation par direction est idéale pour les tâches de transfert. En effet, cette dernière travaille sur chaque direction indépendamment, elle pourrait donc s'exécuter *sans communication* à un niveau parallélisé par direction. On le voit, les deux types de parallélisations ont leurs avantages et leurs inconvénients. Nous allons maintenant présenter de quelle manière nous en avons tiré partie.

4.3.3.4 Nouvelles tâches et nouvelles sous-listes

Nous avons choisi de favoriser une approche mixte, en utilisant la parallélisation par cellule comme méthode par défaut, et en basculant vers l'autre type de distribution uniquement pour traiter les phases de transfert aux niveaux les plus hauts. Pour tous les niveaux situés entre le niveau plafond et un niveau intermédiaire appelé « niveau distribué », les transferts sont traités de la manière suivante :

1. Passage de la distribution par cellule à la distribution par direction pour toutes les fonctions de radiation $\mathcal{F}$ du niveau considéré ;
2. Réalisation des transferts sans communication ;
3. Retour à la distribution par cellule pour les fonctions de radiation $\mathcal{G}$.

Pour les niveaux situés au-delà du niveau distribué, les transferts et les communications sont inchangés.

En conservant la parallélisation par cellule comme parallélisation par défaut, on minimise les modifications à faire dans le code. Seules certaines phases de transfert (définies à la section 4.3.1.2.2) vont être modifiées. Pour cela, on va avoir besoin de deux nouvelles tâches élémentaires, réalisant le passage d'une distribution à une autre.

ScatterF($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$
Action :	Distribue par direction la fonction $\mathcal{F}_{\mathcal{C}}$ entre tous les processeurs.
Contribution :	néant.
Utilisation :	Incrémente $C^{\Uparrow}_{\mathcal{F}_{\mathcal{C}}}$ sur le processeur $n_{\mathcal{C}}$.

La tâche élémentaire `ScatterF` réalise le passage de la parallélisation par cellule à la parallélisation par direction pour la fonction $\mathcal{F}_{\mathcal{C}}$. Avant l'exécution de cette tâche, la fonction $\mathcal{F}_{\mathcal{C}}$ est entièrement stockée sur le processeur $n_{\mathcal{C}}$. Après cette tâche, les valeurs prises par $\mathcal{F}_{\mathcal{C}}$ sont équitablement distribuées entre tous les processeurs. La fonction utilisée est la routine de la librairie MPI appelée `MPI_Scatter`. Durant cette communication, il y a un seul processeur source, mais tous les processeurs sont destinataires.

GatherG($\mathcal{C}$)

Paramètres :	Une cellule $\mathcal{C}$
Action :	Regroupe sur une seule cellule la fonction $\mathcal{G}_{\mathcal{C}}$.
Contribution :	Incrémente $C^{\Downarrow}_{\mathcal{G}_{\mathcal{C}}}$ sur le processeur $n_{\mathcal{C}}$.
Utilisation :	néant.

La tâche élémentaire `GatherG` réalise l'inverse de `ScatterF`, c'est-à-dire le passage de la parallélisation par direction à la parallélisation par cellule pour la fonction $\mathcal{G}_{\mathcal{C}}$. En entrée de cette tâche, la fonction $\mathcal{G}_{\mathcal{C}}$ a ses valeurs distribuées entre tous les processeurs. Un appel à la commande `MPI_Gather` regroupe toutes ces données sur le processeur $n_{\mathcal{C}}$ qui héberge la cellule $\mathcal{C}$ et ses deux fonctions de radiations.

Puisque l'opération `ScatterF` s'applique forcément à une fonction $\mathcal{F}$, et `GatherG` à une fonction $\mathcal{G}$, ces deux tâches élémentaires peuvent se contenter de recevoir un seul argument : la cellule $\mathcal{C}$. Il n'y a pas d'ambiguïté sur la fonction à traiter. A un niveau autre que le niveau plafond, la sous-liste réalisant les transferts est créée par l'algorithme 4.6.

Si on travaillait au niveau plafond, il suffirait de remplacer dans l'algorithme 4.6 le test « $\mathcal{C}$ banlieue de $\mathcal{C}'$ » par « $\mathcal{C}$ non-voisine de $\mathcal{C}'$ » comme déjà vu notamment au paragraphe 2.1.3.4. On peut faire deux remarques supplémentaires : d'une part, l'opération élémentaire `transfert`($\mathcal{C}$,$\mathcal{C}'$) se fait désormais sur tous les processeurs, que $\mathcal{C}'$ soit locale ou non. Bien sûr, il s'agit d'un transfert uniquement sur les directions $\vec{k} \in \mathcal{S}$ gérées localement. D'autre part, les opérations `GatherG` devant se faire après la sous-liste $\mathcal{L}_i$, elles sont placées dans la sous-liste de communication $\mathcal{L}'_{i+1}$.

4.3.3.5 Gain escompté et exemple d'application

On a expliqué à la section 4.3.3.2 les limitations issues de la parallélisation par cellule. Si on s'intéresse uniquement au niveau plafond, que l'on note T sa taille totale (partagée en $T/2$ Mo de fonctions $\mathcal{F}$ et autant pour les $\mathcal{G}$) et N le nombre de processeurs, alors chaque processeur va recevoir et stocker localement presque toutes les fonctions $\mathcal{F}$ (soit $T/2$ Mo) et un N-ième des fonctions $\mathcal{G}$ (soit $1/N \times T/2$ Mo). La quantité totale de données échangées s'élève donc à $N.T/2$.

```
// Passage à la parallélisation par direction
for all C cellules du niveau l do
   L'_i = L'_i ∪ ScatterF(C)
end for
// Transferts
for all C, C' cellules du niveau l do
   if (C banlieue de C') then
      L_i = L_i ∪ transfert(C,C')
   end if
end for
// Retour à la parallélisation par cellule
for all C cellules du niveau l do
   L'_{i+1} = L'_{i+1} ∪ GatherG(C)
end for
```

Algorithme 4.6: Transfert aux niveaux élevés

Dans le cas d'une parallélisation par direction, chaque processeur devra stocker la totalité des fonctions $\mathcal{F}$ et $\mathcal{G}$ locales (soit T/N Mo) , et un N-ième des directions pour les fonctions $\mathcal{F}$ et $\mathcal{G}$ non-locales (soit à nouveau T/N Mo). Pour ce qui est des communications, chaque élément de fonction de radiation circulera une fois et une seule, soit en tout T Mo échangés. En théorie, on gagne un facteur $(N+1)/4$ pour le stockage et $N/2$ pour les communications au niveau plafond. Si par exemple $N = 64$, le gain est conséquent.

Vérifions sur un cas réel l'efficacité de cette amélioration. Considérons le cas d'une sphère de diamètre égal à 150 longueurs d'onde, maillée avec 10 points par longueur d'onde. Le maillage comporte 25.579.200 inconnues. L'octree comporte 11 niveaux. Le niveau plafond est le niveau 4, sa taille est de 458,3 Mo. Si on tentait de réaliser un calcul monoprocesseur sur cet objet, la taille mémoire maximum requise pour stocker les fonctions de radiation serait de 9538,6 Mo. On réalise nos tests sur une machine IBM SP3 à $N = 64$ processeurs. Le tableau 4.5 souligne les principales différences entre un code utilisant exclusivement la parallélisation par cellule (colonne de gauche), et un code utilisant la parallélisation par direction aux niveaux 4 et 5 (colonne de droite). Dans ce tableau, on donne tout d'abord les tailles en méga-octets des niveaux 4, 5 et de tous les niveaux de l'octree stockés sur le processeur 0. Puis on détaille les temps d'exécution des transferts à ces deux niveaux, on séparant le calcul pur, les communications, les accès disques (dans le cas out-of-core) et les barrières (i.e. les synchronisations entre les processeurs). On termine avec le temps total d'un produit multipôle dans les deux cas.

Au niveau 4, le gain est net : la quantité de données stockées localement est divisé par 14 (32,4 Mo contre 466,0 Mo) ce qui est assez proche de la valeur espérée de $(N+1)/4$, et le temps de communication est divisé par 4,5 (10,5 s contre 46,8 s). Là, on est par contre assez loin du gain de $N/2$ prévu par la théorie. Une explication possible est que lors du passage de la parallélisation par cellule à la parallélisation par direction, on remplace des communications point-à-point (un émetteur, un receveur) par des communications globales (un émetteur et N receveurs pour le `scatterF`, N émetteurs et un receveur pour le `gatherG`).

	Parallélisation	
	par cellule	par direction
Mémoire locale		
Niveau 4	466,0 Mo	32,4 Mo
Niveau 5	48,4 Mo	29,5 Mo
Total	636,6 Mo	177,1 Mo
Mode	**Out-of-core**	**In-core**
Transferts		
Niveau 4		
Calcul	11,4 s	10,2 s
Communications	46,8 s	10,5 s
Barrière	2,3 s	0,4 s
Disque	0 s	-
Total	82,2 s	21,1 s
Niveau 5		
Calcul	2,8 s	2,4 s
Communications	1,1 s	15,6 s
Barrière	2,6 s	0,6 s
Disque	1,7 s	-
Total	8,2 s	18,6 s
Produit Multipôle	**553,2 s**	**444 s**

Tab. 4.5 – *Performance comparée des deux parallélisations*

Or ces communications globales sont pénalisantes car elles imposent de synchroniser tous les processeurs à chaque fois (une fois par fonction $\mathcal{F}$ avant les transferts, puis une fois par fonction $\mathcal{G}$ après). C'est pourquoi la diminution de la quantité de données échangées ne se traduit pas par une diminution équivalente des temps de communication. Néanmoins, la baisse reste significative.

Au niveau 5 en revanche, l'utilisation des transferts parallélisés par direction ne semble pas intéressante : le gain en stockage est nettement plus faible (29,5 Mo contre 48,4 Mo) et le ralentissement des communications est sensibles (15,6 s contre 1,1 s). Une étude plus complète du paramètre « niveau distribué » sera faite à la section 5.3.3. Remarquons simplement que le niveau 4 étant le niveau plafond, les transferts au niveau 4 se font entre toutes les cellules non-voisines, alors qu'au niveau 5 ils se font seulement entre cellules banlieues. Il y a moins de transferts et moins de communications au niveau 5 qu'au niveau 4, il y a donc moins à gagner à effectuer l'amélioration proposée.

4.4 Autres aspects de la parallélisation

Pour terminer ce chapitre, nous allons évoquer d'autres aspects du travail de parallélisation d'un code multipôle rapide.

4.4.1 Fonctionnalités héritées du mode séquentiel

Nous avons mis au point un code séquentiel aussi performant que possible grâce au techniques exposées dans les chapitres précédents. Nous allons maintenant vérifier que le passage au mode parallèle a néanmoins permis de conserver le bénéfice de ces optimisations.

4.4.1.1 Mode out-of-core

Le mode out-of-core existe toujours. Grâce à l'utilisation de la parallélisation par direction, on arrive à obtenir une très bonne distribution des données, ce qui rend moins fréquent l'utilisation du mode out-of-core. Il est cependant toujours disponible, dans une version améliorée par rapport à celle présentée à la section 2.5. Désormais, le nombre de groupes est déterminé automatiquement pour chaque processeur, à chaque niveau et pour chaque famille de fonctions ($\mathcal{F}$ ou $\mathcal{G}$) à partir de la mémoire vive disponible (qui est un paramètre donné par l'utilisateur, et est 500 Mo par défaut). Dans la mesure où les niveaux les plus bas sont en général les plus volumineux, le mode out-of-core essaie de limiter les accès disque en traitant la FMM en haut de l'arbre en mode in-core, et le bas et les feuilles en mode out-of-core. Quoiqu'il arrive, le seuil de mémoire disponible n'est en tout cas jamais dépassé. Ceci dit, même sur de très gros cas de calcul, l'efficacité de notre distribution des octrees permet de se passer du mode out-of-core. Par exemple, dans le cas de la sphère à 25 millions d'inconnues sur 64 processeurs, les fonctions de radiations utilisent seulement 177 Mo par processeurs c'est-à-dire bien moins que le seuil de 500 Mo. L'intérêt d'avoir un tel mode out-of-core automatique est de pouvoir lancer des calculs de toutes tailles sans se soucier de la quantité de mémoire nécessaire.

4.4.1.2 Traitement des transferts

On a vu à la section 2.2 diverses méthodes pour accélérer le traitement des phases de transferts. Regardons celles qui se conservent en parallèle :

Ecriture creuse des matrices de transfert (cf. 2.2.1) Cela continue à fonctionner, mais aux niveaux parallélisés par direction le seuillage doit se faire en parallèle, puisque chaque processeur ne calcule qu'une partie des matrices de transfert.

Réordonnancement des opérations de transfert (cf. 2.2.2.1) Le fait de réordonner les listes de transferts pour réutiliser au maximum chaque matrice de transfert continue à fonctionner comme dans le cas séquentiel.

Utilisation des symétries pour le calcul des matrices de transfert (cf. 2.2.2.2) Cela ne marche que si le processeur qui calcule une matrice de transfert la calcule entièrement. Cette option est donc automatiquement désactivée aux niveaux parallélisés par direction. Mais en contrepartie, ce calcul est désormais parallélisé entre tous les processeurs, donc il n'y a pas de ralentissement.

Conservation des matrices de transfert antérieures (cf. 2.2.2.3) Cela continue à marcher de manière inchangée.

4.4.2 Interactions proches

La gestion des interactions proches a été évoquée à la section 2.4.3 dans le cas séquentiel. On y avait en particulier souligné l'intérêt de gérer la matrice d'interaction proche non pas degré de liberté par degré de liberté, mais feuille par feuille afin de manipuler la matrice sous forme de petits blocs pleins, plutôt que de grands blocs vides. De ce fait, le traitement

des interactions proches s'appuie sur un octree (tout comme la FMM pour les interactions lointaines). Il semble naturel d'utiliser deux fois le même octree pour ces deux usages. En parallèle, cela ne semble pas souhaitable.

En effet, la répartition de la charge de calcul entre les différents processeurs est de nature tout à fait différente pour les interactions proches et lointaines (c'est-à-dire FMM). Le cas des interactions lointaines a été traité à la section 4.2.3 : la charge de calcul associée à une cellule va dépendre du nombre de montées, descentes, transferts, initialisations et intégrations appliqués aux fonctions de radiation de cette cellule. Pour ce qui est des interactions proches, seules les feuilles de l'arbre vont être concernées, et le bloc de matrice associé à une feuille $\mathcal{C}$ aura pour nombre de lignes le nombre d'inconnues contenues dans $\mathcal{C}$, et pour nombre de colonnes le nombre d'inconnues contenues dans les voisins de $\mathcal{C}$. Ajoutons que la taille de cette matrice locale d'interaction entre $\mathcal{C}$ et ses voisines détermine le temps nécessaire pour réaliser le produit matrice proche-vecteur, mais aussi le temps pour assembler cette matrice. Or, l'assemblage de la matrice des interactions proches est une opération longue qui doit absolument être bien équilibrée.

On le voit, l'évaluation de la charge de travail associée aux cellules de l'octree diffère totalement selon que l'on s'intéresse aux interactions proches ou aux interactions lointaines. C'est pourquoi nous avons choisi de créer deux octrees séparés, l'un pour la partie strictement « multipôle du calcul », l'autre pour la partie « interactions proches ». Le principal inconvénient que l'on pourrait trouver à ce système est le surcoût en place mémoire. En fait, il n'en est rien, car les cellules du nouvel octree sont à la fois moins nombreuses (seules les feuilles sont utilisées) et moins volumineuses (il y a beaucoup moins d'informations à stocker). L'avantage est que l'on peut ainsi réaliser séparément l'équilibrage de charge pour la FMM et pour les interactions proches (assemblage et produit matrice-vecteur). La seule contrainte, à ne pas oublier au moment de créer ces deux octrees, est qu'ils doivent avoir la même racine et la même taille de feuille.

Conclusion

Dans ce chapitre, nous avons présenté de manière approfondie les choix faits et les méthodes utilisées pour paralléliser le code FMM séquentiel étudié à travers les chapitres précédents. Nous avons détaillé les problèmes les plus délicats, comme la distribution des octrees et le traitement effectifs des échanges de messages. Nous avons également souligné le fait que la plupart des optimisations réalisées dans le cas séquentiel restaient valides en parallèle. Afin de vérifier la justesse de nos choix, il convient maintenant de tester exhaustivement cette implémentation parallèle de la méthode multipôle rapide : c'est l'objet du prochain chapitre.

Chapitre 5

Performance parallèle

Sommaire

Introduction **193**
5.1 Présentation des architectures **193**
5.2 Présentation des cas-tests utilisés **197**
5.3 Choix des paramètres **197**
5.4 Scalabilité **201**
5.5 Calculs de grande taille **213**
Conclusion **220**

Introduction

Dans ce chapitre, nous allons exposer les performances de l'implémentation parallèle de la méthode multipôle rapide présentée au chapitre précédent. Ce chapitre est donc le pendant parallèle du chapitre 3 consacré aux performances séquentielles.

Nous allons tout d'abord présenter les différentes machines que nous avons utilisées au cours de nos tests en soulignant leurs spécificités à chacune. Ensuite, nous introduirons les trois cas tests de référence exploités pour nos mesures. La troisième section concerne le choix des paramètres liés à la parallélisation. Puis nous regarderons comment se comportent les performances du code lorsque le nombre d'inconnues ou le nombre de processeurs augmente. Enfin, nous terminerons ce chapitre par la présentation de quelques cas de calcul académiques ou industriels de très grande taille. Au demeurant, il s'agira des plus grands calculs de ce type rendus public à ce jour, tant par le nombre d'inconnues que par le diamètre électrique des objets diffractants.

5.1 Présentation des architectures

Nous allons tout d'abord présenter les quatre machines utilisées pour réaliser nos tests. Nous avons sélectionné ces configurations à la fois pour leur diversité technologique (mémoire distribuée ou partagée, composants standards ou spécifiques) et pour leur représentativité des machines effectivement utilisées dans les milieux de l'industrie et de la recherche.

5.1.1 Station biprocesseur

La première machine présentée est la plus simple et la plus abordable des configurations multiprocesseurs : il s'agit d'un micro-ordinateur de type PC équipé d'une carte mère biprocesseur. Cette configuration Compaq est équipée de deux processeurs pentium III à 866 MHz, et de 1 Go de mémoire vive haute performance de type RDRAM PC800. Cette mémoire est à accès partagé entre les processeurs. Enfin, ce PC est équipée d'un disque dur SCSI Ultra160 de 36 Go. La figure 5.1 représente cette machine.

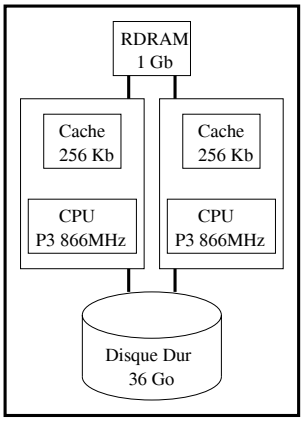


FIG. 5.1 – *Station Compaq biprocesseur*

5.1.2 Ferme de PC

La ferme de PC utilisée est mise en œuvre à l'INRIA par le service d'exploitation et de maintenance informatique. Elle est constituée de 19 noeuds biprocesseurs pentium III 933 MHz, et de 14 noeuds bipentium III 500 MHz. Ces noeuds sont reliés via un réseau fast-ethernet à un commutateur full-duplex. Chaque noeud dispose de 512 Mo de mémoire vive, et d'un disque dur de 18 Go. Les noeuds utilisent le système d'exploitation Linux 2.2, le système de soumission de travaux est LSF, et le code multipôle est compilé avec les compilateurs GNU et la librairie MPICH.

Nous n'utiliserons pas les noeuds à 500 MHz, et ne tirerons pas partie du caractère biprocesseur des machines. En effet, par défaut, le mécanisme de soumission des travaux peut indifféremment nous attribuer un ou deux processeurs sur chaque machine. Ainsi, une demande de 16 processeurs peut nous renvoyer soit 8 noeuds biprocesseurs pour notre usage exclusif, soit 16 noeuds avec un processeur sur chaque (les ressources de la machine pouvant alors être partagées avec un autre travail), soit une combinaison de noeuds exclusifs et de noeuds partagés. Or ces différentes combinaisons donneront des temps d'exécution différents. Aussi, pour être sûr de pouvoir comparer nos résultats, nous demanderons toujours explicitement un processeur par noeud. Cela permettra en outre d'avoir accès à davantage de mémoire vive. De ce fait, la machine telle qu'on l'utilisera est symbolisée sur la figure 5.2.

On est ici en présence d'une machine à mémoire distribuée : chaque processeur n'a accès qu'à sa propre mémoire. A l'exception du switch d'interconnexion, tous les éléments utilisés ici sont standards et peu coûteux. La contrepartie à cette économie est l'obligation d'utiliser des librairies comme MPI ou PVM pour paralléliser les codes (ce qui est en général beaucoup plus

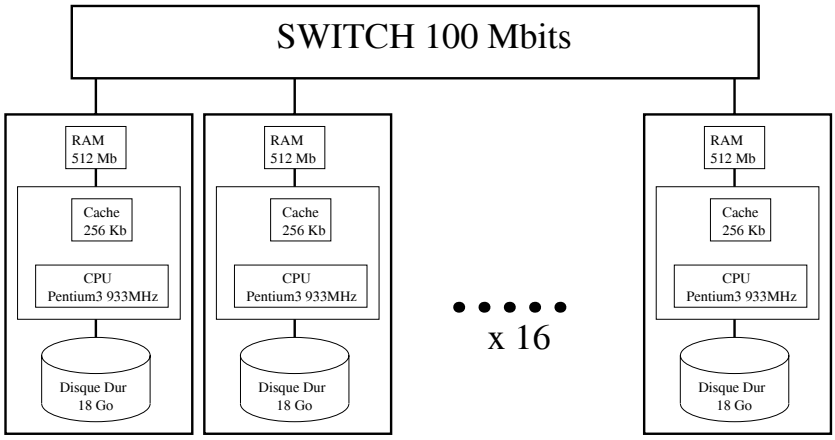


FIG. 5.2 – *Ferme de PC de l'INRIA sophia*

long et complexe que d'utiliser openMP). Les cas-tests seront passés sur 4, 8 ou 16 processeurs, la machine sera alors désignée sous le nom ferme-4, ferme-8 ou ferme-16.

5.1.3 SGI Origin3800

La troisième machine présentée est une authentique machine parallèle, dans la mesure où elle n'est pas constituée de composants issus de l'univers des compatibles PC. En l'occurrence, il s'agit d'un calculateur Silicon Graphics de type Origin 3800. Cette machine est installée au Centre Informatique National de l'Enseignement Supérieur [1] à Montpellier. Elle est équipé de 512 processeurs MIPS RISC R14000 à 500 Mhz, de 256 Go de mémoire partagée, et d'un système de fichier externe d'une capacité de près d'un téraoctet. La mémoire vive est en réalité physiquement distribuée, mais virtuellement partagée. Chaque processeur dispose de 512 Mo de mémoire qui lui sont rattachés directement (comme pour une machine à mémoire distribuée), mais il peut accéder via un réseau dédié *et de manière tout à fait transparente pour l'utilisateur* à la mémoire vive de chacun des autres processeurs (comme pour une machine à mémoire partagée). L'accès aux données est dès lors techniquement différent selon que ces données sont présentes dans la mémoire locale ou non. Cela justifie l'appellation NUMA utilisée par SGI pour désigner l'architecture des machines Origin : NUMA signifie « non-uniform memory access » (« accès non-uniforme à la mémoire »). La figure 5.3 illustre la composition de cette machine. Pour l'utilisateur, tout se passe comme pour une machine à mémoire partagée. Il faut simplement garder à l'esprit qu'au delà des 512 premiers méga-octets de mémoire vive utilisés, les temps d'accès deviennent moins bon. Nous avons utilisé cette machine pour des tests à 4, 8, 16 et parfois 32 processeurs. Ces configurations seront désignées par les noms Origin-4, Origin-8, Origin-16 et Origin-32.

5.1.4 IBM SP3

La quatrième machine utilisée pour nos tests se trouve également au CINES à Montpellier. Il s'agit d'une IBM SP3 composée de 28 noeuds de calcul SMP (*symmetric multi-processor*) à 16 processeurs Power 3+/375 Mhz interconnectés par un commutateur Colony (débit théorique

1. www.cines.fr

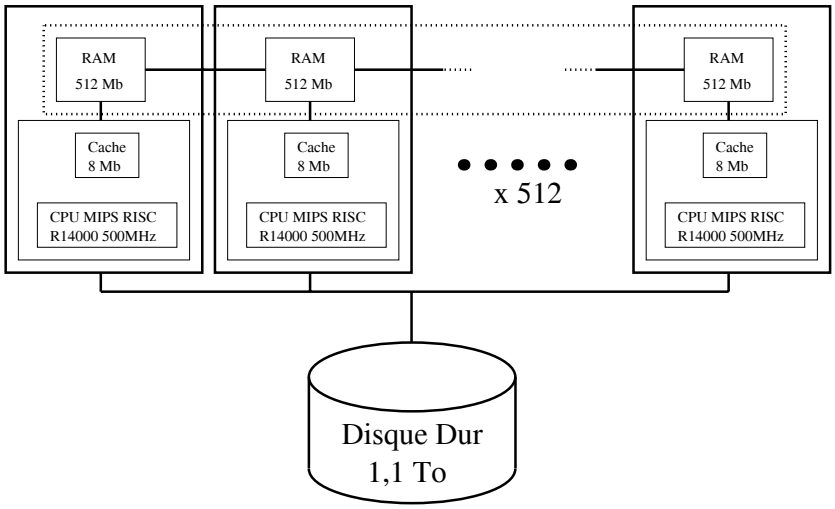


FIG. 5.3 – *Silicon Graphics Origin 3800 du CINES*

de 1 Go par seconde). Chaque noeud est doté de 16 Go de mémoire partagée et développe une puissance crête théorique de 24 Gflops. En outre, chaque noeud dispose d'un disque local d'une capacité anecdotique (2 Go) au regard de la machine. Un système de stockage de 2,2 To est accessible avec une bande passante de 250 Mo/s pour l'ensemble des noeuds via le commutateur Colony.

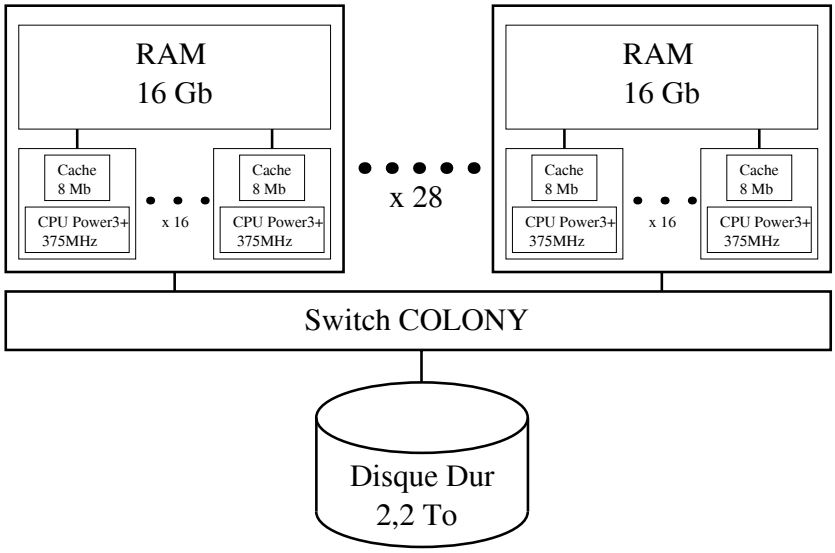


FIG. 5.4 – *IBM SP3 du CINES*

Dans une configuration de ce type, les échanges de messages se font par copie directe dans la mémoire si les deux processeurs source et destination sont dans le même noeud, et par le réseau colony dans les autres cas. Nous avons utilisé cette machine pour des tests à 8, 16 et 32 processeurs. Ces configurations seront désignées par les noms SP3-8, SP3-16 et SP3-32.

Occasionnellement, de plus grands nombres de processeurs seront exploités.

5.2 Présentation des cas-tests utilisés

Nous allons présenter les trois maillages que nous avons utilisés pour les calculs parallèles.

Airbus1M : Il s'agit du maillage d'un airbus A318 raffiné par nos soins. La technique de raffinement est celle de la section 3.2.4.

Furtif4M : C'est une version raffinée du pseudo F117 représenté sur la figure 3.29.

Cetaf5M : Il s'agit d'une maquette CETAF.

Cas test	Airbus1M	Furtif4M	Cetaf5M
Nombre d'inconnues	1,160,124	3,924,000	4,851,900
Nombre de triangles	773,416	2,616,000	3,234,600
Diamètre	101 λ	130 λ	150 λ
Nombre de niveaux FMM	10	11	11
Taille totale de l'octree	706.2 Mo	1128.9 Mo	2327.4 Mo

TAB. 5.1 – *Description des cas-tests utilisés*

Cas test	Airbus1M		Furtif4M		Cetaf5M	
	Taille	Effectif	Taille	Effectif	Taille	Effectif
Niveau 2	65,9	16	188,5	12	204,2	13
Niveau 3	44,1	40	98,9	24	127,8	31
Niveau 4	**39,6**	**126**	55,1	50	**90,3**	**82**
Niveau 5	44,6	466	**45,3**	**144**	128,9	410
Niveau 6	55,6	1779	58,0	606	148,1	1548
Niveau 7	81,1	7357	91.0	2911	189.7	6070
Niveau 8	130,8	29763	128.5	11667	268.4	24362
Niveau 9	287.7	116392	207.1	47125	427.2	97221
Niveau 10			460.9	186449	950.1	384345

TAB. 5.2 – *Taille et nombre de cellules au sommet des octrees*

Tous les paramètres du maillage qui ne sont pas directement liés à la parallélisation (comme la finesse de maillage) sont volontairement passés sous silence, puisqu'ils ont déjà fait l'objet d'une étude au chapitre 3. En revanche, on a souligné dans le tableau 5.2 les tailles (en mégaoctets) et les nombres de cellules aux différents niveaux des octrees respectifs. Le niveau le moins volumineux de chaque octree est écrit en gras.

5.3 Choix des paramètres

5.3.1 Niveau plafond

5.3.1.1 Introduction

Le choix du niveau plafond de la FMM (qui est, rappelons le, le plus haut niveau de l'octree exploré par le calcul multipôle) a déjà fait l'objet d'une étude dans le cas séquentiel (cf. section

3.4.7). Nous étions alors arrivés à la conclusion que le meilleur choix était, en moyenne, de choisir comme niveau plafond le niveau le moins volumineux en mémoire. En parallèle, ce choix peut être remis en cause pour deux raisons. D'une part, l'occupation en mémoire d'un niveau n'a plus le même sens ni la même valeur qu'en séquentiel. D'autre part, pour avoir une bonne répartition de la charge, il est sans doute peu intéressant d'explorer des niveaux où le nombre de cellules est proche du nombre de processeurs. En fait, ce second point n'est vrai que si la parallélisation se fait par cellule à tous les niveaux. Si on utilise la parallélisation par direction aux niveaux les plus hauts (ce qui sera le cas ici), le nombre de cellules ne pose plus de problème.

5.3.1.2 Tests

Dans un premier temps, nous allons étudier le cas airbus1M sur la ferme de PC, dans des configurations à 4, 8 et 16 processeurs. Le temps requis à chaque fois pour la réalisation d'un produit multipôle complet est donné dans le tableau 5.3. Le meilleur temps est mentionné en caractères gras. La précision FMM est la même à chaque fois et vaut $1{,}18.10^{-2}$. Elle ne dépend ni du nombre de processeurs (fort heureusement), ni du niveau plafond (comme on avait pu le constater dans le cas séquentiel, cf. tableau 3.15). C'est pourquoi nous ne nous intéresserons ici qu'aux temps de calcul du produit multipôle (donnés en seconde).

Niveau Plafond	Ferme-4	Ferme-8	Ferme-16
2	317.68	170.21	118.51
3	269.33	140.02	91.23
4	**255.98**	**129.36**	**81.31**
5	277.17	152.21	97.07
6	403.46	787.40	729.46

TAB. 5.3 – *Choix du niveau plafond : cas Airbus1M sur ferme de PC (temps en s)*

Nous avons ensuite mené le même type de calcul sur IBM SP3, avec 8, 16 puis 32 processeurs, sur le cas-test cetaf5M. Les résultats sont exposés dans le tableau 5.4.

Niveau Plafond	SP3-8	SP3-16	SP3-32
2	488.98	317.90	184.78
3	342.92	224.68	127.03
4	**322.94**	**198.73**	**117.60**
5	346.01	212.15	134.13
6	492.27	349.66	272.07

TAB. 5.4 – *Choix du niveau plafond : cas Cetaf5M sur IBM SP3 (temps en s)*

Enfin, nous avons exécuté cette même série de tests pour le cas furtif4M sur Origin3800 avec 4, 8 et 16 processeurs (tableau 5.5).

5.3.1.3 Interprétation

Si l'on rapproche les résultats obtenus (tableaux 5.3, 5.4 et 5.5) de la présentation des maillages (tableau 5.2), on constate que quels que soient la machine et le nombre de processeurs utilisés le temps d'exécution optimal est toujours atteint en choisissant comme niveau plafond

Niveau Plafond	Origin-4	Origin-8	Origin-16
2	840.73	411.04	258.34
3	847.04	257.23	157.82
4	714.55	224.67	143.51
5	**609.68**	**221.51**	**138.54**
6	661.73	270.35	153.79

TAB. 5.5 – *Choix du niveau plafond : cas furtif4M sur SGI Origin3800(temps en s)*

le niveau le plus petit. On a donc mis à jour un critère très simple pour déterminer le nombre de niveaux à explorer dans l'octree en séquentiel comme en parallèle : il faut s'arrêter au niveau le moins volumineux globalement.

5.3.2 Niveau de séparation

5.3.2.1 Introduction

Nous allons maintenant étudier l'impact du niveau de séparation sur la performance du code FMM. Rappelons que le niveau de séparation désigne le niveau à partir duquel les cellules enfants sont d'office placées sur le même processeur que le parent. Cela n'a pas d'impact sur le résultat du calcul, mais seulement sur le temps de réalisation d'un produit multipôle. Plus le niveau de séparation s'éloigne du niveau des feuilles, plus le risque d'avoir un mauvais équilibrage de la charge augmente. Mais en même temps, on supprime les communications (et donc les barrières) avant les phases de montée et de descente.

5.3.2.2 Tests

On étudie ce temps (en secondes) pour le cas Airbus1M sur la ferme de PC, les résultats sont reportés dans le tableau 5.6. Sur ce cas, le niveau plafond est le niveau 4, mais il est néanmoins possible de choisir un niveau de séparation au-dessus. Le cas où le niveau de séparation est le niveau des feuilles (le niveau 9, ici) correspond au cas non modifié.

Niveau de séparation	Ferme-4	Ferme-8	Ferme-16
2	300.40	159.39	134.52
3	256.83	138.31	91.74
4	278.39	150.76	95.07
5	**258.77**	**136.61**	83.20
6	262.49	137.29	81.73
7	266.31	137.66	81.01
8	263.69	137.23	**80.83**
9	265.97	137.14	81.76

TAB. 5.6 – *Choix du niveau de séparation : cas Airbus1M sur ferme de PC(temps en s)*

Vu le faible écart entre les différentes lignes dans ce tableau, nous ne montrons pas les résultats des tests sur les calculateurs Origin et SP3. En effet, sur ces derniers les temps mesurés sont sujets à des variations en fonction de l'occupation de la machine par l'ensemble des autres utilisateurs (c'est la conséquence de la mémoire et du disque partagés). Or ces

variations peuvent avoir une amplitude supérieure aux écarts observés dans le tableau 5.6. Il n'aurait pas été possible d'interpréter les résultats.

5.3.2.3 Interprétation

Dans le tableau 5.6, tous les temps d'exécution sont très proches à partir d'un niveau de séparation égal à 5. Sur les cas à 8 et 16 processeurs, les écarts sont tellement faibles qu'on ne peut même rien en conclure. Tout ce qu'on peut dire, c'est que sur cette machine le gain lié à l'introduction d'un niveau de séparation est nul. Il est probable que sur une machine avec un réseau de communication très lent, le bénéfice serait plus important.

5.3.3 Niveau distribué

5.3.3.1 Introduction

Le niveau distribué indique le plus bas niveaux où les transferts sont parallélisés par direction. Si on le souhaite, on peut traiter tous les transferts en mode classique (c'est-à-dire parallélisé par cellule). Il suffit de choisir comme niveau distribué un niveau supérieur au niveau plafond (0 par exemple). Sinon, tous les niveaux compris entre le niveau plafond et le niveau distribué (inclus) vont traiter leurs transferts en mode parallélisation par direction, comme vu à la section 4.3.3. On avait alors vu l'intérêt de cette modification pour un très gros cas de calcul. On va s'intéresser ici à des problèmes de tailles plus modestes, et essayer de déterminer un critère simple pour fixer le niveau distribué.

5.3.3.2 Tests

Regardons tout d'abord le cas Airbus1M sur la ferme de PC. Le niveau plafond étant le niveau 4, on fait varier le niveau distribué de 3 (synonyme de « pas de niveau distribué ») à 6. Les résultats sont donnés dans le tableau 5.7, où l'on donne la durée d'un produit FMM en secondes.

Niveau distribué	Ferme-4	Ferme-8	Ferme-16
< 4	260.98	131.02	84.59
4	**257.36**	**129.16**	**82.13**
5	260.11	136.14	86.50
6	267.34	143.11	96.01

TAB. 5.7 – *Choix du niveau distribué : cas Airbus1M sur ferme de PC(temps en s)*

Regardons le cas furtif4M sur Origin3800. Le niveau plafond étant alors le niveau 5, on fait varier le niveau distribué de 4 à 7 (tableau 5.8).

Niveau distribué	Origin-4	Origin-8	Origin-16	Origin-32
< 5	310.15	160.96	86.97	55.40
5	303.44	160.85	**83.98**	**48.00**
6	**301.00**	**159.49**	86.86	51.46
7	304.77	163.85	92.86	65.19

TAB. 5.8 – *Choix du niveau distribué : cas furtif4M sur SGI Origin3800(temps en s)*

Sur SP3, les piètres performances que l'on a obtenu du système de stockage nous empêche de faire des mesures fiables de temps d'exécution pour de grands nombres de processeurs : des ralentissements incontrôlables dus aux accès disques donnent des valeurs fantaisistes aux temps mesurés. Sur la ferme de PC (où les disques sont locaux) et sur l'Origin 3800 (où les accès disque sont cachés par la mémoire vive du système), nous n'avons jamais rencontré ce genre de comportement.

5.3.3.3 Analyse

Sur de petits nombres de processeurs (4 par exemple), les communications globales ne sont pas tellement pénalisées par rapport aux communications point-à-point. Cela est d'autant plus vrai que les communications sont rapides, comme c'est le cas sur Origin. C'est pourquoi dans le tableau 5.8 on obtient des scores très proches dans la colonne Origin-4.

Lorsque le nombre de processeurs croît, il semblerait que le meilleur temps d'exécution soit toujours obtenu en prenant comme niveau distribué le niveau plafond, c'est à dire en traitant les transferts en mode « parallélisation par direction » uniquement au niveau plafond. Cela confirme le résultat obtenu à la section 4.3.3.5 pour un maillage à 25 millions d'inconnues sur 64 processeurs.

5.4 Scalabilité

Cette section est divisée en deux parties. Dans un premier temps, nous allons nous intéresser à la scalabilité numérique, c'est-à-dire la manière dont se comporte le code lorsque l'on fait croître la taille du problème traité pour une machine et un nombre de processeurs donnés. Dans un deuxième temps, nous étudierons la scalabilité parallèle, i.e. le comportement du programme lorsque le nombre de processeurs augmente pour un cas de calcul donné.

5.4.1 Scalabilité numérique

5.4.1.1 Introduction

Dans cette section, nous allons réaliser le pendant parallèle des expérimentations séquentielles de la section 3.6. Nous allons étudier le comportement en temps et en mémoire de notre code multipôle pour des cas de calcul de taille croissante. Pour des raisons de simplicité, nous allons à nouveau utiliser des maillages de sphères. Mais il faut souligner que même si la forme de l'objet peut influer sur les *valeurs* des temps d'exécution et de la mémoire consommée, l'*évolution* de ces grandeurs en fonction du nombre d'inconnues n_{dl} ne devrait pas en dépendre. Soulignons enfin qu'on ne s'intéresse ici qu'à l'étude de ces grandeurs en fonction de n_{dl}, en non en fonction du nombre de processeurs (ce dernier point fera l'objet d'une étude sur la scalabilité parallèle du code à la section 5.4.2). Le but est de voir le comportement de la FMM pour n_{dl} grand, le parallélisme n'est qu'un outil pour permettre de pousser cette étude plus loin.

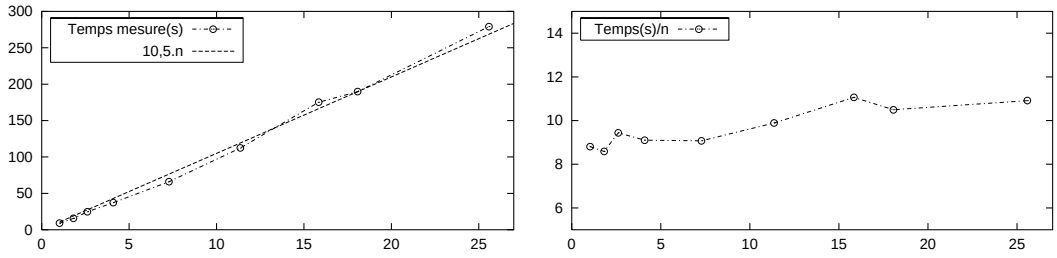
Nous allons réaliser nos tests sur la Silicon Graphics Origin3800 dans une configuration à 16 processeurs. Pour des sphères de taille croissante, comportant de 1.023.168 à 25.579.200 inconnues et dont le diamètre varie de 30 à 150 longueurs d'onde, on va étudier les temps d'exécution d'un produit matrice-vecteur, les temps de démarrage du calcul, la mémoire (vive et disque) consommée, et la précision. Contrairement au cas séquentiel, où nous étions seul

utilisateur de la machine de test, nous allons ici utiliser des ressources partagées (comme la mémoire vive, le réseau de communication, le disque). C'est pourquoi les temps mesurés risquent d'être « bruités » et donc difficiles à interpréter.

5.4.1.2 Temps

Lors de l'analyse de la scalabilité numérique séquentielle, nous avions séparé les différentes phases du produit matrice-vecteur multipôle en fonction des estimations théoriques. Nous étions alors arrivés au résultat que pour les tailles de problèmes considérées (jusqu'à 1,2 millions d'inconnues) tous les temps croissaient en $\mathcal{O}(n_{dl})$. Pour cette étude sur 16 processeurs, nous allons reprendre le même type de partage. Dans toute la suite, on note n_{Mdl} le nombre de millions d'inconnues. Il varie de 1 à 25,6.

5.4.1.2.1 Initialisations, intégrations, interactions proches Pour ces trois types de tâches élémentaires qui s'effectuent au niveau des feuilles, on prévoit un temps d'exécution en $\mathcal{O}(n_{Mdl})$. La courbe 5.5a donne les temps obtenus (en seconde) sur 16 processeurs en fonction du nombre d'inconnues n_{Mdl}. La courbe 5.5b représente ce même temps divisé par n_{Mdl}. Cette dernière semble se stabiliser autour d'une valeur de 10,5 d'où le temps $t \approx 10{,}5 n_{Mdl}$.

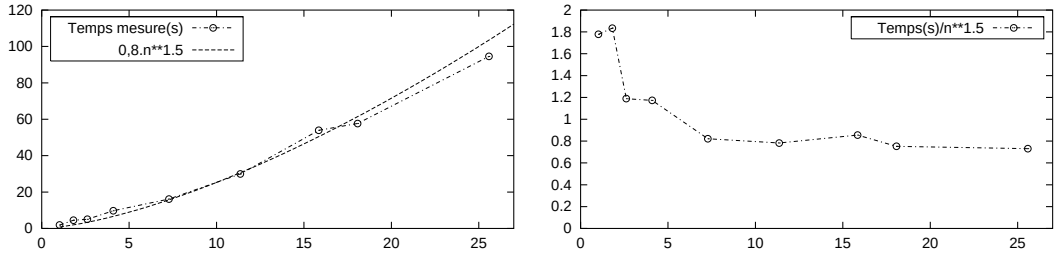


(a) Temps (en s) (b) Temps divisé par n_{Mdl}

FIG. 5.5 – *Durée cumulée des initialisations + intégrations + interactions proches*

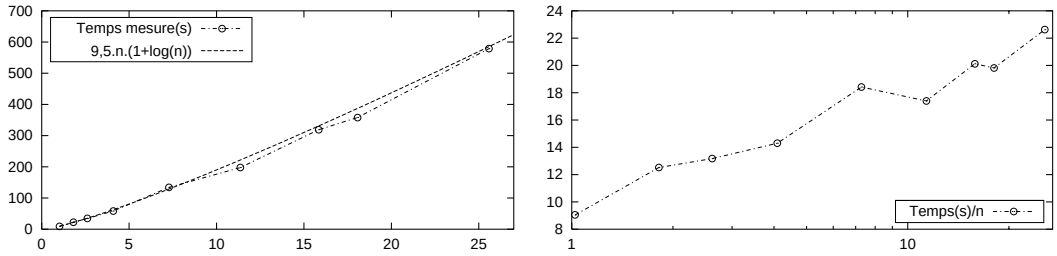
5.4.1.2.2 Montées et Descentes Pour ces tâches, la théorie prévoit un temps en $\mathcal{O}(n_{Mdl}^{3/2})$ pour les extrapolations/réductions, et un temps en $\mathcal{O}(n_{Mdl} \log n_{Mdl})$ pour les translations. Les courbes 5.6a et 5.6b donnent respectivement le temps et le temps divisé par $n_{Mdl}^{3/2}$ en fonction de n_{Mdl}. Graphiquement, on trouve un temps $t \approx 0{,}8 n_{Mdl}^{3/2}$.

5.4.1.2.3 Transferts Pour ces tâches aussi, on prévoit une croissance des temps de calcul en $\mathcal{O}(n_{Mdl} \log n_{Mdl})$. On a représenté ces temps sur la figure 5.7a. Sur la figure 5.7b, on a tracé ce temps divisé par n_{Mdl} en échelle logarithmique sur l'axe des abscisses, ce qui donne des points qui semblent s'accumuler autour d'une droite d'équation $y = 9{,}5(1 + x)$. On en déduit le temps pour les transferts : $t \approx 9{,}5 n_{Mdl}(1 + \log_{10} n_{Mdl})$.



(a) Temps (en s) (b) Temps divisé par $n_{Mdl}^{3/2}$

FIG. 5.6 – *Durée des montées et descentes*

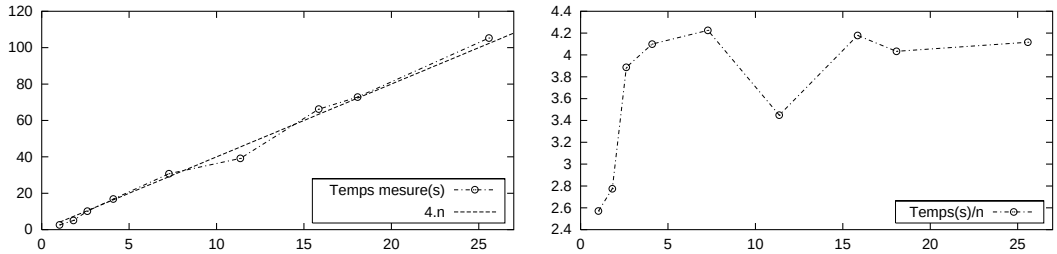


(a) Temps (en s) (b) Temps divisé par n_{Mdl}

FIG. 5.7 – *Durée des transferts*

5.4.1.2.4 Communications et barrières On mesure ici le temps passé dans les phases de communications. Chaque étape de communication est précédée d'une opération appelée barrière destinée à synchroniser les processeurs. Même s'il ne s'agit pas a proprement parler de communication, on mesure également le temps passé dans ces barrières. On évalue ainsi le temps perdu dans le parallélisme en fonction du nombre de degrés de liberté. Notons que les temps de communication et de barrière sont par défaut intégrés dans les autres temps mesurés. Nous les étudions ici séparément, mais nous ne les rajouterons pas au total à la fin, car ce serait les prendre en compte deux fois.

Sur la courbe 5.8b, le temps divisé par n_{Mdl} semble stagner, donc on en déduit $t \approx 4n_{Mdl}$. Vu que d'autres parties du code croissent plus vite que cela, on en déduit que les communications vont prendre de moins en moins de temps comparativement au reste pour n_{Mdl} grand.



(a) Temps (en s) (b) Temps divisé par n_{Mdl}

FIG. 5.8 – *Durée des communications*

5.4.1.2.5 Temps cumulé Pour finir, on somme les résultats obtenus dans les paragraphes précédents pour obtenir le temps total d'exécution d'un produit multipôle sur 16 processeurs d'Origin3800. D'après les paragraphes précédents, ce temps vaut :

$$t \approx 20n_{Mdl} + 0{,}8n_{Mdl}^{3/2} + 9{,}5n_{Mdl}\log_{10} n_{Mdl} \tag{5.1}$$

La figure 5.9 montre un très bon accord entre les temps mesurés et ce temps approximatif. Il est également remarquable de retrouver les trois termes prévus par la théorie (section 1.3.2.3.6), et notamment le terme en $n_{Mdl}^{3/2}$ issu des extrapolations et des réductions. Dans le tableau 5.9, on étudie l'importance relative des trois termes de l'équation 5.1 en fonction du nombre de millions d'inconnues. On voit que pour notre plus gros test $n_{Mdl} = 25$, le terme en $n_{Mdl}^{3/2}$ ne représente que 10 % du total, et c'est le terme en n_{Mdl} qui domine. Quand n_{Mdl} augmente, l'importance relative du $n_{Mdl}^{3/2}$ augmente. Un point intéressant est alors de savoir pour quelle valeur de n_{Mdl} ce terme devient prépondérant : il atteindrait 50 % pour $n_{Mdl} \approx 5000$ (soit 5 *milliards* d'inconnues). A la section 1.3.2.3.6, la théorie prévoyait que le terme en $n_{Mdl}^{3/2}$ deviendrait prépondérant pour environ 10 millions d'inconnues. Les tests réalisés ici repoussent semble-t-il cette limite bien plus loin.

5.4.1.2.6 Démarrage On étudie ici le temps (en minutes) nécessaire au démarrage d'un calcul multipôle. Cela inclut la relecture du maillage (qui peut être très longue : dans le cas

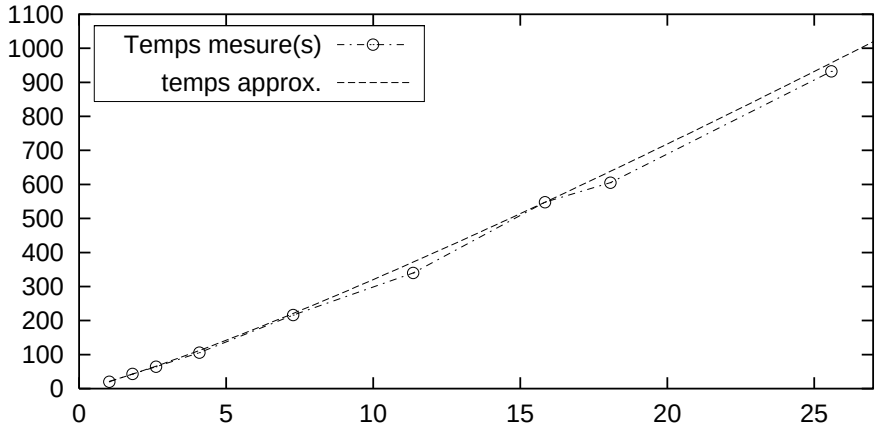
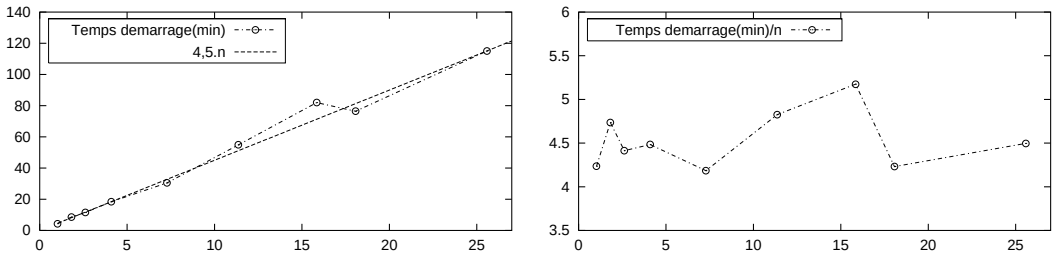


FIG. 5.9 – *Durée totale d'un produit multipôle*

Complexité	$n_{Mdl} = 25$	$n_{Mdl} = 100$	$n_{Mdl} = 500$	$n_{Mdl} = 5000$
$\mathcal{O}(n_{Mdl})$	54 %	43 %	32 %	18 %
$\mathcal{O}(n_{Mdl} \log n_{Mdl})$	36 %	40 %	40 %	31 %
$\mathcal{O}(n_{Mdl}^{3/2})$	10 %	17 %	28 %	51 %

TAB. 5.9 – *Part des différents termes dans l'équation 5.1*

à 18 millions d'inconnues, le fichier d'entrée pèse 2,4 Go), la création de l'octree et l'assemblage de la matrice des interactions proches. A priori, la phase de création de l'octree est en $\mathcal{O}(n_{Mdl} \log n_{Mdl})$, et les deux autres sont en $\mathcal{O}(n_{Mdl})$. Les résultats sont tracés sur la courbe 5.10a et le temps divisé par le nombre de millions d'inconnues sur la figure 5.10b.



(a) Temps (en s)

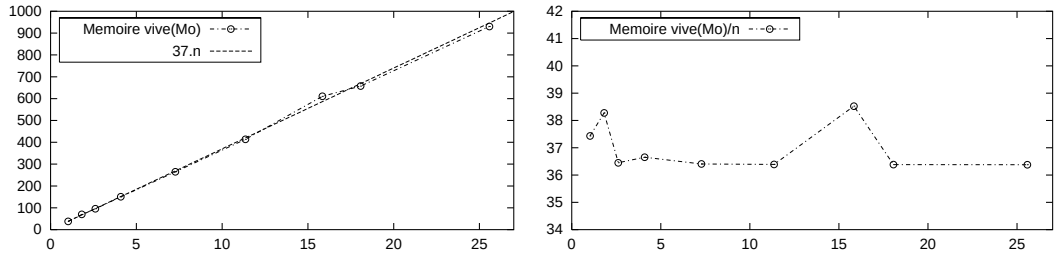
(b) Temps divisé par n_{Mdl}

FIG. 5.10 – *Démarrage d'un calcul multipôle*

Nous avons ici la parfaite illustration du problème de bruitage des mesures de temps évoqué en début de section. C'est particulièrement exacerbé ici dans la mesure où on utilise énormément de mémoire vive et d'accès disque. Les courbes sont donc très bruitées mais graphiquement, on trouve un temps $t \approx 4{,}5.n_{Mdl}$ où t est en minutes. Il ne semble pas y avoir de croissance en $\mathcal{O}(n_{Mdl} \log n_{Mdl})$.

5.4.1.3 Mémoire

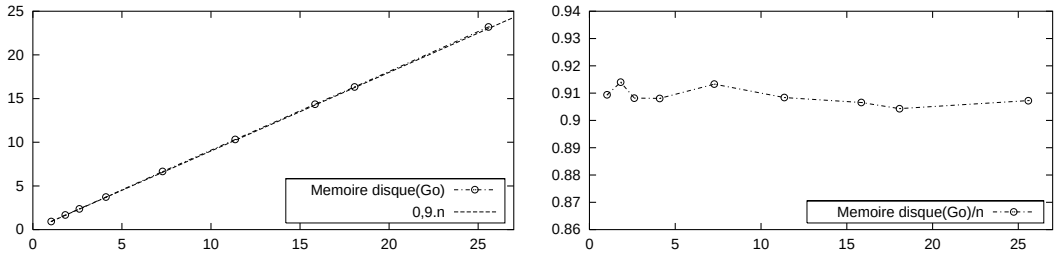
5.4.1.3.1 Mémoire vive On s'intéresse à la mémoire vive maximum consommée par le processeur 0 au cours du calcul complet. Même si le résultat peut varier légèrement d'un processeur à l'autre, la tendance dégagée serait la même si on choisissait de suivre un autre processeur. On observe une croissance linéaire, donnant une mémoire vive (en Mo) égale à $37.n_{Mdl}$.



(a) Mémoire vive (en Mo) (b) Mémoire vive divisée par n_{Mdl}

FIG. 5.11 – *Mémoire vive utilisée par un calcul multipôle*

5.4.1.3.2 Mémoire disque Regardons maintenant la quantité d'espace disque utilisé par le programme à titre temporaire pour stocker les données sur le maillage, sur l'octree, les fonctions de transferts, les interactions proches, les vecteurs. Tous les calculs étant in-core, il n'y a aucune fonction de radiation écrite sur disque. Les chiffres de la figure 5.12 sont donnés en gigaoctets, en fonction du nombre de millions d'inconnues. Ici aussi on observe une croissance linéaire, donnant une mémoire disque égale à $0{,}9.n_{Mdl}$.



(a) Espace disque (en Go) (b) Espace disque divisé par n_{Mdl}

FIG. 5.12 – *Espace disque utilisé par un calcul multipôle*

5.4.1.4 Précision

Enfin, terminons ce tour d'horizon des performances du code par l'analyse de la précision du calcul multipôle (calculée comme expliqué à la section 3.4.1.3). Le résultat est présenté sur

la figure 5.13. On voit que la précision reste la même quand le nombre d'inconnues augmente.

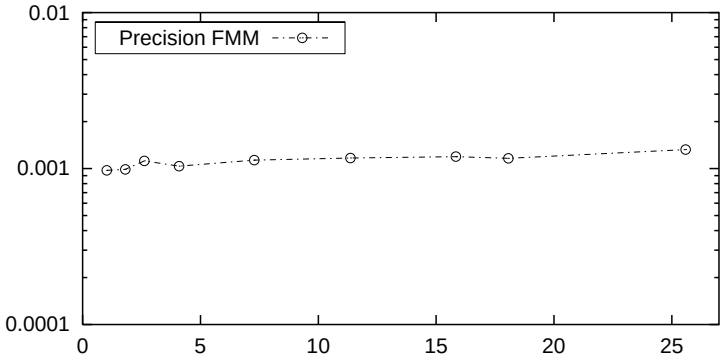


FIG. 5.13 – *Précision du produit multipôle*

5.4.1.5 Synthèse

Le tableau 5.10 présente un résumé des besoins en terme de temps et de mémoire (par processeur) pour la réalisation d'un calcul multipôle sur 16 processeurs d'origin3800 en fonction du nombre de millions de degrés de liberté.

Mémoire :	
Vive	37 Mo par Mdl
Disque	0,9 Go par Mdl
Temps :	
Itérations (s)	$20n_{Mdl} + 0{,}8n_{Mdl}^{3/2} + 9{,}5n_{Mdl}\log_{10} n_{Mdl}$
Démarrage (min)	$4{,}5.n_{Mdl}$

TAB. 5.10 – *Scalabilité de la FMM sur 16 processeurs : récapitulatif*

5.4.2 Scalabilité parallèle

Nous allons maintenant étudier l'impact du nombre de processeurs sur les performances de notre code multipôle.

5.4.2.1 Airbus1M sur PC biprocesseur

Regardons tout d'abord le cas d'une station biprocesseur traitant un cas relativement conséquent : en l'occurrence, un airbus de diamètre 101 longueurs d'onde et portant 1,16 millions d'inconnues. La machine comporte 1 Go de RAM, donc pour laisser de la place aux données autre que l'octree, la mémoire allouée par l'utilisateur aux fonctions de radiations sera de 600 Mo en monoprocesseur, et de 300 Mo en biprocesseur. L'octree complet pesant 706,2 Mo, le calcul sera out-of-core dans les deux cas. Nous avons résumé les résultats dans le tableau 5.11.

Analysons ces résultats. Concernant la mémoire vive, les chiffres sont conformes à nos attentes, puisqu'on observe bien une division par 2. Et même un peu plus, car du fait de la

	Monoprocesseur	Biprocesseur	Efficacité
Mémoire :			
Vive	707.9 Mo	326.5 Mo	108 %
Disque	6547.1 Mo	3977.8 Mo	82 %
Temps :			
Itérations (s)	717.85 s	502.20 s	71 %
Démarrage (min)	57.10 min.	44.35 min.	64 %

TAB. 5.11 – *Airbus1M sur PC biprocesseur*

répartition des fonctions de radiation en un nombre entier de groupes (cf. section 2.5), si 1 groupe de permet pas de passer sous le seuil des 300 Mo de mémoire, 2 groupes permettent de passer largement en dessous. La mémoire disque n'est pas divisé par 2 car certaines données (comme le maillage) sont stockées deux fois : c'est là une caractéristique héritée de la programmation en mémoire distribuée.

Concernant les temps d'exécution, il ne faut pas être surpris du faible gain réalisé. En effet, dans une station de travail biprocesseur, les composants comme la mémoire vive ou le disque dur sont partagés entre les deux processeurs, et la bande passante vers ceux-ci est donc divisée par deux lorsque nos deux processus tentent d'y accéder simultanément. De plus, des problèmes plus pointus comme la cohérence entre les mémoires caches des deux processeurs ou le basculement des processus d'un processeur à l'autre peuvent grever la performance en biprocesseur (tous ces problèmes ne se posent pas sur une ferme de PC). Ceci dit, l'accélération reste convenable, et permet d'obtenir des temps d'exécution nettement inférieur pour un surcoût matériel raisonnable.

5.4.2.2 cetaf5M sur SP3

Regardons maintenant le cas Cetaf5M sur IBM SP3. Rappelons que ce maillage comporte 4,85 millions d'inconnues, et représente un objet de diamètre égal à 150 longueurs d'onde. La mémoire nécessaire pour stocker toutes les fonctions de radiation lors d'un calcul multipôle est de 2,3 gigaoctets. Sur la machine utilisée, avec 1 Go de mémoire vive disponible par noeud, on a fixé à 500 Mo le seuil de mémoire accordé à l'octree. C'est pourquoi les calculs sur 1 et 4 processeurs seront en mode out-of-core, tandis que les suivants seront en mode in-core. En pratique, on a mesuré le temps nécessaire à la réalisation d'un produit multipôle sur un nombre croissant de processeurs variant de 1 à 80. Les résultats sont exposés dans le tableau 5.12. On y indique, en fonction du nombre de processeurs, le temps réels en secondes pour un produit matrice-vecteur multipôle, l'efficacité parallèle (par rapport au cas monoprocesseur), et le pourcentage de ce temps consacré au calcul (CPU), au parallélisme (COMM) et aux accès disque(I/O).

Il convient de souligner que l'efficacité parallèle mesurée au-delà de 8 processeurs est biaisée par le fait que l'on compare un calcul in-core et un calcul monoprocesseur out-of-core. Cela explique notamment le score supérieur à 100 % obtenu avec 8 processeurs. Il faut donc voir ce score comme étant non pas l'efficacité parallèle de la FMM en générale, mais plutôt l'efficacité parallèle de notre implémentation, avec tout ce qu'elle a de particulier. Les tests sur SGI en adressage 64 bits en mode in-core de la section suivante seront de ce point de vue plus objectifs.

Ceci dit, force est de constater que l'on obtient des scores tout à fait satisfaisants. Par

Proc	Produit matrice-vecteur				
	temps	efficacité	% CPU	% COMM	% I/O
Out-of-core					
1	2884,1 s	-	67	-	33
4	728,8 s	99 %	72	17	11
In-core					
8	305,3 s	118 %	73	23	4
16	204,1 s	88 %	65	27	8
32	116,4 s	77 %	59	33	8
48	82,2 s	73 %	58	33	9
64	80,9 s	56 %	55	36	9
80	71,3 s	51 %	50	40	10

TAB. 5.12 – *Test de scalabilité pour le Cetaf5M sur IBM SP3*

exemple, une efficacité parallèle avec 80 processeurs de 51 % est un très bon score. Si on calcule cette efficacité par rapport au premier cas in-core à 8 processeurs, on obtient environ 40 %, ce qui reste convenable. Il faut souligner qu'avec 80 processeurs, l'octree (qui mesure en tout 2,3 Go) ne représente que 46 Mo de données par processeur, ce qui est très peu. Pour un tel cas de calcul à (environ) 5 millions d'inconnues, un nombre de processeurs de 40 semble être le maximum raisonnable. Nous reviendrons sur ce point à la section 5.4.2.4 consacrée au choix du nombre de processeurs.

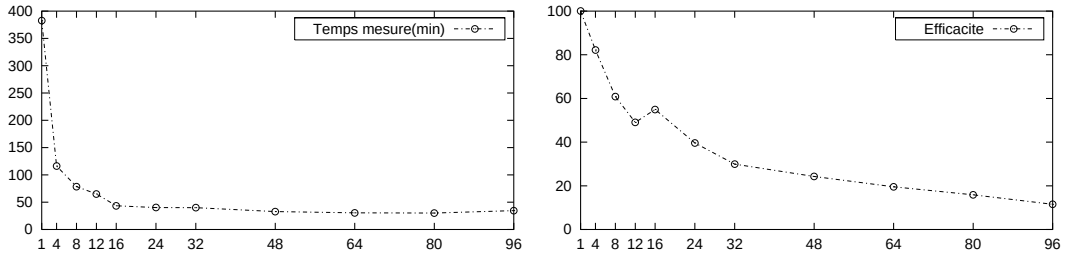
5.4.2.3 furtif4M sur Origin3800

Notre série de tests suivante va être réalisée sur Origin3800, en mode 64 bits. L'intérêt de ce choix est de nous ôter toute limitation entre terme de mémoire vive disponible ou accessible. On va donc pouvoir réaliser tous nos tests, y compris monoprocesseur, en mode in-core. Nous allons tester le cas furtif4M sur un nombre croissant de processeurs, et regarder en particulier le temps de démarrage du calcul (création des octrees et assemblage des matrices), la durée d'un produit multipôle, la mémoire vive et la mémoire disque consommées.

5.4.2.3.1 Temps de démarrage du calcul Le temps de démarrage du calcul inclus principalement deux parties :

- D'une part la relecture du maillage et la création de l'octree. Même si la création de l'octree se fait en parallèle (surtout pour économiser la mémoire), cette partie est très peu « scalable » en temps. Dans une formulation intégrale, chaque point interagit avec tous les autres, et donc chaque processeur a inévitablement tôt ou tard besoin de connaître la totalité du problème. C'est pourquoi cette phase du calcul accélérera peu en augmentant le nombre de processeurs.
- D'autre part l'assemblage de la matrice des interactions proches qui est a contrario très scalable.

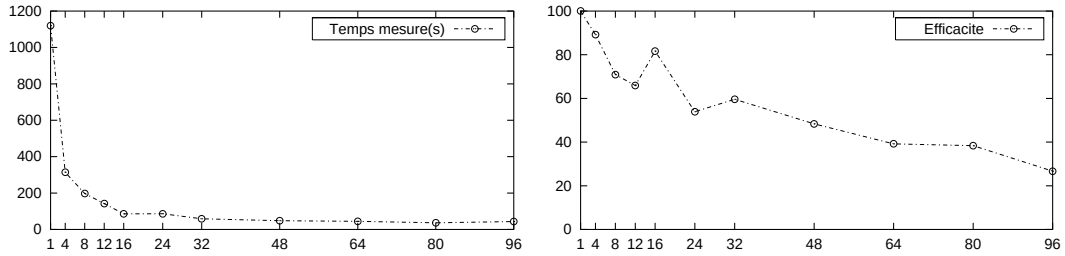
Les résultats numériques se trouvent sur la figure 5.14, avec les temps à gauche en minutes et l'efficacité correspondante à droite en pourcentage, dans les deux cas en fonction du nombre de processeurs. On voit que le temps ne passe pas en dessous d'un certain seuil, quel que soit le nombre de processeurs. Cela explique que l'efficacité tende vers 0 lorsque le nombre de processeurs augmente.



(a) Temps (en minutes) (b) Efficacité

FIG. 5.14 – *Temps de démarrage du calcul multipôle en fonction du nombre de processeurs*

5.4.2.3.2 Durée d'un produit multipôle Regardons maintenant la durée (en secondes) d'un produit multipôle complet. Ce temps n'est pas le temps CPU ou machine, c'est le temps réel *elapsed* tel qu'il serait mesuré par un utilisateur devant sa console. La figure 5.15a représente ce temps en fonction du nombre de processeurs, et l'efficacité par rapport au cas monoprocesseur est donnée sur la courbe 5.15b.



(a) Temps (en secondes) (b) Efficacité

FIG. 5.15 – *Durée d'un produit multipôle en fonction du nombre de processeurs*

Jusqu'à 48 processeurs, on reste au-dessus de 50 % d'efficacité, ce qui reste satisfaisant. Afin de savoir quelles parties de la FMM sont bien ou mal parallélisées, on a tracé sur la figure 5.16 les efficacités de chaque partie du calcul : initialisations, montées/descentes, transferts, intégrations, interactions proches.

Les courbes sont assez chahutées pour 24 processeurs et moins, sans doute à cause de la charge de la machine au moment du test. Il faudrait refaire ces calculs sur une machine dédiée (dont on serait le seul « occupant ») pour être certain que notre parallélisation n'est pas la cause de ces courbes accidentées, mais les résultats lisses obtenus pour 32 processeurs et plus nous incitent à penser que cela n'est pas le cas. Au hit-parade de l'efficacité, les phases d'initialisation et d'intégration arrivent en tête, avec un score parfois supérieur à 100 %. Il faut dire que ce calcul se fait sans aucune communication. Viennent ensuite les montées et les descentes, qui en pratique nécessitent très peu d'échanges de données. Arrivent enfin les interactions proches et surtout les transferts, qui réalisent le plus gros des communications

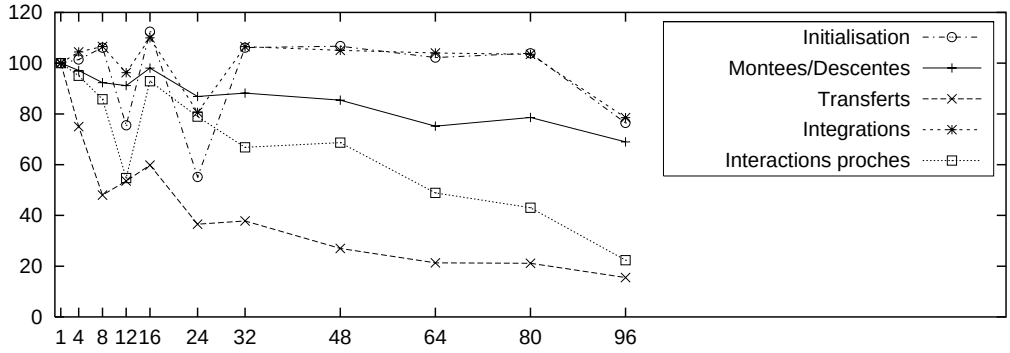
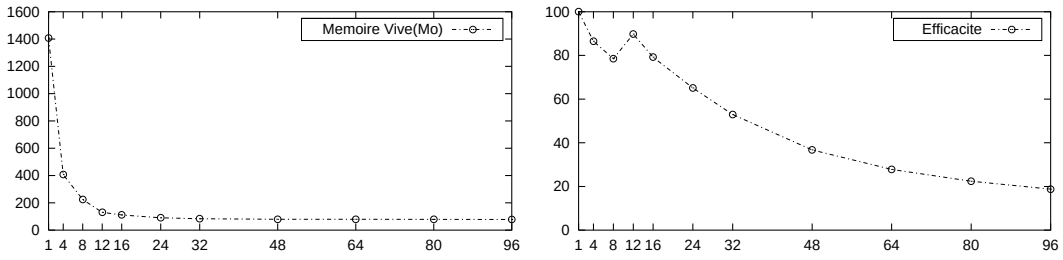


FIG. 5.16 – *Efficacité des différentes tâches élémentaires de la FMM*

et obtiennent donc les plus faibles scores d'efficacité : on passe en dessous de 50 % dès 24 processeurs. Un utilisateur pressé souhaitant donc réaliser des calculs de cette taille sur un très grand nombre de processeurs (plus de 100) devrait avant tout chercher à améliorer l'efficacité des phases de transfert.

5.4.2.3.3 Mémoire vive consommée Regardons la mémoire vive maximum consommée par un processeur au cours du calcul. On sait que l'efficacité ne pourra pas être optimale, puisque tout au long du calcul certaines données sont répliquées sur plusieurs processeurs. Néanmoins, sur la figure 5.17a, on peut observer que la courbe tend vers une asymptote horizontale située à environ 150 Mo, ce qui est très peu pour une machine actuelle (calculateur parallèle ou station de travail). On peut donc déplorer que l'efficacité ne soit pas meilleure, mais pour autant il ne faut pas s'en alarmer : la consommation en mémoire vive ne sera pas le facteur limitant (du moins pas sur un problème de cette taille).

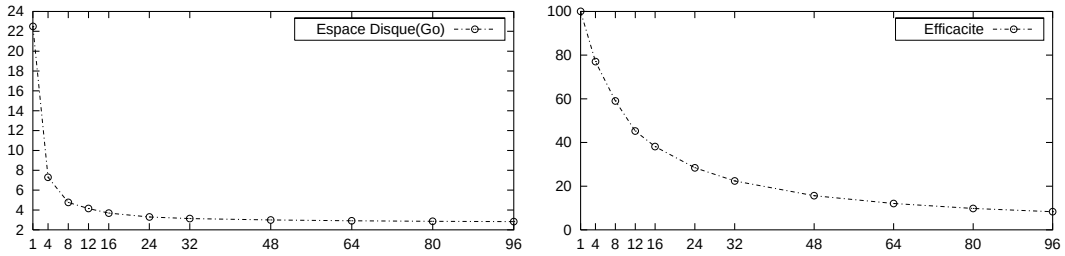


(a) Mémoire (b) Efficacité

FIG. 5.17 – *Mémoire vive maximum consommée au cours d'un calcul multipôle en fonction du nombre de processeurs*

5.4.2.3.4 Espace disque consommé Regardons enfin l'espace disque consommé par un processeur. Il se décompose en deux parties : d'une part le maillage de départ, dont la taille reste constante quel que soit le nombre de processeurs, d'autre part la matrice des interactions

proches et les données liées à l'octree, qui sont au contraire bien distribuées entre les processeurs. La partie non-distribuée explique la mauvaise efficacité observée sur la figure 5.18b.



(a) Disque (b) Efficacité

FIG. 5.18 – *Espace disque consommé au cours d'un calcul multipôle en fonction du nombre de processeurs*

Néanmoins comme pour la mémoire vive cela ne pose pas vraiment de problème, car les besoins en espace disque (environ 4 Go par processeurs) restent modestes et surtout très inférieurs à ceux des méthodes non-multipôles. Par exemple, une matrice complexe pleine de taille $3.924.000 \times 3.924.000$ répartie entre 24 processeurs nécessiterait 9560 Go d'espace disque par processeur (!).

5.4.2.4 Comment choisir le nombre de processeurs

Pour terminer cette section consacrée à la scalabilité parallèle, nous allons tenter d'établir un critère simple afin de trouver le nombre de processeurs optimum pour un calcul donné. En général, si on diminue le nombre de processeurs, on augmente l'efficacité du calcul, si on augmente ce nombre, on augmente sa vitesse. Il y a donc un compromis à faire, dont nous cherchons le critère. Un tel critère sera bien sûr purement indicatif, puisqu'il dépendra de la vitesse de calcul des processeurs, du débit du réseau de communication, de la quantité de mémoire disponible, et du cas de calcul traité. Néanmoins, on peut souligner les quelques points suivants :

- Il vaut mieux être in-core : cela ressemble à une lapalissade, cela veut surtout dire que le fait d'augmenter le nombre de processeurs peut aussi augmenter l'efficacité, si on passe du mode out-of-core au mode in-core (cf. par exemple le tableau 5.12).
- Pas besoin de puissances de 2 ou de carrés : souvent, les programmes parallélisés s'accommodent mieux de ces nombre de processeurs (car cela facilite le partage des données et des calculs). Rien de tel ici, il n'y a aucune contrainte sur le choix du nombre de processeurs. Si nous avons souvent choisis des multiples de 16, c'était pour s'adapter aux topologies des machines (la SP3 surtout, cf figure 5.4).
- Point trop n'en faut : Le cas du Cetaf5M sur SP3 montre qu'en passant de 32 à 80 processeurs (soit $\times$2,5), le temps de calcul d'une itération est seulement divisé par 1,6.

De manière indicative, on donne dans le tableau 5.13 un nombre a priori correct de processeurs en fonction du nombre d'inconnues pour obtenir une bonne efficacité parallèle quelle que soit la machine. D'après notre expérience, ce nombre donnera un résultat convenable en

toutes circonstances et pourra être pris comme premier choix. Bien sûr, il pourra ensuite être ajusté au cas par cas, en fonction du maillage, de la machine, du nombre de seconds membres, etc.

Nombre d'inconnues	Nombre de processeurs
10^3	1
10^4	6
10^5	12
10^6	20
10^7	50

TAB. 5.13 – *Nombre de processeurs en fonction de n_{dl}*

5.5 Calculs de grande taille

5.5.1 Sphère à 25 millions d'inconnues

Commençons par présenter des résultats sur une sphère, afin de vérifier la validité des résultats obtenus en les comparant aux séries de Mie. On utilise un maillage de sphère comportant 25.579.200 inconnues dont le diamètre est de 150 longueurs d'onde. On utilise un solveur GMRES (restart=20) pour résoudre le système linéaire issu de la formulation CFIE. 81 itérations sont nécessaires pour atteindre un résidu final inférieur à 10^{-5}. L'évolution du résidu est donnée sur la figure 5.19 : elle est parfaitement linéaire en échelle logarithmique (ce qui n'a rien de surprenant pour un maillage sphérique).

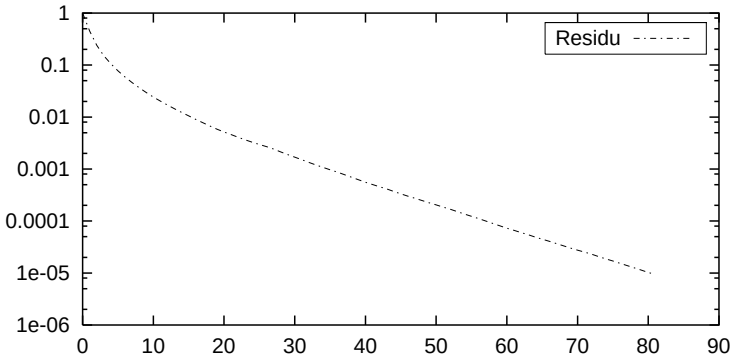
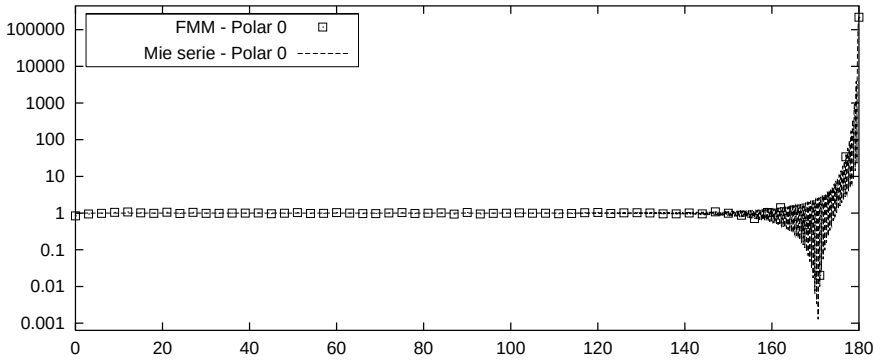
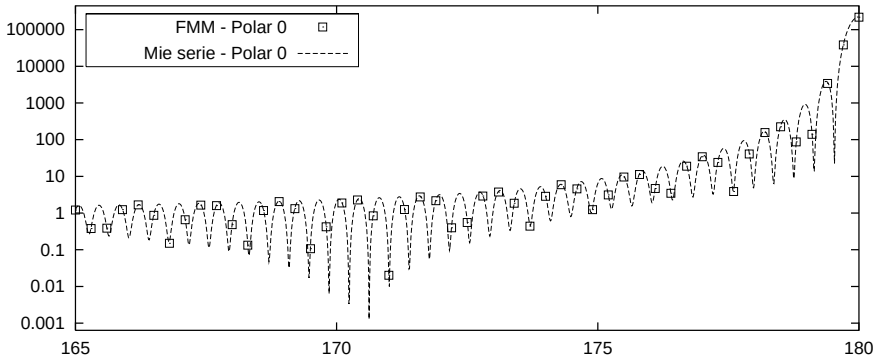


FIG. 5.19 – *Convergence du solveur GMRES pour la sphère 25M*

La SER en polarisation V-V est donnée sur la figure 5.20. La première courbe correspond à toutes les directions d'observation de 0 à 180 degrés, tandis que la seconde représente un zoom sur les seules directions comprises entre 165 et 180 degrés. Sur la première, on constate que même sur la partie « plate » de la courbe (où les niveaux d'énergie sont 5 ordres de grandeurs inférieurs au maximum) notre FMM est très précise. Sur la courbe du bas, on vérifie que sur la partie finale du tracé la FMM reste confondue avec la série de Mie.



(a) Courbe complète



(b) Courbe partielle

FIG. 5.20 – *SER bistatique en polarisation VV*

Ce test a été réalisé sur IBM SP3, sur 4 nœuds et 64 processeurs. Chaque produit matrice vecteur a duré environ 440 s dont 13 % utilisés pour le parallélisme (communications et barrières). L'efficacité parallèle du seul produit matrice-vecteur est donc supérieure à 80 %. Si on rapproche ce chiffre de ceux obtenus à la section sur la scalabilité parallèle (et en particulier le tableau 5.12), on remarque que sur un même nombre de processeurs (64), l'efficacité parallèle passe de environ 50 % à plus de 80 % lorsque le nombre d'inconnues passe de 5 à 25 millions. Le calcul de la SER a pris 45 minutes, et le calcul entier 18 heures.

5.5.2 Avions préconditionnés

5.5.2.1 Présentation

Nous allons réaliser une série de calculs sur la configuration Origin-16 décrite à la section 5.1.3, en utilisant un maillage d'airbus A318 comportant 1,16 millions d'inconnues, dans le but d'étudier la convergence de la résolution en fonction du solveur itératif, de la formulation et du préconditionnement. En effet, la méthode multipôle rapide permet de repousser beaucoup de limites (notamment celle du nombre d'inconnues), mais elle impose l'utilisation de solveurs itératifs sur des systèmes en général mal conditionnés, ce qui dévoile de nouvelles contraintes, comme le nombre d'itérations nécessaires pour converger.

Nous allons étudier un cas à un seul second membre, que nous allons résoudre avec :

- GMRES [10] (avec restart=20) ou TFQMR [20] comme solveur ;
- EFIE ou CFIE comme formulation (on a le choix car l'objet est fermé) ;
- SPAI ou rien comme préconditionneur.

Nous allons donc réaliser $2^3 = 8$ résolutions. A chaque fois, le vecteur de départ sera le vecteur nul et le critère d'arrêt sera un résidu normalisé inférieur à 10^{-5} ou un nombre d'itérations de 300 (le solveur s'arrête au premier des deux seuils atteint).

Le SPAI que nous utilisons ici est une variante de celui décrit dans [2]. L'idée de base est de rechercher une matrice approchée du problème sous une forme creuse en ne conservant que les interactions proches (selon un critère à fixer), puis de rechercher un inverse approché de cette matrice en utilisant le même masque et en résolvant des problèmes au moindre carré. La spécificité de notre SPAI est qu'il s'appuie sur des structures d'octree (similaires à celles utilisées par la FMM) pour déterminer les interactions proches à prendre en compte dans le calcul de la matrice d'interactions proches puis dans la matrice de préconditionnement. L'intérêt est double : d'une part grâce à la structure hiérarchique de l'octree la recherche des interactions proches est très rapide (en $\mathcal{O}(n_{dl} \log n_{dl})$ au lieu de $\mathcal{O}(n_{dl}^2)$ par une méthode brute), d'autre part on ne fait qu'une résolution au moindre carré par feuille de l'octree (au lieu de une par inconnue dans un SPAI classique). Des résultats plus détaillés sur les performances de ce préconditionneur peuvent être trouvés dans [9].

5.5.2.2 Résultats

La figure 5.21 montre les courbes de convergence pour chacun des 8 tests réalisés. Les courbes correspondant à des calculs préconditionnés sont en gras, alors que celles utilisant le solveur GMRES se reconnaissent à leur descente régulière par opposition à la descente par à-coups du solveur TFQMR.

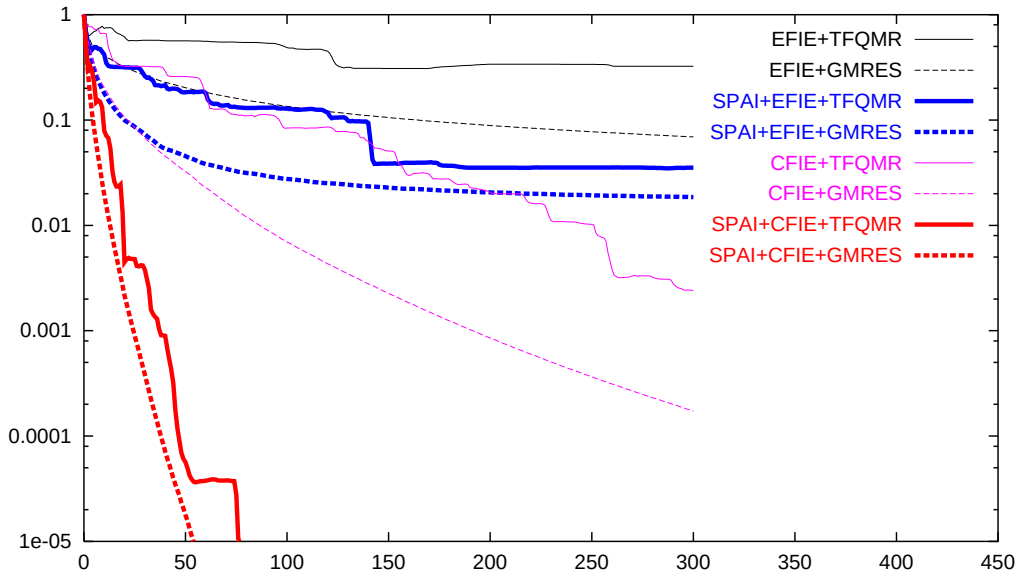


FIG. 5.21 – *Convergence des différents solveurs sur le cas de calcul airbus1M*

5.5.2.3 Interprétation

Tout d'abord, on peut remarquer que dans tous les cas le solveur GMRES est plus efficace que le solveur TFQMR. Il descend de manière plus régulière et plus rapide. Ensuite, remarquons que – comme prévu – la formulation CFIE converge mieux que l'EFIE, et que l'usage d'un préconditionneur SPAI accélère la convergence. Dans le tableau 5.14, on voit que le SPAI utilisé divise en moyenne par 5 le nombre d'itérations de GMRES nécessaires pour atteindre un résidu donné quelle que soit la formulation utilisée. Si la formulation de départ converge vraiment très mal, comme l'EFIE, la même formulation préconditionnée par SPAI ne convergera guère mieux. On voit que la convergence CFIE+SPAI est vraiment excellente. Malheureusement, elle nécessite un maillage fermé pour s'appliquer.

Résidu	EFIE	EFIE+SPAI	CFIE	CFIE+SPAI
0.5	6	3	3	1
0.2	52	9	11	3
0.1	165	20	22	5
0.05	> 300	45	38	7
0.02		219	65	10
0.01		> 300	87	13
0.005			114	17
0.002			156	21
0.001			192	25
0.0005			231	29
0.0002			291	34
0.0001			> 300	38

TAB. 5.14 – *Nombre d'itérations nécessaires pour atteindre un résidu normalisé donné*

Regardons maintenant les temps d'exécution de ces calculs dans le tableau 5.15. Le temps d'assemblage de la matrice des interactions proches de la FMM est un peu plus lent en CFIE qu'en EFIE (car cette dernière comporte moins d'intégrales à calculer et est en plus symétrique). En revanche pour l'assemblage du SPAI ces temps s'équilibrent, car le plus gros du calcul consiste à réaliser des factorisation QR pour résoudre des problèmes au moindre carré. Pour ce qui est des temps d'itération, la CFIE est là aussi un peu plus lente, mais le supplément dû au préconditionnement est négligeable. Ce qui est important, c'est de voir que l'assemblage du SPAI coûte un temps équivalent à environ 300 itérations, ce qui est raisonnable compte-tenu des gains apportés (cf. tableau 5.14). La rentabilité du SPAI serait encore meilleure si nous avions plusieurs seconds membres à traiter.

Temps	EFIE	CFIE
Assemblage		
du SPAI	11950.0	12179.9
de la matrice FMM	140.7	432.3
Itération	42,6	43,5
Produit SPAI	2,5	2,5

TAB. 5.15 – *Durée (en secondes) des différentes étapes du calcul*

On voit que – pour ce calcul – l'utilisation conjointe du SPAI et de la formulation CFIE offre une solution qui converge très rapidement vers la solution. Certes, l'assemblage du SPAI est long, mais la matrice SPAI obtenue peut être conservée sur disque pour un usage ultérieur (dans le cas de l'airbus1M, cette matrice SPAI pèse globalement 2,5 Go).

5.5.3 Cetaf préconditionné

Un cetaf maillé avec 13.477.500 inconnues est illuminé par une onde plane incidente de fréquence 150 GHz. La longueur d'onde est de 2 millimètres, et la plus grande dimension de l'objet (50 cm) vaut alors 250 longueurs d'onde. Les éléments du cas tests sont présentés sur la figure 5.22 : l'onde plane incidente arrive de la direction ($\theta = 45$,$\phi = 180$), et l'on s'intéresse au champ lointain diffracté dans toutes les directions du plan (xOz) : $\phi = 180$ et θ varie de 0 à 360 degrés par pas de 0,2.

Pour résoudre ce problème, on utilise une formulation CFIE (applicable car le Cetaf est un objet fermé) préconditionnée par un SPAI [2] et résolue par un solveur itératif GMRES [10] (de valeur de restart égale à 20). Le résidu normalisé final recherché est de 10^{-2}, ce qui est en général suffisant pour obtenir des champs lointains convergés. La figure 5.23 montre la convergence du solveur en 37 itérations seulement. Il faut noter que si on poursuit les itérations au-delà d'un résidu de 5.10^{-3} la convergence devient très difficile, et que sans préconditionnement ou sans CFIE la convergence est quasiment impossible. Néanmoins, avec la combinaison GMRES+CFIE+SPAI retenue ici, la convergence est satisfaisante.

On utilise une FMM basée sur un octree à 11 niveaux, comportant 1.069.293 feuilles et 1.430.486 cellules en tout. Le calcul est exécuté sur 64 processeurs de SP3. L'octree, dont la taille totale est de 6476,5 Mo, occupe alors 115,4 Mo par processeurs. Le calcul est traité in-core. En terme d'usage du disque, environ 280 Go sont utilisés pour l'ensemble des 64 processeurs.

Le calcul complet dure 7,8 heures qui se décomposent en :

– 1,4 heures pour créer l'octree et les autres structures de la FMM ;

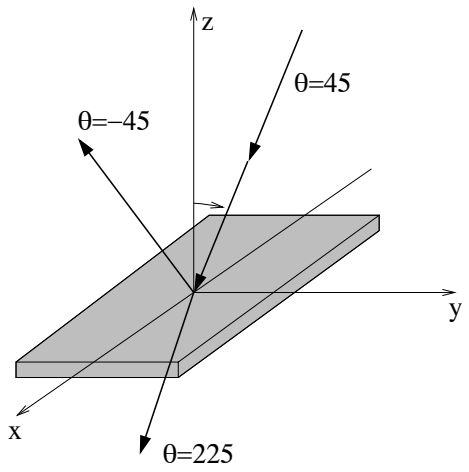


FIG. 5.22 – *Cas Test : Cetaf à 13,5 millions d'inconnues*

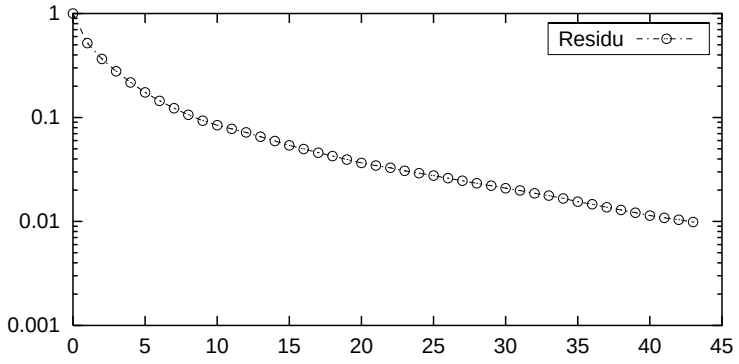


FIG. 5.23 – *Convergence du solveur GMRES pour le cetaf préconditionné*

– 0,5 heures pour assembler la matrice des interactions proches ;
– 2,0 heures pour assembler le SPAI ;
– 3,3 heures pour la résolution du système par le solveur GMRES ;
– 0,6 heure pour le calcul de la SER.

Chaque itération dure 187 secondes, correspondant à :
– 58 % de calcul ;
– 22 % d'accès disque ;
– 11 % de communications ;
– 9 % de barrière.

On voit que la partie liée au parallélisme (communication et barrière) représente 20 % du temps total d'une itération. Par conséquent, l'efficacité parallèle de cette partie du code est de 80 %. Si on regarde comment se décompose le temps d'une itération multipôle en fonction des tâches élémentaires, on obtient :
– 15 % pour les initialisations ;
– 7 % pour les montées/descentes ;
– 29 % pour les transferts ;
– 22 % pour les intégrations ;
– 27 % pour les interactions proches.

Remarquons que la partie qui est supposée être (asymptotiquement) la plus coûteuse – à savoir les montées/descentes, en $\mathcal{O}(n_{dl}^{3/2})$ – est ici la plus rapide (c'est à rapprocher du résultat de la section 5.4.1.2.5). Mais il est vrai que le niveau plafond étant le niveau 4, les montées/descentes les plus coûteuses aux niveaux les plus hauts ne sont pas réalisées.

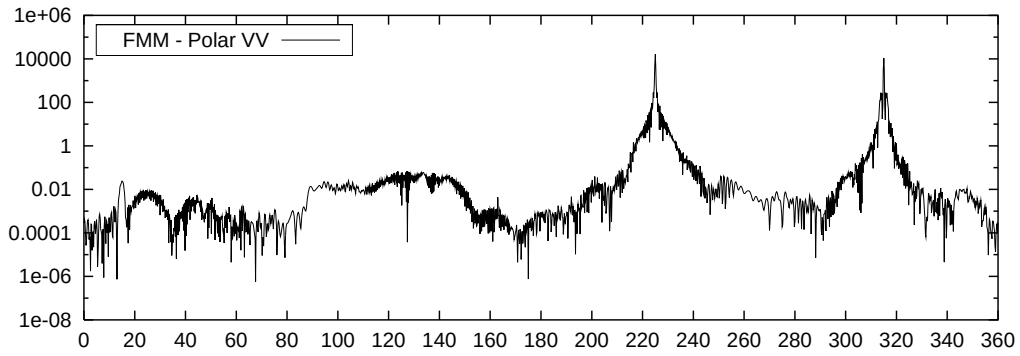


FIG. 5.24 – *SER bistatique*

La figure 5.24 est le résultat final du calcul, à savoir la SER dans le plan (xOz) où $\phi = 180$. Conformément à la figure 5.22, on s'attend à observer un premier pic dans la direction $\theta = -45$ correspondant à l'onde réfléchie géométriquement, et un second pic dans la direction $\theta = 225$ correspondant à la zone d'ombre, dans laquelle le champ total est proche de zéro, et donc le champ diffracté est opposé au champ incident, avec une amplitude quasiment égale. On retrouve bien ces deux pics sur la courbe. Pour le reste du tracé, nous attendons d'avoir des résultats équivalents issus d'un code d'approximation géométrique haute fréquence pour analyser la pertinence et la validité des résultats. Mais en première approximation, nos courbes sont très satisfaisantes.

Conclusion

Dans ce chapitre nous avons tout d'abord souhaité analyser les performances parallèles de notre code sur des cas de taille raisonnable. Par rapport au cas séquentiel, il y a quelques paramètres nouveaux dont nous avons déterminé les valeurs optimales. Nous avons ensuite étudié les scalabilités numériques et parallèles de notre FMM, et ainsi établit une complexité en $\mathcal{O}(n_{dl} \log n_{dl})$ et une très bonne efficacité parallèle. Nous avons terminé ce chapitre par la réalisation de quelques cas de calcul de très grande taille, établissant par la même un nouveau record en la matière (25,6 millions d'inconnues). Nous estimons cependant n'avoir pas encore totalement exploité le potentiel de la méthode multipôle. En particulier, l'utilisation de solveur tirant partie de la précision ajustable de la FMM semble prometteuse.

Troisième partie

Applications de la méthode multipôle rapide

Chapitre 6

Acoustique

Sommaire

Introduction **223**
6.1 Cas d'une onde incidente **223**
6.2 Cas d'un guide d'ondes **235**
Conclusion **246**

Introduction

On s'intéresse à la résolution de l'équation de Helmholtz dans un domaine non borné de l'espace par la méthode des équations intégrales. On se propose d'utiliser les développements réalisés autour de la méthode multipôle pour accélérer la résolution itérative de ce type de système linéaire. Dans un premier temps, on cherchera à résoudre un problème acoustique en présence d'une onde incidente. Dans un second temps, on traitera le cas d'un guide d'ondes. A chaque fois, on présentera succinctement l'écriture « classique » (c'est-à-dire non multipôle), puis on introduira les nouvelles formulations multipôles adaptées à chacun de ces problèmes. Par rapport à la FMM écrite dans le cas électromagnétique, seuls quelques modules (en particulier ceux d'initialisation et d'intégration) ont dû être modifiés, mais nous verrons apparaître un éclairage nouveau sur certains points de la méthode multipôle.

6.1 Cas d'une onde incidente

La présentation théorique que l'on va faire ici constitue juste un rappel destiné à introduire les développements spécifiquement FMM. Pour plus de détails, se reporter par exemple à [17].

6.1.1 Modèle physique considéré

On se donne un objet quelconque Ω^- régulier (cf. figure 6.1) se trouvant dans un fluide parfait barotrope (le plus souvent de l'air, éventuellement de l'eau). Une onde plane acoustique incidente u_{inc} de fréquence f et de pulsation $\omega = 2\pi f$ est diffractée par l'objet. On cherche à calculer le champ ainsi diffracté u_{diff} dans le domaine extérieur noté Ω^+. La frontière du domaine, notée S, est constituée d'une partie traitée Γ, d'impédance η, et d'une partie rigide

Σ. On a $S = \Gamma \cup \Sigma$. On note $\vec{\nu}$ la normale unitaire *sortante* en tout point de la frontière de l'objet.

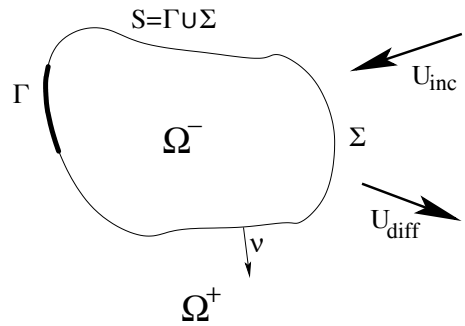


FIG. 6.1 – *Problème acoustique : cas d'une onde incidente*

6.1.2 Modélisation mathématique

6.1.2.1 Equations de Helmholtz

Dans le domaine non-borné Ω^+, on cherche à résoudre le problème (P^+) suivant : trouver $u^+ \in H^1_{loc}(\Omega^+)$ tel que

$$(P^+) \left\{ \begin{array}{ll} \Delta u^+ + k^2 u^+ = 0 & \textit{dans } \Omega^+, \\ \dfrac{\partial u^+}{\partial \nu} = -\dfrac{\partial u_{inc}}{\partial \nu} & \textit{sur } \Sigma, \\ \dfrac{\partial u^+}{\partial \nu} + i\dfrac{k}{\eta} u^+ = -\left(\dfrac{\partial u_{inc}}{\partial \nu} + i\dfrac{k}{\eta} u_{inc}\right) & \textit{sur } \Gamma, \\ \lim\limits_{r \to +\infty} r\left(\dfrac{\partial u^+}{\partial r} - iku^+\right) = 0 & \end{array} \right.$$

On note c la célérité des ondes de pression dans Ω^+ (typiquement $c = 340\ m/s$ dans le cas de l'air), et $k = \omega/c$ le nombre d'onde.

On note $G(x,y)$ la fonction de Green solution de l'équation $\Delta u + k^2 u = -\delta_0$ associée à la condition de radiation apparaissant dans le problème (P^+) :

$$G(x,y) = \frac{e^{ik\|x-y\|}}{4\pi\|x-y\|}$$

A ce problème (P^+), on choisit d'associer le problème (P^-) dans le domaine intérieur Ω^- défini par : trouver $u^- \in H^1(\Omega^-)$ tel que

$$(P^-) \left\{ \begin{array}{ll} \Delta u^- + k^2 u^- = 0 & \textit{dans } \Omega^-, \\ \dfrac{\partial u^-}{\partial \nu} = -\dfrac{\partial u_{inc}}{\partial \nu} & \textit{sur } \Sigma, \\ \dfrac{\partial u^-}{\partial \nu} - i\dfrac{k}{\eta} u^- = -\left(\dfrac{\partial u_{inc}}{\partial \nu} - i\dfrac{k}{\eta} u_{inc}\right) & \textit{sur } \Gamma \end{array} \right.$$

6.1.2.2 Représentation intégrale

Les deux problèmes (P^+) et (P^-) étant posés, nous allons dans cette section présenter les résultats du théorème de représentation qui permet de déterminer les solutions u^+ et u^-. Pour cela, on note ϕ et p les sauts des traces tangentielles de u et de $\partial u/\partial\nu$, plus exactement :

$$\begin{cases} \phi(x) = u^-_{|S}(x) - u^+_{|S}(x) & x \in S, \\ p(x) = \left(\dfrac{\partial u^-}{\partial \nu}\right)_{|S}(x) - \left(\dfrac{\partial u^+}{\partial \nu}\right)_{|S}(x) & x \in S \end{cases}$$

On sait que l'on a $\phi \in H_S^{1/2}$ et $p \in H_S^{-1/2}$. A noter que compte tenu des conditions aux limites imposées sur Σ, p y est nul. La solution à notre problème alors peut s'écrire sous la forme suivante en tout point x de l'espace (voir [29]) :

$$u(x) = \int_S \left(G(x,y)p(y) - \frac{\partial G(x,y)}{\partial \nu_y}\phi(y) \right) dy \tag{6.1}$$

Autrement dit, la connaissance de p et ϕ sur la frontière de l'objet Ω^- résout entièrement notre problème.

On peut également exprimer avec p et ϕ les traces de u et $\partial u/\partial\nu$ sur S. Pour cela, on introduit les opérateurs suivants :

$$\begin{cases} S\phi(x) = \displaystyle\int_S G(x,y)\phi(y)dy, \\ K\phi(x) = \displaystyle\int_S \frac{\partial G(x,y)}{\partial \nu_x}\phi(y)dy, \\ K'p(x) = \displaystyle\int_S \frac{\partial G(x,y)}{\partial \nu_y}p(y)dy, \\ Dp(x) = \displaystyle\oint_S \frac{\partial^2 G(x,y)}{\partial \nu_x \partial \nu_y}p(y)dy \end{cases} \tag{6.2}$$

$\oint$ est une partie finie au sens de Hadamard de l'intégrale singulière. On peut alors exprimer les traces u et $\partial u/\partial\nu$ sur S à partir de p et ϕ :

$$\begin{cases} u^+ = Sp - (\dfrac{I}{2} + K')\phi, \\ u^- = Sp + (\dfrac{I}{2} - K')\phi, \\ \dfrac{\partial u^+}{\partial \nu} = (-\dfrac{I}{2} + K)p - D\phi, \\ \dfrac{\partial u^-}{\partial \nu} = (\dfrac{I}{2} + K)p - D\phi \end{cases} \tag{6.3}$$

Par ailleurs, les conditions aux limites des problèmes (P^+) et (P^-) peuvent être réécrites de la manière suivante :

$$\begin{cases} \dfrac{\partial u^+}{\partial \nu} = -\dfrac{\partial u_{inc}}{\partial \nu} & sur\ \Sigma, \\ \dfrac{\partial u^+}{\partial \nu} + \dfrac{\partial u^-}{\partial \nu} + i\dfrac{k}{\eta}(u^+ - u^-) = -2\dfrac{\partial u_{inc}}{\partial \nu} & sur\ \Gamma, \\ \dfrac{\partial u^+}{\partial \nu} - \dfrac{\partial u^-}{\partial \nu} + i\dfrac{k}{\eta}(u^+ + u^-) = -2i\dfrac{k}{\eta}u_{inc} & sur\ \Gamma \end{cases} \tag{6.4}$$

On va utiliser le résultat du théorème de représentation (6.3) pour reformuler les conditions aux limites (6.4). On obtient le système d'équations suivant :

$$\begin{cases} -D\phi + Kp = -\dfrac{\partial u_{inc}}{\partial \nu} & sur\ \Sigma, \\ -D\phi + Kp - i\dfrac{k}{2\eta}\phi = -\dfrac{\partial u_{inc}}{\partial \nu} & sur\ \Gamma, \\ -\dfrac{\eta}{2ik}p + Sp - K'\phi = -u_{inc} & sur\ \Gamma \end{cases}$$

On opère un changement de variable en posant $\lambda = -p/ik$. On introduit une formulation variationnelle à l'aide de fonctions test ϕ^t (pour les deux premières équations) et λ^t (pour la troisième). On obtient finalement le système à résoudre : trouver (ϕ,λ) tels que $\forall(\phi^t,\lambda^t)$, on ait

$$\begin{cases} -\displaystyle\int_{\Sigma\times\Gamma} \frac{\partial G}{\partial \nu_x}\lambda(y)\phi^t(x)dydx - \frac{1}{ik}\oint_{\Sigma\times S} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y}\phi(y)\phi^t(x)dydx \\ \qquad = -\displaystyle\frac{1}{ik}\int_{\Sigma} \frac{\partial u_{inc}(x)}{\partial \nu}\phi^t(x)dx, \\ -\displaystyle\frac{1}{ik}\oint_{\Gamma\times S} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y}\phi(y)\phi^t(x)dydx - \frac{1}{2\eta}\int_{\Gamma} \phi(x)\phi^t(x)dx \\ \qquad - \displaystyle\int_{\Gamma\times\Gamma} \frac{\partial G}{\partial \nu_x}\lambda(y)\phi^t(x)dydx = -\frac{1}{ik}\int_{\Gamma} \frac{\partial u_{inc}(x)}{\partial \nu}\phi^t(x)dx, \\ \displaystyle\frac{\eta}{2}\int_{\Gamma} \lambda(x)\lambda^t(x)dx - ik\int_{\Gamma\times\Gamma} G\lambda(y)\lambda^t(x)dydx - \int_{\Gamma\times S} \frac{\partial G}{\partial \nu_y}\phi(y)\lambda^t(x)dydx \\ \qquad = -\displaystyle\int_{\Gamma} u_{inc}(x)\lambda^t(x)dx \end{cases} \tag{6.5}$$

Notons que l'opérateur D étant hypersingulier, le calcul des termes où apparaît $\partial^2 G/\partial\nu_x\partial\nu_y$ pose quelques difficultés. En pratique, on calculera ces intégrales sous la forme suivante :

$$\int_{S\times S} G(x,y)\left[-r\vec{ot}_S\phi(y).r\vec{ot}_S\phi^t(x) + k^2\nu_x.\nu_y\phi(x)\phi^t(y)\right]dydx$$

où $r\vec{ot}_S$ est le rotationnel surfacique défini par $r\vec{ot}_S\phi = \nu_x \wedge gr\vec{ad}_x\phi$. Pour simplifier, on conservera les termes en $\oint$ sous leur forme actuelle dans la suite du calcul.

6.1.3 Discrétisation

Pour notre discrétisation, on utilise une méthode d'éléments finis de Lagrange et un maillage triangulaire de la surface S. On note N_T le nombre de triangles et N_S le nombre de sommets du maillage.

Pour ϕ (qui représente le saut de pression), les fonctions de base seront notées ω_j avec $1 \leq j \leq N_S$. Ces fonctions sont affines, valant 1 sur le sommet S_j et 0 sur tous les autres (cf. figure 6.2b). L'approximation de ϕ s'écrit alors :

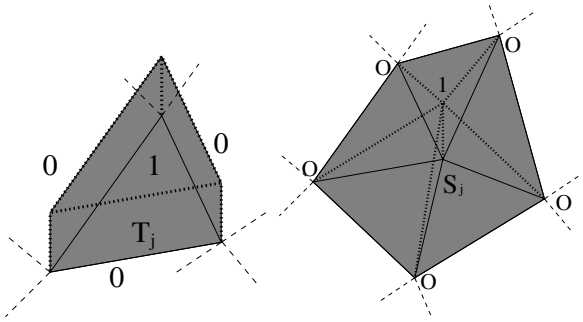
$$\phi(x) = \sum_{1\leq j\leq N_S} \phi_j\omega_j(x)$$

On note ϕ_j^Σ et ϕ_j^Γ les coefficients de ϕ associés aux sommets du maillage appartenant respectivement à Σ (partie non traitée de S) et Γ (partie traitée de S).

Pour λ (qui représente le saut de la dérivée normale de pression), les fonctions de base seront notées χ_j avec $1 \leq j \leq N_T$. Ces fonctions sont constantes par triangle, valant 1 sur le triangle T_j et 0 sur tous les autres (cf. figure 6.2a). L'approximation de λ s'écrit alors :

$$\lambda(x) = \sum_{1 \leq j \leq N_T} \lambda_j \chi_j(x)$$

λ étant nul sur Σ, on notera indifféremment λ_j ou λ_j^Γ les coefficients de λ sur Γ.



(a) Eléments P0 (b) Eléments P1

FIG. 6.2 – *Fonctions de base pour l'acoustique*

Dans le système (6.5), on réécrit la première équation avec $\phi^t = \omega_i^\Sigma$, pour la seconde on prend $\phi^t = \omega_i^\Gamma$, et pour la troisième $\lambda^t = \chi_i^\Gamma$. On obtient le système suivant :

$$\begin{cases} -\displaystyle\int_{\Sigma\times\Gamma} \frac{\partial G}{\partial \nu_x} \lambda(y)\omega_i^\Sigma(x) - \frac{1}{ik} \oint_{\Sigma\times S} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \phi(y)\omega_i^\Sigma(x) \\ \qquad\qquad = -\displaystyle\frac{1}{ik} \int_\Sigma \frac{\partial u_{inc}(x)}{\partial \nu} \omega_i^\Sigma(x), \\ -\displaystyle\frac{1}{ik} \oint_{\Gamma\times S} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \phi(y)\omega_i^\Gamma(x) - \frac{1}{2\eta} \int_\Gamma \phi(x)\omega_i^\Gamma(x) \\ \qquad\qquad - \displaystyle\int_{\Gamma\times\Gamma} \frac{\partial G}{\partial \nu_x} \lambda(y)\omega_i^\Gamma(x) = -\frac{1}{ik} \int_\Gamma \frac{\partial u_{inc}(x)}{\partial \nu} \omega_i^\Gamma(x), \\ \displaystyle\frac{\eta}{2} \int_\Gamma \lambda(x)\chi_i^\Gamma(x) - ik \int_{\Gamma\times\Gamma} G\lambda(y)\chi_i^\Gamma(x) - \int_{\Gamma\times S} \frac{\partial G}{\partial \nu_y} \phi(y)\chi_i^\Gamma(x) \\ \qquad\qquad = -\displaystyle\int_\Gamma u_{inc}(x)\chi_i^\Gamma(x) \end{cases}$$

Dans les intégrales doubles, il est sous-entendu que la variable x appartient au premier ensemble d'intégration, et y au second. En outre, le noyau de Green noté G signifie toujours $G(x,y)$, et les dx et dy sont partout implicites.

6.1.4 Résolution numérique classique

6.1.4.1 Ecriture matricielle

On recherche λ et ϕ sous la forme suivante :

$$\begin{cases} \phi(x) = \sum_{S_j \in \Gamma} \phi_j^\Gamma \omega_j^\Gamma(x) + \sum_{S_j \in \Sigma} \phi_j^\Sigma \omega_j^\Sigma(x), \\ \lambda(x) = \sum_{T_j \in \Gamma} \lambda_j^\Gamma \chi_j^\Gamma(x) \end{cases}$$

On aboutit à l'écriture matricielle suivante :

$$\begin{cases} -\sum_{T_j \in \Gamma} \lambda_j^\Gamma \int_{\Sigma\times\Gamma} \frac{\partial G}{\partial \nu_x} \chi_j^\Gamma(y) \omega_i^\Sigma(x) - \frac{1}{ik} \sum_{S_j \in \Gamma} \phi_j^\Gamma \oint_{\Sigma\times\Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \omega_j^\Gamma(y) \omega_i^\Sigma(x) \\ \qquad -\frac{1}{ik} \sum_{S_j \in \Sigma} \phi_j^\Sigma \oint_{\Sigma\times\Sigma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \omega_j^\Sigma(y) \omega_i^\Sigma(x) = -\frac{1}{ik} \int_\Sigma \frac{\partial u_{inc}(x)}{\partial \nu} \omega_i^\Sigma(x), \\ \sum_{S_j \in \Gamma} \phi_j^\Gamma \left\{ -\frac{1}{ik} \oint_{\Gamma\times\Gamma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \omega_j^\Gamma(y) \omega_i^\Gamma(x) - \frac{1}{2\eta} \int_\Gamma \omega_j^\Gamma(x) \omega_i^\Gamma(x) \right\} \\ \qquad -\frac{1}{ik} \sum_{S_j \in \Sigma} \phi_j^\Sigma \oint_{\Gamma\times\Sigma} \frac{\partial^2 G}{\partial \nu_x \partial \nu_y} \omega_j^\Sigma(y) \omega_i^\Gamma(x) \\ \qquad -\sum_{T_j \in \Gamma} \lambda_j^\Gamma \int_{\Gamma\times\Gamma} \frac{\partial G}{\partial \nu_x} \chi_j^\Gamma(y) \omega_i^\Gamma(x) = -\frac{1}{ik} \int_\Gamma \frac{\partial u_{inc}(x)}{\partial \nu} \omega_i^\Gamma(x), \\ \sum_{T_j \in \Gamma} \lambda_j^\Gamma \left\{ \frac{\eta}{2} \int_\Gamma \chi_j^\Gamma(x) \chi_i^\Gamma(x) - ik \int_{\Gamma\times\Gamma} G \chi_j^\Gamma(y) \chi_i^\Gamma(x) \right\} - \sum_{S_j \in \Gamma} \phi_j^\Gamma \int_{\Gamma\times\Gamma} \frac{\partial G}{\partial \nu_y} \omega_j^\Gamma(y) \chi_i^\Gamma(x) \\ \qquad -\sum_{S_j \in \Sigma} \phi_j^\Sigma \int_{\Gamma\times\Sigma} \frac{\partial G}{\partial \nu_y} \omega_j^\Sigma(y) \chi_i^\Gamma(x) = -\int_\Gamma u_{inc}(x) \chi_i^\Gamma(x) \end{cases}$$

Si l'on reprend les opérateurs S, K, K' et D tels qu'ils ont été définis par (6.2), on peut schématiser le système ci-dessus de la manière suivante :

$$\begin{bmatrix} -\frac{1}{ik}D & -\frac{1}{ik}D & -K \\ -\frac{1}{ik}D & -\frac{1}{ik}D - \frac{1}{2\eta}Id_\phi & -K \\ -K' & -K' & -ikS + \frac{\eta}{2}Id_\lambda \end{bmatrix} \times \begin{bmatrix} \phi_j^\Sigma \\ \phi_j^\Gamma \\ \lambda_j^\Gamma \end{bmatrix} = \begin{bmatrix} -\frac{1}{ik}\frac{\partial u_{inc}}{\partial \nu}|\Sigma \\ -\frac{1}{ik}\frac{\partial u_{inc}}{\partial \nu}|\Gamma \\ -u_{inc}|\Gamma \end{bmatrix} \tag{6.6}$$

La résolution de ce système conduit à la connaissance de ϕ et de $\lambda = -p/ik$, qui permettent à leur tour de connaître u^+ grâce au théorème de représentation (6.1).

6.1.4.2 Solveur direct ou itératif

On est donc tout naturellement amené à résoudre un système linéaire. Comme dans le cas électromagnétique, deux grandes classes de solveurs existent pour traiter ce problème : les solveurs directs et les solveurs itératifs. Toutes les remarques des sections 1.1.4.2 et 1.1.4.3 à ce sujet restent pleinement valables ici. Pour les mêmes raisons, on va donc s'orienter vers l'utilisation de solveurs itératifs, associés à une méthode rapide permettant d'accélérer les produits matrice-vecteur : la méthode multipôle rapide.

6.1.5 Résolution numérique multipôle

On va présenter dans cette section une écriture multipôle permettant de réaliser des produits par la matrice du problème (6.6).

6.1.5.1 Nécessité de la FMM

Avant toute chose, il convient de voir si la méthode multipôle rapide présente un intérêt pour ce type de problème. Pour ce faire, dans le tableau 6.1, on présente les ordres de grandeur caractéristiques des problèmes de grande taille que l'on souhaiterait traiter dans le domaine de l'électromagnétisme et de l'acoustique. Pour l'électromagnétisme, il s'agit (par exemple) de calculer l'interaction d'une onde radar de fréquence 3 GHz avec la structure d'un avion de diamètre 30 m. Dans le cas acoustique, on s'intéresse au bruit généré par les réacteurs d'un avion aux fréquences audibles.

Equation considérée	Electromagnétisme	Acoustique
Fréquence	de 300 MHz à 3 GHz	de 100 à 2000 Hz
Célérité des ondes	$3.10^8\ m.s^{-1}$	$340\ m.s^{-1}$
Longueur d'onde	de 1 m à 10 cm	de 3,4 m à 17 cm
Taille des objets (en m)	jusqu'à 30 m	jusqu'à 30 m
Taille en longueurs d'onde	jusqu'à 300	jusqu'à 180

TAB. 6.1 – *Grandeurs caractéristiques des problèmes traités*

Dans les deux cas, on obtient des problèmes dont les tailles en terme de longueurs d'onde sont de l'ordre de la centaine, soit bien au-delà de ce qu'il est possible de traiter avec un solveur usuel. De cette petite analyse, on peut déduire que la FMM présente réellement un intérêt pour la résolution de ce type d'équations.

6.1.5.2 Position du problème

On se donne deux fonctions ϕ et λ, définies respectivement sur S et Γ, et représentées par le vecteur de leurs coordonnées noté X :

$$X = \begin{bmatrix} \phi_j^\Sigma \\ \phi_j^\Gamma \\ \lambda_j^\Gamma \end{bmatrix}$$

On souhaite réaliser un produit matrice-vecteur par la matrice A définie par :

$$A = \begin{bmatrix} -\dfrac{1}{ik}D & -\frac{1}{ik}D & -K \\ -\dfrac{1}{ik}D & -\frac{1}{ik}D - \frac{1}{2\eta}Id_\phi & -K \\ -K' & -K' & -ikS + \frac{\eta}{2}Id_\lambda \end{bmatrix} \tag{6.7}$$

Autrement dit, pour chaque fonction de base ω_i^Σ, ω_i^Γ et χ_i^Γ, on désire calculer les trois quantités suivantes :

$$\begin{cases} -\dfrac{1}{ik}\displaystyle\oint_{\Sigma\times S} \dfrac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\omega_i^\Sigma(x) - \int_{\Sigma\times\Gamma}\dfrac{\partial G}{\partial\nu_x}\lambda(y)\omega_i^\Sigma(x), \\ -\dfrac{1}{ik}\displaystyle\oint_{\Gamma\times S} \dfrac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\omega_i^\Gamma(x) - \dfrac{1}{2\eta}\int_\Gamma \phi(x)\omega_i^\Gamma(x) - \int_{\Gamma\times\Gamma}\dfrac{\partial G}{\partial\nu_x}\lambda(y)\omega_i^\Gamma(x), \\ -\displaystyle\int_{\Gamma\times S}\dfrac{\partial G}{\partial\nu_y}\phi(y)\chi_i^\Gamma(x) - ik\int_{\Gamma\times\Gamma} G\lambda(y)\chi_i^\Gamma(x) + \dfrac{\eta}{2}\int_\Gamma \lambda(x)\chi_i^\Gamma(x) \end{cases} \tag{6.8}$$

6.1.5.3 Suppression des interactions proches

On le sait, la méthode multipôle s'appuie sur un octree, et ne traite que les interactions entre degrés de liberté ne se trouvant pas dans des feuilles voisines de cet arbre. Les autres interactions sont traitées classiquement par le biais d'une matrice creuse. Or dans la matrice A mentionnée ci-dessus (6.7) apparaissent deux matrices d'interactions proches Id_ϕ et Id_λ dont les coefficients valent respectivement :

$$\begin{cases} (Id_\phi)_{i,j} = \displaystyle\int_S \omega_j(x)\omega_i(x), \\ (Id_\lambda)_{i,j} = \displaystyle\int_\Gamma \chi_j^\Gamma(x)\chi_i^\Gamma(x) \end{cases}$$

Dans la matrice Id_λ, compte tenu de la définition des fonctions χ_j (cf. figure 6.2a), seuls les termes diagonaux sont non-nuls. Cette matrice ne sera donc pas traitée par la FMM. Dans la matrice Id_ϕ, compte tenu de la définition des fonctions ω_j (cf. figure 6.2b), les seuls termes $(Id_\phi)_{i,j}$ non-nuls sont ceux correspondant à des sommets S_i et S_j reliés par une arête ou confondus. Ce sont en effet les deux seuls cas où les supports des fonctions de base ω_i et ω_j sont non-disjoints. A priori, deux tels sommets vont se retrouver dans la même feuille, ou dans deux feuilles voisines de l'octree. Néanmoins, on peut tout à fait rencontrer des cas parasites où deux sommets reliés par une arête sont dans des feuilles distantes de l'arbre. Il suffit pour cela d'avoir une arête $[S_iS_j]$ de dimension supérieure à la taille des feuilles (cf. figure 6.3).

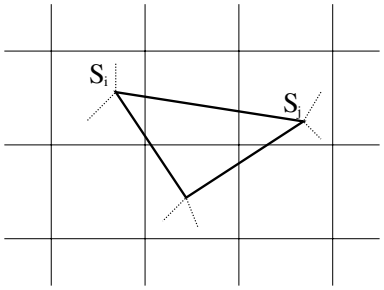


FIG. 6.3 – *Cas parasite : terme lointain dans Id_ϕ*

Avec une taille de feuille de l'ordre d'un quart de longueur d'onde, contre un dixième de longueur d'onde pour les arêtes du maillage, ce genre de situation devrait toutefois se faire rare. Par souci de simplification, nous allons mettre de côté ces cas exceptionnels, et considérer que tous les termes de la matrice Id_ϕ sont « proches » au sens de la FMM. Il convient bien sûr de vérifier que cette hypothèse est juste, et de traiter explicitement les éventuels cas parasites rencontrés.

Moyennant ces remarques, on peut supprimer Id_ϕ et Id_λ de l'expression (6.7), et ne plus distinguer les fonctions de base ϕ_j^Σ et ϕ_j^Γ. Dès lors, pour les traitements des interactions lointaines, on peut considérer que la matrice A que la FMM va tenter d'approcher s'écrit, avec le vecteur X associé :

$$A = \begin{bmatrix} -\frac{1}{ik}D & -K \\ -K' & -ikS \end{bmatrix} \qquad X = \begin{bmatrix} \phi_j \\ \lambda_j \end{bmatrix} \tag{6.9}$$

Bien sûr, pour le traitement des interactions proches, la matrice 6.9 est fausse, et c'est à la formule 6.7 qu'il faut se reporter.

Et ainsi, les termes d'interactions lointaines à calculer deviennent :

$$\left\{\begin{array}{l} -\dfrac{1}{ik}\displaystyle\oint_{S\times S}\dfrac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\omega_i(x)-\displaystyle\int_{S\times\Gamma}\dfrac{\partial G}{\partial\nu_x}\lambda(y)\omega_i(x), \\ -\displaystyle\int_{\Gamma\times S}\dfrac{\partial G}{\partial\nu_y}\phi(y)\chi_i^\Gamma(x)-ik\displaystyle\int_{\Gamma\times\Gamma}G\lambda(y)\chi_i^\Gamma(x) \end{array}\right. \tag{6.10}$$

6.1.5.4 Préparation à un calcul multipôle

Nous allons dans cette section revenir sur les étapes préparatoires de la FMM. Tout d'abord, on doit créer un découpage hiérarchique de la surface S. Pour cela, on construit un octree autour de l'objet Ω^-, et on répartit les degrés de liberté entre les feuilles de cet arbre (cf. section 1.2.2.3) : les fonctions de base χ_j sont localisées au centre de gravité du triangle T_j, et les fonctions de base ω_j sont localisées en leur sommet S_j (cf. figure 6.2). A chaque niveau de l'arbre, on détermine le nombre de pôles utilisés pour calculer les fonctions de transfert, et le nombre de directions (θ,φ) utilisées pour stocker les fonctions de transfert.

Les interactions proches, entre degrés de liberté situés dans des feuilles voisines, sont traitées classiquement. Il reste donc à calculer les interactions lointaines. Pour ce faire, on se donne deux feuilles $\mathcal{C}$ et $\mathcal{C}'$ non-voisines, de centre M et M', et on cherche à calculer le terme d'interaction entre deux sous-domaines $S\cap\mathcal{C}$ et $S\cap\mathcal{C}'$ (avec les précisions de la section 1.2.2.3.1 concernant les intégrales sur $S\cap\mathcal{C}$).

6.1.5.5 Cas d'un objet rigide

Pour simplifier, nous allons d'abord étudier le cas d'un objet entièrement rigide, sur lequel aucune surface n'est traitée. Dans ce cas, Γ est vide, et $S=\Sigma$. Le vecteur X et la matrice A se réduisent à :

$$X=\left[\ \phi_j\ \right]\qquad A=\left[\ -\frac{1}{ik}D\ \right]$$

Dans (6.10), le seul terme à calculer s'écrit alors :

$$-\frac{1}{ik}\oint_{S\times S}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\omega_i(x)$$

Par conséquent, le terme d'interaction entre les feuilles $\mathcal{C}$ et $\mathcal{C}'$ vaut :

$$-\frac{1}{ik}\int_{x\in S\cap\mathcal{C}'}\oint_{y\in S\cap\mathcal{C}}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi(y)\omega_i(x)dydx$$

On va utiliser la décomposition du noyau (6.11), avec une petite différence par rapport aux chapitres précédents : ici, y est le point « de départ » et x est le point « d'arrivée » du transfert.

$$G(x,y)=\frac{ik}{16\pi^2}\lim_{L\to+\infty}\int_{\vec{s}\in\mathcal{S}}e^{ik\vec{s}.y\vec{M}}T^L_{M\vec{M}'}(\vec{s})e^{ik\vec{s}.\vec{M'}x}d\vec{s} \tag{6.11}$$

La variable d'intégration $\vec{s}$ parcourt la sphère unité $\mathcal{S}$ (à ne pas confondre avec la surface de l'objet S qui est parcourue par les variables x et y). Il faut garder à l'esprit qu'en pratique,

le nombre de pôles L est fixé à l'avance, et que l'intégrale sur la sphère unité est en fait discrétisée.

Les dépendances en x et y étant séparées, le calcul des dérivées de G est alors simplifié. On a en effet :

$$\left\{\begin{array}{l} \dfrac{\partial(e^{ik\vec{s}.\vec{M'}x})}{\partial\vec{\nu}_x} = \vec{grad}_x(e^{ik\vec{s}.\vec{M'}x}).\vec{\nu}_x = ik\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x}, \\ \dfrac{\partial(e^{ik\vec{s}.y\vec{M}})}{\partial\vec{\nu}_y} = \vec{grad}_y(e^{ik\vec{s}.y\vec{M}}).\vec{\nu}_y = -ik\vec{s}.\vec{\nu}_y e^{ik\vec{s}.y\vec{M}} \end{array}\right. \tag{6.12}$$

On en déduit la décomposition suivante pour la dérivée double du noyau :

$$\frac{\partial^2 G}{\partial\nu_x\partial\nu_y} = \frac{ik}{16\pi^2}\lim_{L\to+\infty}\int_{\vec{s}\in\mathcal{S}} k\vec{s}.\vec{\nu}_y e^{ik\vec{s}.y\vec{M}} T^L_{M\vec{M'}}(\vec{s}) k\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x} d\vec{s}$$

On a maintenant tous les éléments pour écrire notre algorithme multipôle acoustique dans le cas d'un objet rigide.

Initialisation : on calcule la fonction $\mathcal{F}_\mathcal{C}$ définie sur la sphère unité $\mathcal{S}$ par :

$$\mathcal{F}_\mathcal{C}(\vec{s}) = -\int_{y\in S\cap\mathcal{C}} k\vec{s}.\vec{\nu}_y e^{ik\vec{s}.y\vec{M}}\phi(y)dy \tag{6.13}$$

Montée, Transfert, Descente : toutes ces phases de calcul sont inchangées par rapport au cas électromagnétique.

Intégration : on termine le calcul en intégrant le résultat du transfert à la fois sur la sphère unité $\mathcal{S}$ et sur $S\cap\mathcal{C}'$:

$$\frac{1}{16\pi^2}\int_{x\in S\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_\mathcal{C}(\vec{s}) k\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x}\omega_i(x) d\vec{s}dx \tag{6.14}$$

En ajoutant cette quantité au terme d'interaction proche, on obtient le résultat du produit matrice-vecteur recherché.

6.1.5.6 Cas général

Dans le cas général, il faut traiter les 4 termes apparaissant dans l'équation (6.10). Pour chacune de ces quatre intégrales doubles, il faut préciser l'expression des étapes d'initialisation et d'intégration correspondantes. En utilisant (6.11) et (6.12), on obtient les décompositions suivantes pour les dérivées premières de $G(x,y)$:

$$\left\{\begin{array}{l} \dfrac{\partial G(x,y)}{\partial\nu_x} = -\dfrac{k}{16\pi^2}\lim\limits_{L\to+\infty}\displaystyle\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.y\vec{M}} T^L_{M\vec{M'}}(\vec{s}) k\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x} d\vec{s}, \\ \dfrac{\partial G(x,y)}{\partial\nu_y} = \dfrac{k}{16\pi^2}\lim\limits_{L\to+\infty}\displaystyle\int_{\vec{s}\in\mathcal{S}} k\vec{s}.\vec{\nu}_y e^{ik\vec{s}.y\vec{M}} T^L_{M\vec{M'}}(\vec{s}) e^{ik\vec{s}.\vec{M'}x} d\vec{s} \end{array}\right.$$

On en déduit facilement l'écriture FMM correspondant aux autres termes à calculer.

Pour le terme $-\int_{S\times\Gamma}\frac{\partial G}{\partial\nu_x}\lambda(y)\omega_i(x)$, l'initialisation et l'intégration s'écrivent :

$$\left\{\begin{array}{l} \mathcal{F}_\mathcal{C}(\vec{s}) = \displaystyle\int_{y\in\Gamma\cap\mathcal{C}} k e^{ik\vec{s}.y\vec{M}}\lambda(y)dy, \\ \text{Calculer : } \dfrac{1}{16\pi^2}\displaystyle\int_{x\in S\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \mathcal{G}_\mathcal{C}(\vec{s}) k\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x}\omega_i(x) d\vec{s}dx \end{array}\right. \tag{6.15}$$

Le terme $-\int_{\Gamma\times S}\frac{\partial G}{\partial \nu_y}\phi(y)\chi_i^\Gamma(x)$ est obtenu avec les phases d'initialisation et d'intégration qui suivent :

$$\left\{\begin{array}{l}\mathcal{F}_{\mathcal{C}}(\vec{s}) = -\displaystyle\int_{y\in S\cap\mathcal{C}} k\vec{s}.\vec{\nu}_y e^{ik\vec{s}.y\vec{M}}\phi(y)dy,\\ \text{Calculer}: \dfrac{1}{16\pi^2}\displaystyle\int_{x\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\mathcal{G}_{\mathcal{C}}(\vec{s})ke^{ik\vec{s}.\vec{M'}x}\chi_i^\Gamma(x)d\vec{s}dx\end{array}\right. \tag{6.16}$$

Enfin, le dernier terme $-ik\int_{\Gamma\times\Gamma}G\lambda(y)\chi_i^\Gamma(x)$ se calcule de la manière suivante :

$$\left\{\begin{array}{l}\mathcal{F}_{\mathcal{C}}(\vec{s}) = \displaystyle\int_{y\in\Gamma\cap\mathcal{C}} ke^{ik\vec{s}.y\vec{M}}\lambda(y)dy,\\ \text{Calculer}: \dfrac{1}{16\pi^2}\displaystyle\int_{x\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\mathcal{G}_{\mathcal{C}}(\vec{s})ke^{ik\vec{s}.\vec{M'}x}\chi_i^\Gamma(x)d\vec{s}dx\end{array}\right. \tag{6.17}$$

On obtient donc une formulation multipôle complète à quatre composantes pour calculer les 4 termes apparaissant dans (6.10). Cependant, on peut simplifier tout cela. En effet, on remarque que les étapes d'initialisation sont identiques dans (6.13) et (6.16) d'une part, dans (6.15) et (6.17) d'autre part. En outre, les étapes d'intégration sont identiques dans (6.14) et (6.15) d'une part, dans (6.16) et (6.17) d'autre part. On peut tirer partie de ces similitudes pour proposer une formulation multipôle à une seule composante.

Initialisation : on calcule la fonction $\mathcal{F}_{\mathcal{C}}$ définie sur la sphère unité $\mathcal{S}$ par :

$$\mathcal{F}_{\mathcal{C}}(\vec{s}) = \int_{y\in S\cap\mathcal{C}} k\left[\lambda(y) - \vec{s}.\vec{\nu}_y\phi(y)\right]e^{ik\vec{s}.y\vec{M}}dy \tag{6.18}$$

Montée, Transfert, Descente : toutes ces phases de calcul sont inchangées par rapport au cas électromagnétique. Elles se font sur des fonctions de radiation à une seule composante.

Intégration : on termine le calcul en intégrant le résultat du transfert à la fois sur la sphère unité $\mathcal{S}$ et sur $S\cap\mathcal{C}'$. On a deux formules d'intégration, selon que l'on intègre sur une fonction ω_i (première ligne de (6.10)) ou χ_i (seconde ligne de (6.10)) :

$$\left\{\begin{array}{l}\dfrac{1}{16\pi^2}\displaystyle\int_{x\in S\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\mathcal{G}_{\mathcal{C}}(\vec{s})k\vec{s}.\vec{\nu}_x e^{ik\vec{s}.\vec{M'}x}\omega_i(x)d\vec{s}dx,\\ \dfrac{1}{16\pi^2}\displaystyle\int_{x\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}}\mathcal{G}_{\mathcal{C}}(\vec{s})ke^{ik\vec{s}.\vec{M'}x}\chi_i^\Gamma(x)d\vec{s}dx\end{array}\right.$$

En ajoutant ces quantités aux termes d'interaction proche correspondants, on obtient le résultat du produit matrice-vecteur recherché.

On peut rapprocher la formule d'initialisation (6.18) de l'expression du théorème de représentation (6.1) que l'on rappelle ici :

$$u(x) = \int_S\left(G(x,y)p(y) - \frac{\partial G(x,y)}{\partial\nu_y}\phi(y)\right)dy \tag{6.19}$$

La première représente le champ lointain diffracté par la portion de l'objet $S\cap\mathcal{C}$ dans la direction $\vec{s}\in\mathcal{S}$. La seconde permet de calculer u en tout point de l'espace à partir de la connaissance de p (qui vaut $-ik\lambda$) et de ϕ sur l'objet. A une constante près, (6.18) est la traduction « multipôle » de (6.19). La ressemblance entre les deux n'est donc par fortuite, et le fait de parvenir à écrire une FMM à une seule composante est donc logique et n'a rien d'un miracle.

6.1.5.7 Synthèse

En pratique, cette formulation multipôle est très proche de l'écriture présentée dans le cas des équations de Maxwell. Au chapitre des nouveautés, on peut souligner les points suivants :

- La coexistence de plusieurs types de degrés de liberté. Les fonctions ϕ et λ, et les fonctions de base associées ω_i et χ_i, sont de natures très différentes (voir leurs définitions figure 6.2). On est donc amené à faire coexister dans notre octree des inconnues disparates, ce qui n'était pas le cas au chapitre 1.
- Une nouvelle définition des étapes d'initialisation et d'intégration. Toute formulation intégrale s'appuyant sur le noyau de Green $G(x,y) = e^{ik|x-y|}/4\pi|x-y|$ peut être adaptée à notre FMM, pour peu qu'on réécrive correctement ces deux phases du calcul. Ici, on a proposé une formulation efficace à une seule composante, pouvant traiter aussi bien les surfaces rigides que les surfaces traitées. Le coefficient d'impédance η n'apparaît d'ailleurs pas dans la FMM, puisqu'il se cantonne aux interactions proches (cf. équation 6.8). Ceci n'est pas dû à la physique du problème mais au choix du problème intérieur.

6.1.6 Application

Dans cette section, nous allons présenter quelques résultats illustrant cette formulation multipôle acoustique. Tout au long du chapitre 3, nous avons examiné la vitesse et la précision de la notre FMM pour les équations de Maxwell. Compte tenu du peu de différence entre cette dernière et notre nouvelle FMM acoustique, il est clair que nous allons retrouver des performances proches. Nous nous bornerons donc à une courte étude en deux parties : d'une part, une comparaison entre notre nouvelle FMM et les méthodes existantes, et d'autre part quelques tests de vitesse et de précision sur des cas de grande taille.

6.1.6.1 Comparaison avec les méthodes usuelles

On étudie le cas de sphères rigides illuminées par une onde plane. Les deux maillages utilisés comportent 1446 et 5778 inconnues, sont de diamètre 2 et 4 longueurs d'onde, et sont maillés avec 10 points par longueur d'onde. Regardons tout d'abord les temps d'exécution sur un PC à 866 MHz pour chacun des trois solveurs possibles. On note n_{dl} le nombre d'inconnues du problème, N_{iter} le nombre d'itérations du solveur itératif, et N_{rhs} le nombre de seconds membres ($N_{rhs} = 1$ pour l'exemple considéré).

Les résultats du tableau 6.2 sont bien sûr à mettre en parallèle avec leur équivalent électromagnétique du tableau 3.1. Comme prévu, la résolution par méthode multipôle est la plus rapide, et c'est la seule dont la croissance théorique n'augure pas de temps de calcul astronomique pour n_{dl} grand. Les solveurs itératifs ont utilisé $N_{iter} = 27$ itérations pour converger avec la petite sphère, $N_{iter} = 86$ itérations avec le grande sphère, avec une tolérance $\varepsilon = 10^{-3}$ dans les deux cas. Les écarts effectifs entre le résultat du solveur direct (considéré ici comme référence) et les deux autres sont donnés dans le tableau 6.3 :

Ces précisions sont excellentes, meilleures encore que dans le cas électromagnétique. Néanmoins, pour en tirer des conclusions il faudrait faire des tests plus nombreux (ce que nous n'avons pas encore fait, faute de temps et de maillages adaptés).

	Sphère1446	Sphère5778	Croissance
Solveur direct :			
initialisation	171 s	4752 s	${n_{dl}}^3$
résolution	1 s	6 s	$N_{rhs}{n_{dl}}^2$
Solveur itératif :			
initialisation	100 s	1720 s	${n_{dl}}^2$
résolution	10 s	523 s	$N_{rhs}.N_{iter}.{n_{dl}}^2$
Solveur multipôle :			
initialisation	12 s	58 s	n_{dl}
résolution	58 s	828 s	$N_{rhs}.N_{iter}.n_{dl}\log n_{dl}$

TAB. 6.2 – *Temps d'exécution pour chaque type de solveur sur PC 866MHz*

	Sphère1446	Sphère5778
Solveur itératif	0,12 %	0,26 %
Solveur multipôle	0,48 %	0,73 %

TAB. 6.3 – *Précision pour différents types de solveurs itératifs*

6.1.6.2 Tests sur des problèmes de grande taille

On va utiliser deux maillages dans cette section. Le premier est le maillage d'une sphère rigide, de diamètre 25 longueurs d'onde, comportant 236198 inconnues. Le second est une nacelle de moteur d'avion, partiellement traitée, comportant 99229 inconnues, de diamètre 12,5 longueurs d'onde. Environ 5 % des 180000 triangles sont traités et portent deux types de degrés de liberté.

	Sphère236198	Nacelle99229
Temps de démarrage	4080 s	1762 s
Temps par itération	361 s	223 s
Mémoire vive	278 Mo	54 Mo
Espace disque	3,3 Go	2,1 Go
Précision de la FMM	$7{,}3.10^{-4}$	$1{,}8.10^{-3}$

TAB. 6.4 – *Calculs acoustiques de grande taille sur PC 866MHz*

On observe de nouveau une très bonne précision, et des temps de calcul très satisfaisants, compte tenu de la taille des problèmes traités, et de la machine utilisée (un simple PC Pentium III à 866 MHz).

6.2 Cas d'un guide d'ondes

On va maintenant étudier un modèle physique plus élaboré développé par notre partenaire industriel (EADS).

6.2.1 Modèle physique considéré

On s'intéresse ici au bruit à l'avant d'un moteur d'avion de ligne. La principale cause de ce bruit est la soufflante (c'est-à-dire l'ensemble hélices, turbines, compresseur,...) que nous allons modéliser par un guide d'ondes. Ce guide se trouvera à l'intérieur d'une nacelle que l'on supposera pleine et rigide. La figure 6.4 schématise le modèle physique considéré.

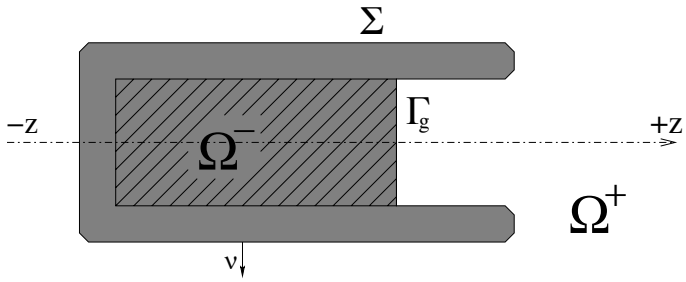


FIG. 6.4 – *Problème acoustique : cas d'un guide d'ondes*

On note Ω^- l'ensemble nacelle+guide et Ω^+ le domaine infini extérieur. Sur la figure 6.4, le guide d'ondes est représenté en grisé hachuré, la nacelle en grisé, et le domaine extérieur en blanc. On introduit une frontière fictive Γ_g marquant la séparation entre le guide d'ondes et Ω^+. On note Σ le reste de la frontière de Ω^-. On note S la frontière de Ω^- : $S = \partial\Omega^- = \Sigma \cup \Gamma_g$. Sur cette frontière, on note ν la normale extérieure à Ω^-. La surface Σ sera considérée comme rigide.

6.2.2 Modélisation mathématique

6.2.2.1 Guide d'ondes

Tout d'abord, on va s'intéresser au guide d'ondes seul. Pour simplifier, on va se placer dans le cas d'un guide d'ondes noté $\mathcal{G}$, infini dans la direction de l'axe (Oz), axisymétrique de section circulaire Γ_g et de rayon r_g. A l'intérieur de ce cylindre, on cherche à résoudre l'équation de Helmholtz avec conditions aux limites de types Neumann (rigide) sur la frontière :

$$\left\{ \begin{array}{ll} \Delta u + k^2 u = 0 & \textit{dans } \mathcal{G}, \\ \dfrac{\partial u}{\partial \nu} = 0 & \textit{sur } \partial\mathcal{G} \end{array} \right. \tag{6.20}$$

On cherche les solutions de ce problème en coordonnées cylindriques sous la forme $u(x,y,z) = V(r,\phi)e^{i\gamma z}$ où γ est un nombre d'onde (éventuellement complexe) caractérisant la propagation de u dans le guide, et qui reste à déterminer. V est alors solution du problème aux valeurs propres :

$$\left\{ \begin{array}{l} \Delta_{2D} V + (k^2 - \gamma^2) V = 0, \\ \left(\dfrac{\partial V}{\partial r} \right)_{|r=r_g} = 0 \end{array} \right.$$

On note $k_c^2 = k^2 - \gamma^2$. L'opérateur Δ_{2D} étant autoadjoint, ses vecteurs propres (aussi appelés *modes*) forment une base Hilbertienne. On cherchera donc les solutions de (6.20) sous la forme de combinaisons linéaires de ces modes. La méthode de séparation des variables nous conduit

à rechercher les vecteurs propres en coordonnées polaires (r,ϕ) sous la forme $W_m(r)e^{im\phi}$. On trouve alors que W_m doit vérifier :

$$\begin{cases} W_m''(r) + \dfrac{1}{r}W_m'(r) + \left(k_c^2 - \dfrac{m^2}{r^2}\right) W_m(r) = 0, \\ W_m'(r_g) = 0 \end{cases}$$

La solution de la première équation s'écrit $W_m(r) = J_m(k_c r)$ où J_m est la fonction de Bessel d'ordre m solution de $f'' + 1/rf' + (1 - m^2/r^2)f = 0$ (appelée équation de Bessel d'ordre m). La condition au bord impose que $k_c r_g$ soit un zéro de J_m'. On note (z_{mn}) la suite infinie dénombrable des zéros positifs de J_m', et on note $k_{cmn} = z_{mn}/r_g$.

Les solutions de l'équation (6.20) se décomposent sur les modes :

$$J_m(k_{cmn}r)e^{im\phi}e^{i\gamma_{mn}z} \text{ avec } \gamma_{mn}^2 = k^2 - k_{cmn}^2 \tag{6.21}$$

On distingue deux types de modes :

- Les modes propagatifs : ce sont ceux pour lesquels $k > k_{cmn}$. γ_{mn} est alors réel, et vaut $\sqrt{k^2 - k_{cmn}^2}$ (au signe près). Ces modes se propagent dans le guide d'ondes avec une amplitude constante. Pour une valeur de k donnée, c'est-à-dire pour une fréquence donnée, le nombre de zéros de J_m' vérifiant $z_{mn}/r_g < k$ est fini, il y a donc un nombre fini de modes propagatifs.
- Les modes évanescents : ce sont les autres, pour lesquels $k < k_{cmn}$. γ_{mn} est alors imaginaire pur, et vaut $i\sqrt{k^2 - k_{cmn}^2}$ (au signe près). Ces modes se propagent dans le guide d'ondes avec une amplitude décroissante. Ils existent en quantité infinie dénombrable.

Pour simplifier les notations, on va utiliser le caractère dénombrable de ces modes pour n'utiliser qu'un seul indice : on notera donc ces modes $(u_m)_{m\geq 0}$, associé au nombre d'onde γ_m, les (u_m) étant rangés dans l'ordre des γ_m^2 décroissants (i.e. d'abord les propagatifs, qui sont en nombre fini, puis les évanescents). Enfin, on distinguera les modes incidents (qui se propagent vers les z négatifs, i.e. $\gamma_m < 0$) et les modes réfléchis ou diffractés (qui se propagent vers les z positifs, i.e. $\gamma_m > 0$). Ces modes seront donc notés u_m^{inc} ou u_m^{diff}.

Nous allons utiliser ce guide d'ondes de la manière suivante : en fonction de la dimension de ce guide, et de la précision recherchée dans notre calcul, on fixe le nombre de modes incidents et diffractés que l'on souhaite conserver, soit M^{inc} et M^{diff} ces nombres (qui en pratique valent au plus quelques centaines). On décompose alors le champ u sur la frontière du guide d'ondes Γ_g en deux termes :

- D'une part le champ incident u^{inc}, qui se décompose sur les modes incidents $u^{inc} = \sum_{0\leq m<M^{inc}} \alpha_m u_m^{inc}$. C'est la donnée du problème.
- D'autre part le champ diffracté $u^{diff} = \sum_{0\leq m<M^{diff}} \beta_m u_m^{diff}$. C'est une inconnue du problème.

Le champ total s'écrit bien sûr $u = u^{inc} + u^{diff}$. Les coefficients $(\beta_m)_{0\leq m<M^{diff}}$ sont les coefficients de réflexion du champ incident u^{inc} sur les modes diffractés u_m^{diff}.

On a ici présenté le cas d'un guide d'ondes de section circulaire, pour lequel les modes peuvent être calculés de manière analytique. Dans les cas plus complexes, la solution analytique peut être remplacée par un calcul numérique de résolution d'un problème aux valeurs propres sur une section du guide d'ondes. Cela ne change rien à notre propos, ni à ce qui va suivre.

6.2.2.2 Représentation intégrale

On revient au problème entier représenté sur la figure 6.4. Comme on l'a fait dans le cas d'une onde plane incidente, on définit un problème (P^+) dans le domaine Ω^+ de la manière suivante :

$$(P^+)\left\{\begin{array}{ll} \Delta u^+ + k^2 u^+ = 0 & dans\ \Omega^+, \\ \dfrac{\partial u^+}{\partial \nu} = 0 & sur\ \Sigma, \\ u^+ = \displaystyle\sum_{0\le m<M^{inc}} \alpha_m u_m^{inc} + \sum_{0\le m<M^{diff}} \beta_m u_m^{diff} & sur\ \Gamma_g, \\ \dfrac{\partial u^+}{\partial \nu} = \displaystyle\sum_{0\le m<M^{inc}} \alpha_m \frac{\partial u_m^{inc}}{\partial \nu} + \sum_{0\le m<M^{diff}} \beta_m \frac{\partial u_m^{diff}}{\partial \nu} & sur\ \Gamma_g, \\ \displaystyle\lim_{r\to+\infty} r\left(\frac{\partial u^+}{\partial r} - iku^+\right) = 0 & \end{array}\right. \tag{6.22}$$

On définit dans le domaine Ω^- un problème arbitraire (P^-) :

$$(P^-)\left\{\begin{array}{ll} \Delta u^- + k^2 u^- = 0 & dans\ \Omega^-, \\ u^- = 0 & sur\ S \end{array}\right.$$

Ω^- étant borné, la solution de (P^-) est uniformément nulle : $u^- = 0$. L'application du théorème de représentation (6.3) conduit alors à écrire :

$$\left\{\begin{array}{l} -ikS\lambda + (\dfrac{I}{2} - K')\phi = 0, \\ -\dfrac{1}{ik}D\phi - (\dfrac{I}{2} + K)\lambda = 0 \end{array}\right.$$

avec

$$\left\{\begin{array}{ll} \phi(x) = -u^+_{|S}(x) & x \in S, \\ \lambda(x) = \dfrac{1}{ik}\left(\dfrac{\partial u^+}{\partial \nu}\right)_{|S}(x) & x \in S \end{array}\right.$$

On introduit les fonctions test ϕ^t et λ^t et on intègre sur S (rappelons que $S = \Gamma_g \cup \Sigma$). λ et λ^t étant nuls sur Σ, on obtient :

$$\left\{\begin{array}{l} -ik\displaystyle\int_{\Gamma_g\times\Gamma_g} G\lambda(y)\lambda^t(x)dydx + \frac{1}{2}\int_{\Gamma_g}\phi(x)\lambda^t(x)dx - \int_{\Gamma_g\times S}\frac{\partial G}{\partial \nu_y}\phi(y)\lambda^t(x)dydx = 0, \\ -\dfrac{1}{2}\displaystyle\int_{\Gamma_g}\lambda(x)\phi^t(x)dx - \int_{S\times\Gamma_g}\frac{\partial G}{\partial \nu_x}\lambda(y)\phi^t(x)dydx - \frac{1}{ik}\oint_{S\times S}\frac{\partial^2 G}{\partial \nu_x \partial \nu_y}\phi(y)\phi^t(x)dydx = 0 \end{array}\right. \tag{6.23}$$

6.2.3 Discrétisation

On conserve bien sûr un maillage à base de triangles, tant sur Σ que sur la frontière Γ_g du guide d'ondes. On utilise toujours les fonctions de base ω_i (sur S) et χ_i (sur Γ_g uniquement).

De manière classique, sur Σ on prend pour fonction test ϕ^t une fonction de base ω_i. Sur l'interface du guide d'ondes Γ_g, on va prendre pour fonction test les modes diffractés :

$$\begin{cases} \phi^t = u_m^{diff}, \\ \lambda^t = -\dfrac{1}{ik}\dfrac{\partial u_m^{diff}}{\partial \nu} \end{cases}$$

Notons qu'il y a une différence de nature entre les deux types de fonctions tests : les fonctions de base ω_i sont locales (leurs supports sont tous différents, et se réduisent à quelques triangles), alors que les fonctions modales u_m^{diff} sont globales (leur support est commun, c'est l'interface du guide d'ondes Γ_g toute entière). On va noter $v_m^{diff} = -\frac{1}{ik}\frac{\partial u_m^{diff}}{\partial \nu}$. Sur Γ_g, on décompose ϕ et λ en partie incidente (connue) et diffractée (inconnue) :

$$\begin{cases} \phi = \phi^{inc} + \phi^{diff}, \\ \lambda = \lambda^{inc} + \lambda^{diff} \\ \text{avec :} \\ \phi^{inc} = \displaystyle\sum_{0 \le m < M^{inc}} \alpha_m u_m^{inc}, \\ \phi^{diff} = \displaystyle\sum_{0 \le m < M^{diff}} \beta_m u_m^{diff}, \\ \lambda^{inc} = \displaystyle\sum_{0 \le m < M^{inc}} \alpha_m v_m^{inc}, \\ \lambda^{diff} = \displaystyle\sum_{0 \le m < M^{diff}} \beta_m v_m^{diff} \end{cases}$$

Sur Σ, par analogie on décompose $\phi = \phi^{inc} + \phi^{diff}$, étant entendu que $\phi^{inc} = 0$ et $\phi^{diff} = \sum \phi_j \omega_j$.

On a alors toutes les briques pour réécrire le système (6.23). On va passer les données incidentes dans le terme de droite, et utiliser les fonctions tests (ω_i^{Σ},u_m^{diff},v_m^{diff}). On obtient :

$$\left\{\begin{array}{l}
-ik\displaystyle\int_{\Gamma_g\times\Gamma_g} G\lambda^{diff}(y)v_m^{diff}(x)dydx + \frac{1}{2}\int_{\Gamma_g}\phi^{diff}(x)v_m^{diff}(x)dx \\
-\displaystyle\int_{\Gamma_g\times S}\frac{\partial G}{\partial\nu_y}\phi^{diff}(y)v_m^{diff}(x)dydx = ik\int_{\Gamma_g\times\Gamma_g} G\lambda^{inc}(y)v_m^{diff}(x)dydx \\
-\displaystyle\frac{1}{2}\int_{\Gamma_g}\phi^{inc}(x)v_m^{diff}(x)dx + \int_{\Gamma_g\times\Gamma_g}\frac{\partial G}{\partial\nu_y}\phi^{inc}(y)v_m^{diff}(x)dydx \qquad \forall v_m^{diff}, \\
\\
-\displaystyle\frac{1}{2}\int_{\Gamma_g}\lambda^{diff}(x)u_m^{diff}(x)dx - \int_{\Gamma_g\times\Gamma_g}\frac{\partial G}{\partial\nu_x}\lambda^{diff}(y)u_m^{diff}(x)dydx \\
-\displaystyle\frac{1}{ik}\oint_{\Gamma_g\times S}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi^{diff}(y)u_m^{diff}(x)dydx = \frac{1}{2}\int_{\Gamma_g}\lambda^{inc}(x)u_m^{diff}(x)dx \\
+\displaystyle\int_{\Gamma_g\times\Gamma_g}\frac{\partial G}{\partial\nu_x}\lambda^{inc}(y)u_m^{diff}(x)dydx + \frac{1}{ik}\oint_{\Gamma_g\times\Gamma_g}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi^{inc}(y)u_m^{diff}(x)dydx \qquad \forall u_m^{diff}, \\
\\
-\displaystyle\int_{\Sigma\times\Gamma_g}\frac{\partial G}{\partial\nu_x}\lambda^{diff}(y)\omega_i(x)dydx - \frac{1}{ik}\oint_{\Sigma\times S}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi^{diff}(y)\omega_i(x)dydx \\
=\displaystyle\int_{\Sigma\times\Gamma_g}\frac{\partial G}{\partial\nu_x}\lambda^{inc}(y)\omega_i(x)dydx + \frac{1}{ik}\oint_{\Sigma\times\Gamma_g}\frac{\partial^2 G}{\partial\nu_x\partial\nu_y}\phi^{inc}(y)\omega_i(x)dydx \qquad \forall\omega_i
\end{array}\right. \tag{6.24}$$

6.2.4 Ecriture matricielle

Nous allons passer en notation matricielle pour la suite. Soient donc les matrices :

$$X^{inc} = \begin{bmatrix} u_m^{inc} \\ v_m^{inc} \end{bmatrix} \qquad X^{diff} = \begin{bmatrix} u_m^{diff} \\ v_m^{diff} \end{bmatrix} \qquad \alpha = [\alpha_m] \qquad \beta = [\beta_m]$$

X^{inc} et X^{diff} sont les projections des modes incidents et diffractés sur les fonctions de base sur Γ_g. Ce sont des matrices à M^{inc} et M^{diff} colonnes, et $n_{dl}^{\Gamma_g}$ lignes. On a :

$$X^{inc}.\alpha = \begin{bmatrix} \phi^{inc} \\ \lambda^{inc} \end{bmatrix} \qquad X^{diff}.\beta = \begin{bmatrix} \phi^{diff}|_{\Gamma_g} \\ \lambda^{diff} \end{bmatrix}$$

Soient également les matrices (avec toujours les notations (6.2) pour D, K, K', et S) :

$$B = \begin{bmatrix} -\frac{1}{ik}D \\ -K' \end{bmatrix} \qquad C = \begin{bmatrix} -\frac{1}{ik}D & -K \\ -K' & -ikS \end{bmatrix}$$

La troisième équation de (6.24) peut s'écrire :

$$-\frac{1}{ik}D\phi_j + {}^tBX^{diff}.\beta = -{}^tBX^{inc}.\alpha$$

On va sommer les deux premières équations de (6.24). Remarquons que les termes d'intégrales simples sur Γ_g dans les membres de gauche s'éliminent. En effet, on a :

$$\frac{1}{2}\int_{\Gamma_g}\phi^{diff}(x)v_m^{diff}(x)dx = \frac{1}{2}\int_{\Gamma_g}\sum_q \beta_q u_q^{diff}(x)v_m^{diff}(x)dx$$

Sachant que $v_m^{diff} = -\frac{1}{ik}\frac{\partial u_m^{diff}}{\partial \nu} = -\frac{\gamma_m}{k}u_m^{diff}$ (cf. (6.21)), et sachant que les modes guidés sont orthogonaux, seul le terme $q = m$ reste. L'intégrale ci-dessus est donc égale à :

$$-\frac{1}{2}\int_{\Gamma_g} \frac{\gamma_m}{k}\beta_m u_m^{diff}(x)u_m^{diff}(x)dx$$

Le coefficient $i\gamma_m$ vient de la dépendance de u_m^{diff} par rapport à z (cf. (6.21)). Or le même calcul sur l'intégrale $-\frac{1}{2}\int_{\Gamma_g}\lambda^{diff}(x)u_m^{diff}(x)dx$ (de la seconde équation de (6.24)) donnerait l'opposé.

En revanche, les intégrales des seconds membres

$$-\frac{1}{2}\int_{\Gamma_g}\phi^{inc}(x)v_m^{diff}(x)dx \text{ et } \frac{1}{2}\int_{\Gamma_g}\lambda^{inc}(x)u_m^{diff}(x)dx$$

sont égales et leur somme vaut (au choix) :

$$-\int_{\Gamma_g}\phi^{inc}(x)v_m^{diff}(x)dx = \int_{\Gamma_g}\lambda^{inc}(x)u_m^{diff}(x)dx = \int_{\Gamma_g}\alpha_m\frac{\gamma_m}{k}u_m^{inc}(x)u_m^{diff}(x)dx$$

Ici, la simplification n'a pas lieu car les modes incidents et diffractés ont des dépendances en z avec des signes opposés ($e^{i\gamma_m z}$ pour les incidents, $e^{-i\gamma_m z}$ pour les diffractés).

La somme des deux premières équations de (6.24) peut s'écrire :

$${}^tX^{diff}B\phi_j + {}^tX^{diff}CX^{diff}.\beta = -{}^tX^{diff}(C+I_g)X^{inc}.\alpha$$

où I_g est la matrice dont l'élément (i,j) vaut $\int_{\Gamma_g}\omega_i(x)\chi_j(x)dx$.

Au final, on est donc amené à résoudre :

$$\begin{bmatrix} -\frac{1}{ik}D & {}^tBX^{diff} \\ {}^tX^{diff}B & {}^tX^{diff}CX^{diff} \end{bmatrix} \times \begin{bmatrix} \phi_j \\ \beta \end{bmatrix} = -\begin{bmatrix} {}^tBX^{inc} \\ {}^tX^{diff}(C+I_g)X^{inc} \end{bmatrix}.\alpha \tag{6.25}$$

La résolution de ce système linéaire conduit à déterminer tous les coefficients ϕ_j et β_m. On en déduit ϕ et λ (ou p) sur Γ_g grâce aux conditions aux limites de (6.22). Le théorème de représentation (6.19) permet ensuite de connaître le champ de pression u en tout point de l'espace.

6.2.5 Résolution numérique classique

On pose $C' = {}^tX^{diff}CX^{diff}$. En pratique, pour résoudre ce système avec un solveur itératif, on sort β de la seconde équation, et on l'insère dans la première :

$$\begin{cases} \beta = C'^{-1}\left(-{}^tX^{diff}(C+I_g)X^{inc}.\alpha - {}^tX^{diff}B\phi_j\right), \\ \left(-\dfrac{1}{ik}D - {}^tBX^{diff}.C'^{-1}{}^tX^{diff}B\right)\phi_j \\ \qquad = -{}^tBX^{inc}.\alpha - {}^tBX^{diff}.C'^{-1}(-{}^tX^{diff}(C+I_g)X^{inc}.\alpha) \end{cases}$$

L'inversion de C' n'est pas très coûteuse, car c'est une matrice de taille $M^{diff} \times M^{diff}$ (rappelons qu'en pratique M^{diff} et M^{inc} valent au plus 100). Par ailleurs, on peut remarquer que toutes les matrices nouvelles apparaissant ici (B et C) sont identiques à celle utilisée dans le cas d'une onde plane incidente (formule (6.7)) au terme en η près. De même que cette matrice

A représentait l'action de S sur lui-même, B traduit l'action de Σ sur Γ_g et C l'action de Γ_g sur lui-même, le terme d'impédance en moins. L'assemblage de ces matrices ne présente donc pas de nouveautés par rapport au cas d'une onde incidente. En utilisant un solveur itératif, on n'est même pas obligé d'assembler la matrice entière du problème $-1/ikD - {}^tBX^{diff}.C'^{-1t}X^{diff}B$.

6.2.6 Résolution numérique multipôle

Rappelons que l'on cherche à résoudre le système matriciel suivant :

$$\begin{bmatrix} -\frac{1}{ik}D & {}^tBX^{diff} \\ {}^tX^{diff}B & {}^tX^{diff}CX^{diff} \end{bmatrix} \times \begin{bmatrix} \phi_j \\ \beta \end{bmatrix} = - \begin{bmatrix} {}^tBX^{inc} \\ {}^tX^{diff}(C+I_g)X^{inc} \end{bmatrix}.\alpha \tag{6.26}$$

On note toujours M^{inc} et M^{diff} le nombre de modes incidents et diffractés pris en compte dans notre calcul, $n_{dl}^{\Gamma_g}$ et n_{dl}^{Σ} le nombre de degrés de liberté se trouvant sur Γ_g et Σ, et enfin n_{rhs} le nombre de seconds membres du système. Le tableau 6.5 précise les dimensions des différentes matrices apparaissant dans (6.26).

Nom	Taille
D	$n_{dl}^{\Sigma} \times n_{dl}^{\Sigma}$
C	$n_{dl}^{\Gamma_g} \times n_{dl}^{\Gamma_g}$
I_g	$n_{dl}^{\Gamma_g} \times n_{dl}^{\Gamma_g}$
B	$n_{dl}^{\Gamma_g} \times n_{dl}^{\Sigma}$
X^{inc}	$n_{dl}^{\Gamma_g} \times M^{inc}$
X^{diff}	$n_{dl}^{\Gamma_g} \times M^{diff}$
ϕ_j	$n_{dl}^{\Sigma} \times n_{rhs}$
β	$M^{diff} \times n_{rhs}$
α	$M^{inc} \times n_{rhs}$

TAB. 6.5 – *Dimensions des matrices*

Plaçons nous dans le cas d'un calcul de grande taille : $n_{dl}^{\Sigma} = 10^6$, $n_{dl}^{\Gamma_g} = 10^5$, $M^{inc} = 100$, $M^{diff} = 100$ et $n_{rhs} = 100$. On voit que les trois matrices B, C et D (qui contiennent entre 10^{10} et 10^{12} éléments) doivent absolument être gérées en mode multipôle. I_g ne pose pas de problème puisqu'elle est creuse.

La réalisation de produit multipôle pour les matrices carrées D et C ne présente pas de difficulté nouvelle. L'utilisation de la FMM pour C permet d'assembler CX^{diff} colonne par colonne en M^{diff} produits multipôles, et d'en déduire C'. Précisons que pour le moment notre produit multipôle ne sait pas faire de transposition : on traite donc ${}^tB$ comme une matrice à part, indépendante de B.

On peut souligner trois points intéressants au niveau de l'implémentation :

- Tout d'abord, l'adaptation à la méthode multipôle de cet algorithme montre qu'il est tout à fait possible de traiter avec la FMM des formulations élaborées, qui dépasse le cadre de la simple résolution d'un système linéaire.
- Ensuite, on est amené à gérer simultanément plusieurs FMM séparées, puisque ces différentes matrices (B, C, D) s'appuient sur des portions différentes du maillage. Cette cohabitation ne pose bien sûr aucun problème.

– Enfin, la nouveauté la plus délicate à gérer en terme de programmation est l'apparition d'une FMM non-carrée. Nous allons nous attarder un peu plus sur ce dernier point.

En effet, dans tous les cas vus jusqu'à maintenant, les matrices considérées étaient toujours carrées. Les degrés de libertés sur les lignes et les colonnes étaient les mêmes, et au niveau de la FMM, les initialisations et les intégrations se faisaient sur les mêmes inconnues. Ici, avec la matrice B, cela change car ses inconnues « lignes » sont sur Γ_g et ses inconnues « colonnes » sont sur Σ. On est alors amené à construire deux octrees, un octree ligne basé sur le maillage de Γ_g , et un octree colonne basé sur le maillage de Σ.

Les étapes d'initialisation et de montée se font sur l'arbre colonne, qui est en quelque sorte l'arbre de départ. Les transferts marquent le passage de l'arbre colonne à l'arbre ligne. Enfin, les étapes de descente et d'intégration se font sur l'arbre ligne, qui est l'arbre d'arrivée. Alors que dans le cas classique, le même arbre porte à la fois les fonctions de radiation $\mathcal{F}$ et $\mathcal{G}$, ici les fonctions $\mathcal{F}$ sont sur l'arbre colonne, et les fonctions $\mathcal{G}$ sur l'arbre ligne.

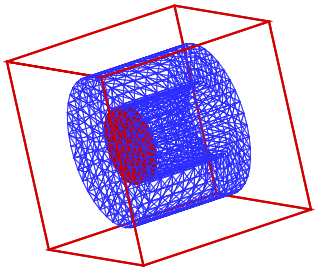
Il est important de souligner que ces deux arbres ne sont pas indépendants : pour pouvoir faire des opérations de transfert de l'un sur l'autre, on a besoin d'avoir des notions de voisins et de banlieues pour des cellules situées sur deux arbres distincts. Si $\mathcal{C}$ est une cellule de l'arbre ligne, et $\mathcal{C}'$ une cellule de l'arbre colonne, il faut pouvoir dire si ces deux cellules sont voisines ou non, et si leurs parents respectifs sont voisins ou non. Le plus simple est de construire ces deux octrees à partir d'une boîte initiale commune, englobant les deux maillages, puis de faire des subdivisions séparées : on parlera d'octrees « associés ». Ainsi les cellules des deux arbres auront la même taille à chaque niveau, et seront inscrites dans la même grille 3D.

La figure 6.5 représente un exemple de construction des deux octrees associés dans le cas d'un guide d'ondes. Dans la colonne de gauche, la frontière du guide d'ondes Γ_g est représentée en « plein » (en rouge, si vous avez la couleur), tandis que le reste du maillage Σ est en fil de fer (bleu). A droite, c'est Σ qui est en plein (Γ_g est alors caché). Autour de ces maillages, on a représenté les premiers niveaux de deux octrees associés construit sur Γ_g et Σ. Comme on peut le voir, le niveau 0 est commun aux deux arbres, les niveaux suivants sont spécifiques.

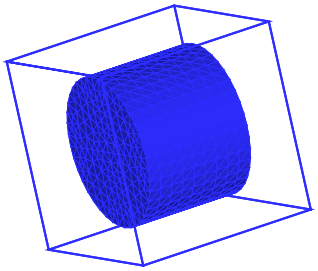
Le tableau 6.6 résume les besoins en terme d'octree pour les 4 matrices à traiter dans le cas d'un guide d'ondes. Il est important de souligner qu'un octree « indépendant » basé sur Γ_g (comme celui de C) sera différent des octrees associés basés sur Γ_g (comme ceux de B et ${}^tB$). Même s'ils sont construits sur le même maillage, ils n'ont a priori pas la même racine. Notons enfin qu'il est sans doute possible de diminuer le nombre d'octrees mis en jeu, mais que cela n'a pas été jugé prioritaire.

Matrice	Octree
D	1 octree sur Σ
C	1 octree sur Γ_g
B	2 octrees associés sur $(\Gamma_g \times \Sigma)$
${}^tB$	2 octrees associés sur $(\Sigma \times \Gamma_g)$
Total	4 FMM, 6 octrees

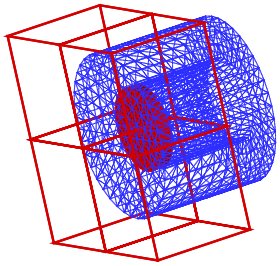
TAB. 6.6 – *Octrees et FMM dans le cas d'un guide d'ondes*



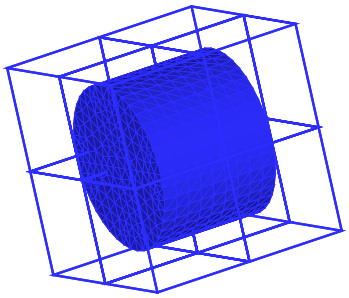
(a) Octree Γ_g, niveau 0



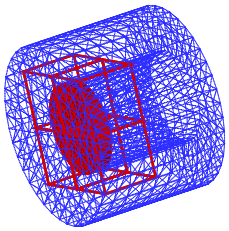
(b) Octree Σ, niveau 0



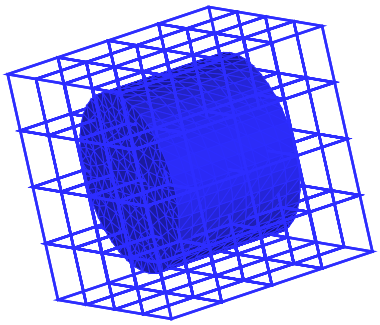
(c) Octree Γ_g, niveau 1



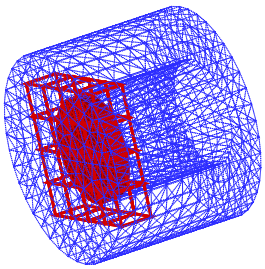
(d) Octree Σ, niveau 1



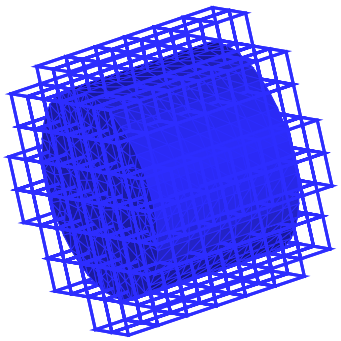
(e) Octree Γ_g, niveau 2



(f) Octree Σ, niveau 2



(g) Octree Γ_g, niveau 3



(h) Octree Σ, niveau 3

FIG. 6.5 – *Octrees associés dans le cas d'un guide d'ondes*

6.2.7 Application

Comme dans le cas d'une onde incidente, nous allons séparer cette section en deux parties consacrées à la comparaison avec les méthodes classiques de résolution, puis à des tests sur des problèmes de grande taille.

6.2.7.1 Comparaison avec les méthodes usuelles

On dispose de deux petits maillages à 1394 et 2438 inconnues. Un tiers de ces inconnues sont situées sur le guide d'ondes Γ_g, le reste sur la partie rigide Σ. Dans les 2 cas, on considère 6 modes incidents et 6 modes diffractés. On résout le problème (6.25) pour 6 seconds membres (chaque mode incident, un par un). On obtient ainsi une matrice de coefficients de réflexion β de taille 6×6. Regardons tout d'abord les temps de résolution (sur PC 866 MHz) :

	Guide1394	Guide2438
Solveur direct :		
initialisation	149 s	430 s
résolution	2 s	6 s
Solveur itératif :		
initialisation	94 s	262 s
résolution	45 s	128 s
Solveur multipôle :		
initialisation	65 s	41 s
résolution	629 s	1160 s

TAB. 6.7 – *Temps d'exécution pour chaque type de solveur (sur PC 866 MHz)*

On a ici utilisé un solveur itératif QMR multi seconds membres [7] avec $\varepsilon = 10^{-3}$ pour tolérance. Les nombres d'itérations sont de 20 sur les deux maillages. Sur des problèmes trop petits, la méthode multipôle n'est bien sûr pas rentable (d'autant qu'elle n'est pas encore optimisée pour traiter plusieurs seconds membres à la fois, ce qui la pénalise par rapport à un solveur itératif classique, et a fortiori face à un solveur direct). En l'occurrence, l'objectif ici était plutôt de vérifier la validité des résultats, ce qui va être fait dans le tableau suivant :

	Guide1394	Guide2438
Solveur itératif	$2{,}0.10^{-4}$	$7{,}7.10^{-4}$
Solveur multipôle	1,4 %	1,5 %

TAB. 6.8 – *Précision pour différents types de solveurs itératifs*

6.2.7.2 Tests sur un problème de grande taille

Nous avons utilisé ici un maillage comportant 99229 inconnues, dont environ 15000 situées sur la frontière du guide d'ondes. Pour un problème de cette taille, les méthodes classiques ne sont envisageables qu'à l'aide de calculateurs parallèles assez importants, ne serait-ce que pour stocker la matrice de 160 Go. Ici, nous allons résoudre notre problème pour un seul mode incident et un seul mode diffracté sur un simple PC. Comme souvent sur des problèmes de grande taille, et en l'absence de préconditionneur, le principal problème vient de la difficulté

qu'éprouve le solveur itératif à converger. Sur ce cas, un résidu de 10 % est atteint en 28 itérations, 5 % en 54 itérations, et 2 % après 100 itérations. Par conséquent, malgré un temps de démarrage du calcul (i.e. création des arbres et assemblage des matrices) très faible, l'ensemble de la résolution requiert près de 7 heures.

	Guide 99229
Temps de démarrage	1762 s
Temps par itération	216 s
Mémoire vive	54 Mo
Espace disque	2,1 Go

TAB. 6.9 – *Guide d'ondes de grande taille (sur PC 866 MHz)*

L'objectif ici était de démontrer la capacité de la méthode multipôle à traiter de gros cas de calcul dans cette formulation avec guide d'ondes même sur des stations de travail standards. Il y a évidemment une grande marge de progression sur le préconditionnement, le solveur itératif, les traitements multi seconds membres, la FMM à précision variable, etc.

Conclusion

Nous avons présenté l'implémentation de notre méthode multipôle rapide dans le code intégral fréquentiel acoustique de EADS. Nous avons pu tester la FMM d'une part sur des calculs d'interaction entre un objet (éventuellement traité) et une onde incidente, d'autre part sur des cas comprenant un guide d'ondes. Nous avons ainsi enrichi notre FMM d'une nouvelle formulation acoustique à une composante, ainsi que de la capacité à traiter des matrices non-carrées. Lors des tests, nous avons retrouvé les qualités de vitesse et de précision de ce code déjà constatées dans le cas électromagnétique. Nous avons également rencontré les mêmes limitations, principalement inhérentes à l'utilisation de solveurs itératifs. Ces limites pourront être levées par l'utilisation de solveurs plus adaptés à la FMM et de préconditionneurs.

Chapitre 7

Autres utilisations de la FMM

Sommaire

Introduction **247**
7.1 Champs proches **248**
7.2 Champs lointains **255**
7.3 Matériaux diélectriques **257**
7.4 Objets hétérogènes **265**
7.5 Structures filaires minces **271**
Conclusion **274**

Introduction

Nous allons présenter de nouveaux champs d'application et de nouvelles formulations de la méthode multipôle rapide. Fondamentalement, il s'agit toujours de la même FMM telle qu'elle a été présentée précédemment. Nous avons vu qu'elle pouvait s'appliquer aux équations de Maxwell dans le cas d'objets parfaitement conducteurs, ou à l'équation d'Helmholtz dans le cas d'objets rigides ou traités. Nous allons maintenant exposer d'autres problèmes pouvant être traités à l'aide d'une méthode multipôle.

D'une part, nous allons nous intéresser aux post-traitements qui généralement accompagnent la résolution des équations de Maxwell : les calculs de champs proches et de champs lointains. Nous verrons que ces derniers font quasiment partie de la FMM, et peuvent être obtenus avec un surcoût presque nul. D'autre part, nous essaierons d'enrichir les problèmes électromagnétiques résolus, en abordant le cas des matériaux diélectriques puis des objets hétérogènes composés de plusieurs sous-domaines homogènes, parfaitement conducteurs ou diélectriques, ainsi que de fils.

Les présentations faites ici sont pour le moment purement théoriques, et n'ont pas encore été effectivement mises en œuvre. Le but n'est pas de présenter de nouveaux développements informatiques, mais de montrer que la méthode multipôle peut s'appliquer dans de nombreux cas.

7.1 Champs proches

7.1.1 Objectif

On se place dans le cas simple d'un objet Ω parfaitement conducteur placé dans le vide et illuminé par une onde plane. Le champ électromagnétique $(\vec{E},\vec{H})$ en un point $y \notin \Omega$ est la somme du champ incident et du champ rayonné par le courant surfacique $\vec{j}$ porté par Γ. Le calcul de ce dernier se fait à partir du théorème de représentation des équations de Maxwell :

$$\left\{\begin{array}{ll} \vec{E}(y) = i\omega\mu_0 \left(\displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx + \frac{1}{k^2}\vec{grad}_y \int_\Gamma G(|y-x|)div_\Gamma \vec{j}(x)dx \right) & y \in \mathbb{R}^3 - \Omega, \\ \vec{H}(y) = \vec{rot}_y \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx & y \in \mathbb{R}^3 - \Omega \end{array}\right. \tag{7.1}$$

où

$$G(R) = \frac{e^{ikR}}{4\pi R}$$

est la solution élémentaire de l'équation de Helmholtz en 3D.

Le but de cette section est de trouver une formulation multipôle permettant de calculer de manière rapide $(\vec{E},\vec{H})$.

7.1.2 Ecriture classique

Le point $y \in \mathbb{R}^3 - \Omega$ est une donnée. Il est choisi par l'utilisateur. La fonction de courant surfacique $\vec{j}$ est une autre donnée du problème. Elle sera le plus souvent issue de la résolution préalable des équations de Maxwell par formulation intégrale. Elle appartient donc à l'espace d'éléments finis présenté à la section 1.1.3, et s'écrit :

$$\vec{j}(x) = \sum_{1 \le i \le n_{dl}} \lambda_i \vec{\varphi}_i(x)$$

Les champs proches $\vec{E}(y)$ et $\vec{H}(y)$ se calculent de la manière suivante :

$$\left\{\begin{array}{l} \vec{E}(y) = i\omega\mu_0 \displaystyle\sum_{1 \le i \le n_{dl}} \lambda_i \left(\int_\Gamma G(|y-x|)\vec{\varphi}_i(x)dx + \frac{1}{k^2}\vec{grad}_y \int_\Gamma G(|y-x|)div\vec{\varphi}_i(x)dx \right), \\ \vec{H}(y) = \displaystyle\sum_{1 \le i \le n_{dl}} \lambda_i \left(\vec{rot}_y \int_\Gamma G(|y-x|)\vec{\varphi}_i(x)dx \right) \end{array}\right. \tag{7.2}$$

On note $\vec{e_i}(y)$ et $\vec{h_i}(y)$ les termes entre parenthèses dans (7.2) :

$$\left\{\begin{array}{l} \vec{e_i}(y) = \displaystyle\int_\Gamma G(|y-x|)\vec{\varphi}_i(x)dx + \frac{1}{k^2}\vec{grad}_y \int_\Gamma G(|y-x|)div\vec{\varphi}_i(x)dx, \\ \vec{h_i}(y) = \vec{rot}_y \displaystyle\int_\Gamma G(|y-x|)\vec{\varphi}_i(x)dx \end{array}\right. \tag{7.3}$$

Les intégrales de (7.3) se calculent très simplement. En effet, les fonctions $\vec{\varphi}_i$ ont un support réduit à deux triangles. Les $\int_\Gamma$ sont alors calculées de manière soit numérique, soit analytique,

selon la distance entre le point y et le support de $\vec{\varphi}_i$. Les champs proches s'obtiennent alors en écrivant :

$$\begin{cases} \vec{E}(y) = i\omega\mu_0 \sum_{1\leq i\leq n_{dl}} \lambda_i \vec{e_i}(y), \\ \vec{H}(y) = \sum_{1\leq i\leq n_{dl}} \lambda_i \vec{h_i}(y) \end{cases}$$

En pratique, il s'agit d'un produit scalaire entre le vecteur (λ_i) issu de la résolution de la formulation intégrale, et les vecteurs $(\vec{e_i}(y))$ et $(\vec{h_i}(y))$. A chaque point y correspond un couple de vecteurs $(\vec{e_i},\vec{h_i})$. Si on note N_{pr} le nombre de points y où l'on souhaite calculer les champs proches, on voit que le coût de l'ensemble du calcul est en $\mathcal{O}(n_{dl}N_{pr})$. Nous reviendrons à la section 7.1.4.4 sur l'intérêt d'utiliser une formulation multipôle pour le traitement des champs proches.

7.1.3 Algorithme multipôle

7.1.3.1 Algorithme multipôle à un niveau

Pour simplifier, on va dans un premier temps chercher un algorithme multipôle à un seul niveau répondant à notre besoin. On se place dans la configuration représentée figure 7.1. Le point x se trouve sur Γ, dans la cellule $\mathcal{C}$ centrée en M, le point y se trouve hors de Ω, dans la cellule $\mathcal{C}'$ centrée en M'. On suppose que $\mathcal{C}$ et $\mathcal{C}'$ ne sont pas voisines. On calcule le champ diffracté par le courant $\vec{j}(x)$ au point y.

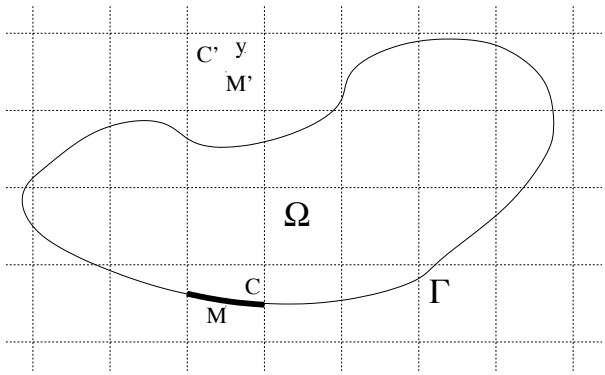


FIG. 7.1 – *Découpage de Γ pour le calcul du champ proche*

On a la décomposition suivante pour le noyau de Green :

$$G(|y-x|) = \frac{e^{ik|y-x|}}{4\pi|y-x|} = \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}} T_{M\vec{M}'}(\vec{s}) e^{ik\vec{s}.\vec{M'}y} d\vec{s} \tag{7.4}$$

Dans cette formule, le nombre de pôles L (tel qu'il apparaît dans (1.4)) est fixé et implicite : l'égalité (7.4) n'est vraie qu'à une erreur ε près. D'après (7.1), la contribution de la cellule $\mathcal{C}$ au champ électromagnétique $(\vec{E},\vec{H})$ en y s'écrit comme somme de trois termes. On note :

$$\vec{E}(y) = i\omega\mu_0 \left(\vec{E_1}(y) + \vec{E_2}(y)\right)$$

et la contribution de $\mathcal{C}$ s'écrit avec les trois termes suivants :

$$\begin{cases} \vec{E_1}(y) = \displaystyle\int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)\vec{j}(x)dx, \\ \vec{E_2}(y) = \dfrac{1}{k^2}\vec{grad}_y \displaystyle\int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)div\vec{j}(x)dx, \\ \vec{H}(y) = \vec{rot}_y \displaystyle\int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)\vec{j}(x)dx \end{cases}$$

7.1.3.2 Premier terme

En utilisant (7.4), le premier terme dans l'expression du champ $\vec{E}(y)$,s'écrit :

$$\begin{aligned} \vec{E_1}(y) &= \int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)\vec{j}(x)dx \\ &= \frac{ik}{16\pi^2}\int_{x\in\Gamma\cap\mathcal{C}} \left(\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}} T_{M\vec{M}'}(\vec{s}) e^{ik\vec{s}.\vec{M'}y} d\vec{s}\right)\vec{j}(x)dx \\ &= \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y} T_{M\vec{M}'}(\vec{s}) \left(\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\vec{j}(x)dx\right) d\vec{s} \end{aligned}$$

On retrouve les trois étapes bien connues d'une méthode multipôle à un niveau :

Initialisation : on calcule la fonction $\vec{\mathcal{F}_\mathcal{C}}$ définie sur $\mathcal{S}$ par :

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\vec{j}(x)dx$$

Transfert : on multiplie la fonction $\vec{\mathcal{F}_\mathcal{C}}$ par la fonction de transfert $T_{M\vec{M}'}$.
Intégration : on termine le calcul en intégrant le résultat du transfert sur $\mathcal{S}$:

$$\frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}} \left[T_{M\vec{M}'}(\vec{s})\vec{\mathcal{F}_\mathcal{C}}(\vec{s})\right] e^{ik\vec{s}.\vec{M'}y} d\vec{s}$$

7.1.3.3 Second terme

Le second terme apparaissant dans l'expression (7.1) s'écrit :

$$\begin{aligned} \vec{E_2}(y) &= \frac{1}{k^2}\vec{grad}_y \int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)div\vec{j}(x)dx \\ &= \frac{ik}{16\pi^2}\frac{1}{k^2}\vec{grad}_y \int_{x\in\Gamma\cap\mathcal{C}} \left(\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}} T_{M\vec{M}'}(\vec{s}) e^{ik\vec{s}.\vec{M'}y} d\vec{s}\right) div\vec{j}(x)dx \end{aligned}$$

Le $\vec{grad}_y$ ne s'applique qu'au terme $e^{ik\vec{s}.\vec{M'}y}$ et donne :

$$\vec{grad}_y e^{ik\vec{s}.\vec{M'}y} = ik\vec{s}e^{ik\vec{s}.\vec{M'}y}$$

D'autre part, on fait une intégration par parties sur la variable x qui transforme la divergence en $-\vec{grad}_x$ sur le terme $e^{ik\vec{s}.x\vec{M}}$. Cette intégration par partie ne génère pas de termes de bord, comme expliqué à la section 1.2.2.3.1. On obtient :

$$\begin{aligned}
\vec{E_2}(y) &= \frac{ik}{16\pi^2}\frac{1}{k^2}\int_{x\in\Gamma\cap\mathcal{C}}\int_{\vec{s}\in\mathcal{S}}\left(-\vec{grad}_x e^{ik\vec{s}.x\vec{M}}.\vec{j}(x)\right)T_{M\vec{M}'}(\vec{s})\left(\vec{grad}_y e^{ik\vec{s}.\vec{M'}y}\right)dxd\vec{s}\\
&= \frac{ik}{16\pi^2}\frac{1}{k^2}\int_{x\in\Gamma\cap\mathcal{C}}\int_{\vec{s}\in\mathcal{S}}\left(ik\vec{s}e^{ik\vec{s}.x\vec{M}}.\vec{j}(x)\right)T_{M\vec{M}'}(\vec{s})\left(ik\vec{s}e^{ik\vec{s}.\vec{M'}y}\right)dxd\vec{s}\\
&= -\frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}e^{ik\vec{s}.\vec{M'}y}T_{M\vec{M}'}(\vec{s})\left(\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}(\vec{s}.\vec{j}(x)\vec{s})dx\right)d\vec{s}
\end{aligned}$$

Ce terme peut être traité en même temps que le précédent, en remplaçant juste la fonction de radiation $\vec{\mathcal{F}_\mathcal{C}}$ par :

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}(\vec{j}(x)-\vec{s}.\vec{j}(x)\vec{s})dx \tag{7.5}$$

On voit que seule la composante de $\vec{j}$ tangente à $\mathcal{S}$ sert. L'étape d'intégration reste inchangée.

7.1.3.4 Troisième terme

Le troisième terme est l'expression du champ magnétique $\vec{H}$. A l'aide du théorème d'addition, on peut écrire :

$$\begin{aligned}
\vec{H}(y) &= \vec{rot}_y\int_{x\in\Gamma\cap\mathcal{C}}G(|y-x|)\vec{j}(x)dx\\
&= \frac{ik}{16\pi^2}\vec{rot}_y\int_{x\in\Gamma\cap\mathcal{C}}\left(\int_{\vec{s}\in\mathcal{S}}e^{ik\vec{s}.x\vec{M}}T_{M\vec{M}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}\right)\vec{j}(x)dx
\end{aligned}$$

Le $\vec{rot}_y$ ne s'applique qu'au vecteur $e^{ik\vec{s}.\vec{M'}y}\vec{j}(x)$, et il donne :

$$\vec{rot}_y\left(e^{ik\vec{s}.\vec{M'}y}\vec{j}(x)\right) = e^{ik\vec{s}.\vec{M'}y}ik\vec{s}\wedge\vec{j}(x)$$

D'où l'expression de $\vec{H}$:

$$\begin{aligned}
\vec{H}(y) &= \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}T_{M\vec{M}'}(\vec{s})\vec{rot}_y(e^{ik\vec{s}.\vec{M'}y}\vec{j}(x))dxd\vec{s}\\
&= \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}T_{M\vec{M}'}(\vec{s})(e^{ik\vec{s}.\vec{M'}y}ik\vec{s}\wedge\vec{j}(x))dxd\vec{s}\\
&= \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}e^{ik\vec{s}.\vec{M'}y}ik\vec{s}\wedge\left[T_{M\vec{M}'}(\vec{s})\left(\int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}\vec{j}(x))dx\right)\right]d\vec{s}
\end{aligned}$$

On peut alors écrire les trois étapes du calcul multipôle :

Initialisation : on calcule la fonction $\vec{\mathcal{F}_\mathcal{C}}$ définie sur $\mathcal{S}$ par :

$$\vec{\mathcal{F}_\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}}e^{ik\vec{s}.x\vec{M}}\vec{j}(x)dx$$

Transfert : on multiplie la fonction $\vec{\mathcal{F}_\mathcal{C}}$ par la fonction de transfert $T_{M\vec{M}'}$.

Intégration : on termine le calcul en intégrant le résultat du transfert sur $\mathcal{S}$:

$$-\frac{k^2}{16\pi^2}\int_{\vec{s}\in\mathcal{S}} \vec{s}\wedge\left[T_{M\vec{M}'}(\vec{s})\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})\right]e^{ik\vec{s}.\vec{M'}y}d\vec{s}$$

Remarquons que grâce au terme $\vec{s}\wedge$, la composante de $\vec{j}$ dirigée selon $\vec{s}$ n'intervient pas. On peut donc très bien utiliser l'étape d'initialisation (7.5).

7.1.3.5 Synthèse

On va donc proposer la définition de la *FMM* mono-niveau pour le calcul du champ proche en un point $y \notin \Omega$ situé dans la cellule $\mathcal{C}'$.

Interactions proches : on passe en revue les cellules $\mathcal{C}$ voisines de $\mathcal{C}'$ pour calculer de manière classique le terme d'interaction :

$$\left\{\begin{array}{l} \displaystyle\sum_{\mathcal{C}\in v(\mathcal{C}')} i\omega\mu_0\int_{x\in\Gamma\cap\mathcal{C}}\left(G(|y-x|)\vec{j}(x)+\frac{1}{k^2}\vec{grad}_y G(|y-x|)div\vec{j}(x)\right)dx, \\ \displaystyle\sum_{\mathcal{C}\in v(\mathcal{C}')} \vec{rot}_y\int_{x\in\Gamma\cap\mathcal{C}} G(|y-x|)\vec{j}(x)dx \end{array}\right. \tag{7.6}$$

Le résultat de ces intégrations constitue la partie "proche" du champ électromagnétique $(\vec{E}(y),\vec{H}(y))$. Dans le cas FMM « habituel », les interactions proches sont traitées via des petits produits matrice-vecteur pleins dont les dimensions sont les nombres de degrés de liberté contenus dans les deux cellules $\mathcal{C}$ et $\mathcal{C}'$. Ici, dans le cas d'une FMM pour calculer des champs proches, les intégrales simples de (7.6) seront traitées numériquement via des produits scalaires.

Interactions Lointaines : le calcul se fait en trois étapes.

- Initialisation : pour toute cellule $\mathcal{C}$, on calcule la fonction de radiation

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{j}(x)-\vec{s}.\vec{j}(x)\vec{s})dx$$

  en tout point $\vec{s}$ de notre quadrature de $\mathcal{S}$. Il faut souligner que cette formule d'initialisation est la même que pour la FMM normale.
- Transfert : $\mathcal{C}'$ étant fixée, on passe en revue les cellules $\mathcal{C}$ non-voisines de $\mathcal{C}'$ pour calculer :

$$\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s}) = \sum_{\mathcal{C}\notin v(\mathcal{C}')} T_{M\vec{M}'}(\vec{s}).\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})$$

- Intégration : on calcule les deux intégrales :

$$\left\{\begin{array}{l} \displaystyle i\omega\mu_0\frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s}, \\ \displaystyle -\frac{k^2}{16\pi^2}\int_{\vec{s}\in\mathcal{S}}\vec{s}\wedge\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s})e^{ik\vec{s}.\vec{M'}y}d\vec{s} \end{array}\right.$$

  Le résultat de ces intégrations constitue la partie "lointaine" du champ électromagnétique $(\vec{E}(y),\vec{H}(y))$, et se rajoute tout naturellement à la partie "proche".

7.1.4 Approfondissement

7.1.4.1 Formulation à deux termes

Comme on l'a fait pour la formulation *EFIE*, on peut écrire une *FMM* à deux termes. En effet, la composante tangentielle de $\vec{j}$ sur la sphère unité $\mathcal{S}$ peut s'écrire :

$$\vec{j}(x) - (\vec{j}(x).\vec{s})\vec{s} = (\vec{j}(x).\vec{s_\theta})\vec{s_\theta} + (\vec{j}(x).\vec{s_\phi})\vec{s_\phi}$$

Comme dans le cas de l'*EFIE* multipôle, il suffit de faire suivre l'étape d'initialisation d'un passage aux coordonnées sphériques tangentielles :

$$\begin{cases} (\mathcal{F}_\mathcal{C})_\theta(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\theta}(\vec{s}), \\ (\mathcal{F}_\mathcal{C})_\phi(\vec{s}) = \vec{\mathcal{F}_\mathcal{C}}(\vec{s}).\vec{s_\phi}(\vec{s}) \end{cases}$$

On fait ensuite les transferts sur deux fonctions de radiation au lieu de trois. Enfin, on fait précéder l'étape d'intégration d'un retour aux coordonnées cartésiennes (x,y,z) :

$$\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}) = (\mathcal{G}_{\mathcal{C}'})_\theta(\vec{s}).\vec{s_\theta}(\vec{s}) + (\mathcal{G}_{\mathcal{C}'})_\phi(\vec{s}).\vec{s_\phi}(\vec{s})$$

Cette simple modification permet de gagner un facteur environ 30% en temps d'exécution. Dans le cas d'une FMM multi-niveau, il faut penser à modifier les opérations de montée/descente en conséquence (cf. section 1.3.1.6.4).

7.1.4.2 Algorithme multipôle multi-niveau

L'écriture multi-niveau de l'algorithme multipôle proposé au paragraphe précédent ne diffère quasiment pas de celle faites à la section 1.3.2.2. Les phases de montée, descente, et transferts aux différents niveaux sont identiques. Il y a néanmoins une différence de taille : la montée et la descente se font sur deux arbres différents.

En effet, le calcul des fonctions de radiation $\mathcal{F}$ se fait récursivement dans un arbre construit autour du maillage Γ (comme dans le cas standard, en somme). Les fonctions $\mathcal{G}$ sont destinées à être intégrées aux points y de calcul du champ proche définis par l'utilisateur, elles doivent donc être calculées dans un arbre bâti sur ces points. On retrouve là un point déjà abordé dans le cas acoustique : une FMM utilisant un arbre de départ et un arbre d'arrivée différents. Cela n'introduit pas de difficulté particulière dès lors que l'on a pris soin de construire ces deux arbres dans la même grille et à partir de la même boîte « racine » (afin de préserver les notions de cellules voisines entre les deux arbres).

7.1.4.3 Insertion dans un solveur itératif

On vient de voir que seule l'étape d'intégration et les calculs des interactions proches diffèrent entre la formulation multipôle EFIE et la formulation multipôle champ proche. Les étapes d'initialisation, de montée, de descente et de transfert sont les mêmes. Or elles représentent une part importante du temps de calcul (de l'ordre de 80 à 90 %). Par conséquent, il est naturel d'envisager de calculer le produit matrice-vecteur et les champs proches simultanément via une formulation multipôle mixte. Pour cela, il suffirait de changer la phase d'intégration pour qu'elle puisse gérer à la fois la partie « équation intégrale » et la partie « champ proche ». L'arbre de la FMM devra donc intégrer dans ses feuilles les points où l'utilisateur souhaite calculer le champ proche.

Ce faisant, *on peut calculer les champs proches à chaque itération du solveur avec un surcoût quasiment nul.* Cela présente deux intérêts : d'une part, on réalise un gain en terme de temps, puisque le post-traitement de calcul du champ proche est intégré à la résolution du système linéaire. D'autre part, si ce même champ proche est le but du calcul (c.à.d. le seul résultat intéressant véritablement l'utilisateur), on peut utiliser un critère de convergence non plus sur les valeurs des degrés de liberté (i.e. des courants surfaciques) mais directement sur les valeurs des champs proches, puisqu'on les calcule désormais à chaque itération. Le calcul des champs proches ayant un effet « moyenneur » sur les valeurs des courants, il y a de grandes chances que l'on puisse diminuer le nombre d'itérations requises. Le critère d'arrêt reste à déterminer.

7.1.4.4 Intérêt d'une FMM champ proche

Nous allons clore cette section consacrée au calcul des champs électromagnétiques proches en soulignant le gain en temps conséquent que représente l'utilisation de la méthode multipôle ici. On note n_{dl} le nombre de degrés de liberté, N_{pr} le nombre de points de calcul des champs proches, et N_{rhs} le nombre de seconds membres du système de Maxwell initial. On souhaite calculer les champs proches $\vec{E}$ et $\vec{H}$ pour chacun des N_{rhs} courants surfaciques $\vec{j}(x)$ en chacun des N_{pr} points y.

Méthode classique : On calcule la matrice des coefficients $(\vec{e_i}(y),\vec{h_i}(y))$ de dimension $6N_{pr}\times n_{dl}$, puis on fait les $6N_{pr} \times N_{rhs}$ produits scalaires de chaque colonne de cette matrice pas chaque vecteur solution. On obtient le nombre d'opérations total :

$$n_{classique} = C_1.N_{pr}.n_{dl} + C_2.6.N_{pr}.N_{rhs}.n_{dl}$$

avec $C_1 \approx 100$ (calcul d'un terme de matrice) et $C_2 = 1$ (produit scalaire).

Méthode multipôle : Pour chaque vecteur solution, on fait une exploration complète de l'arbre. Si l'ensemble des points de calcul y choisis occupent un volume similaire à Ω lui-même, et si N_{pr} est inférieur ou du même ordre que n_{dl}, la phase d'intégration de ce calcul aura un coût proche de celui d'une FMM classique. Le nombre total d'opérations sera alors :

$$n_{fmm} = C_3.N_{rhs}.n_{dl}.\log(n_{dl})$$

avec $C_3 \approx 100$ (on avait trouvé pour valeur approchée $C_3 = 119$ à la section 1.3.2.3.6).

Plaçons nous dans un cas extrême : $n_{dl} = 10^6$ inconnues, $N_{rhs} = 100$ courants, $N_{pr} = 10^4$ points de champ proche. On obtient respectivement 7.10^{12} et $1{,}64.10^{11}$ opérations, soit « seulement » un facteur 42 [1]. En pratique, ce sera beaucoup plus, car on a observé pour la FMM des coûts inférieurs, parfois en $\mathcal{O}(n_{dl})$ (cf. section 3.6.4). De plus, si on insère le calcul des champs proches dans la FMM classique, on ramènera le nombre d'opérations à $n'_{fmm} = C_4 N_{rhs} N_{pr}$ avec $C_4 \approx 100$, tout en diminuant le nombre d'itérations.

Ainsi, quelle que soit la manière dont ils sont traités, il est toujours rentable de réaliser les calculs de champs proches à l'aide de la FMM.

1. Ceux qui ont lu « Le guide du routard galactique » connaissaient déjà la réponse...

7.2 Champs lointains

7.2.1 Objectif

Comme dans la section consacrée au champ proche, on considère le cas d'un objet Ω parfaitement conducteur de frontière Γ placé dans le vide. On s'intéresse uniquement aux champs loin de Ω. On fait alors l'hypothèse que l'on est à une distance infinie de celui-ci dans une direction $\vec{\xi} \in \mathcal{S}$ sphère unité de $\mathbb{R}^3$. L'amplitude de l'onde électromagnétique diffractée dans cette direction est donnée par :

$$\vec{A}(\vec{\xi}) = i\omega\mu_0 \left(\int_{x\in\Gamma} e^{-ik\vec{\xi}.\vec{Ox}} (\vec{j}(x) - \vec{\xi}.\vec{j}(x)\vec{\xi}) dx \right) \qquad \vec{\xi} \in \mathcal{S} \tag{7.7}$$

Dans cette formule, O est un point de référence servant à fixer l'origine des phases, son choix n'influe pas sur le module de $\vec{A}(\vec{\xi})$. Cette expression découle des formules de représentations intégrales à l'extérieur de l'obstacle Ω (équation 7.1). Si $y \in \mathbb{R}^3 - \Omega$ est très grand, on a l'équivalent suivant pour le champ électrique $\vec{E}$:

$$\vec{E}(y) = \frac{e^{ik|y|}}{4\pi|y|} \vec{A}(\frac{y}{|y|}) \left(1 + \mathcal{O}(\frac{1}{|y|}) \right)$$

Le champ magnétique s'en déduit puisque la direction est orthogonale et que le rapport des modules vaut $Z_0 = \sqrt{\mu_0/\epsilon_0}$ l'impédance du vide. C'est pourquoi on ne s'intéresse en général qu'à l'amplitude du champ électrique.

7.2.2 Ecriture classique

En pratique, la direction $\vec{\xi}$ étant donnée, on va calculer l'intégrale :

$$\vec{a}(\vec{\xi}) = \int_{x\in\Gamma} e^{-ik\vec{\xi}.\vec{Ox}} \vec{j}(x) dx$$

On déduit $\vec{A}(\vec{\xi}) = i\omega\mu_0 \left(\vec{a}(\vec{\xi}) - \vec{\xi}.\vec{a}(\xi)\vec{\xi} \right)$. $\vec{A}$ est obtenu en ôtant de $\vec{a}$ la partie dirigée selon $\vec{\xi}$, puis en multipliant par la constante $i\omega\mu_0 = ikZ_0$.

Partant d'un courant surfacique $\vec{j}(x) = \sum_{1\leq i\leq n_{dl}} \lambda_i \vec{\varphi}_i(x)$, le vecteur $\vec{a}(\vec{\xi})$ est obtenu en faisant le produit scalaire de (λ_i) par le vecteur $(\vec{a_i})$ dont la i-ème coordonnée vaut :

$$\vec{a_i}(\vec{\xi}) = \int_{x\in\Gamma} e^{-ik\vec{\xi}.\vec{Ox}} \vec{\varphi}_i(x) dx$$

On a alors :

$$\vec{a}(\vec{\xi}) = \sum_{1\leq i\leq n_{dl}} \lambda_i \vec{a_i}(\vec{\xi})$$

On voit que, comme dans le cas du champ proche, le champ lointain est obtenu par un produit scalaire. Remarquons que le vecteur en question peut également servir pour calculer le second membre du système EFIE (1.1) qui vaut :

$$\frac{i}{kZ_0} \int_\Gamma \vec{E}_{inc}(x).\vec{j^t}(x) dx \quad \text{avec} \quad \vec{E}_{inc}(x) = e^{iku_{\vec{inc}}.\vec{Ox}} \quad \text{et} \quad \vec{j^t} = \vec{\varphi}_i \tag{7.8}$$

Au passage, remarquons que si on parvient à écrire un algorithme multipôle pour calculer le champ lointain (7.7), on pourra sans doute aussi l'utiliser pour calculer des seconds membres comme (7.8) (avec toutefois une efficacité à démontrer).

7.2.3 Algorithme multipôle

7.2.3.1 Lien avec les fonctions de radiation

Difficile de ne pas voir une ressemblance entre la formule d'initialisation de l'algorithme champ proche multipôle rappelée ici :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{j}(x) - \vec{s}.\vec{j}(x)\vec{s})dx$$

et l'expression du champ lointain :

$$\vec{A}(\vec{\xi}) = i\omega\mu_0 \left(\int_{x\in\Gamma} e^{-ik\vec{\xi}.\vec{O}x}(\vec{j}(x) - \vec{\xi}.\vec{j}(x)\vec{\xi})dx \right)$$

On a déjà vu que les fonctions de radiation $\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})$ représentaient l'influence sur l'extérieur de $\mathcal{C}$ des courants $\vec{j}(x)$ pour $x \in \Gamma\cap\mathcal{C}$. On a ici une formulation plus rigoureuse de ce concept : $i\omega\mu_0\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})$ est l'amplitude du champ lointain diffracté dans la direction $\vec{s}$ par la portion d'objet $\Gamma\cap\mathcal{C}$. Si on prend pour cellule $\mathcal{C}$ la racine de l'octree (qui contient donc tout Γ), on voit que

$$\vec{A}(\vec{\xi}) = i\omega\mu_0\vec{\mathcal{F}}_{\mathcal{C}}(\vec{\xi})$$

en prenant pour origine des phases le point M centre de $\mathcal{C}$. Le cœur de l'algorithme multipôle classique consistant à calculer des fonctions de radiation $\vec{\mathcal{F}}_{\mathcal{C}}$, on voit qu'il n'y a pas loin à aller pour calculer les champs lointains. Reste à voir comment faire le rapprochement entre les fonctions $\vec{\mathcal{F}}_{\mathcal{C}}$ connues seulement sur une grille (θ_i,ϕ_j) et les amplitudes $\vec{A}(\vec{\xi})$.

7.2.3.2 Différentes écritures

Comme pour les champs proches, on voit que le calcul des champs lointains d'une certaine manière fait partie intégrante de l'algorithme multipôle classique. On va pouvoir proposer différentes approches du problème en fonction du niveau d'imbrication souhaité entre le module FMM et le module champ lointain.

7.2.3.2.1 Approche pure FMM Une première possibilité consiste à ne calculer les champs lointains que dans les directions $\vec{\xi} \in (\theta_i,\phi_j)$. A chaque niveau de l'octree, cette grille a été choisie de manière à conserver (à une erreur ε près) toute l'information spectrale sur la fonction de radiation $\vec{\mathcal{F}}_{\mathcal{C}}$ (à ce sujet, se reporter aux sections consacrées aux problèmes de discrétisation de la sphère unité 1.2.1.6.2 et 1.2.2.2.2). Si l'on considère la racine de l'octree $\mathcal{C}^{(0)}$ (seule cellule au niveau 0), sa fonction de radiation contient toute l'information sur le champ diffracté loin de $\mathcal{C}^{(0)}$. On peut donc réaliser un module de calcul du champ lointain multipôle, qui se contenterait de calculer $\vec{\mathcal{F}_{\mathcal{C}^{(0)}}}$ à partir d'une distribution de courant $\vec{j}(x)$. Cette fonction pourrait ensuite être sauvegardée soit directement, sous la forme de ses valeurs sur la grille $\vec{\mathcal{F}_{\mathcal{C}^{(0)}}}(\theta_i,\phi_j)$, soit après transformée inverse sous la forme de ses coefficients $A_{l,m}$ calculés avec (1.54). On aurait alors :

$$\vec{\mathcal{F}_{\mathcal{C}^{(0)}}}(\vec{s}) = \sum_{\substack{-l\leq m\leq l \\ 0\leq l\leq L^{(0)/2}}} A_{l,m}Y_{l,m}(\vec{s})$$

où $L^{(0)}$ est le nombre de pôles au niveau 0, et $L^{(0)}/2$ est la largeur de bande de $\vec{\mathcal{F}_{\mathcal{C}^{(0)}}}$.

Avantages : la mise au point serait simple, puisqu'il suffirait de prolonger le calcul des fonctions de radiation $\vec{\mathcal{F}_{\mathcal{C}}}$ jusqu'au niveau 0 au lieu de s'arrêter au niveau « plafond ». Le calcul serait rapide, puisqu'il durerait environ la moitié du temps d'un produit FMM entier (pas d'interaction proche, de descente et d'intégration). Ce calcul pourrait même être intégré au solveur itératif (tout comme pour le champ proche). Enfin, ce mode de fonctionnement dispenserait l'utilisateur de fixer les pas angulaires $\delta\theta$ et $\delta\phi$ dans le cas de calcul dans des fenêtres $[\theta_{min},\theta_{max},\delta\theta] \times [\phi_{min},\phi_{max},\delta\phi]$. Un choix optimal serait fait automatiquement par la FMM.

Inconvénients : pour des objets de grande taille, la grille (θ_i,ϕ_j) au niveau 0 comporte environ $240d^2$ points, où d est le diamètre de Ω en nombre de longueurs d'onde ($d > 100$ pour les plus grands cas) . Cela peut donc représenter des tableaux de plusieurs millions d'éléments, là où l'utilisateur peut très bien n'être intéressé que par quelques milliers d'entre eux. En outre, la manipulation d'une unique fonction de radiation $\vec{\mathcal{F}_{\mathcal{C}^{(0)}}}$ rendrait délicate la parallélisation. Enfin, si le niveau plafond choisi est loin de la racine, cette approche nécessite d'explorer des niveaux de l'arbre normalement inexplorés.

7.2.3.2.2 Approche hybride Une autre mise en œuvre possible consiste à utiliser la FMM pour calculer les champs lointains dans les directions (ξ_k) souhaitées par l'utilisateur, et non pas uniquement dans les directions imposées (θ_i,ϕ_j) par la méthode multipôle. On se place à un niveau d. Pour une fonction de radiation $\vec{\mathcal{F}_{\mathcal{C}^{(d)}}}$ donnée, le passage de la grille (θ_i,ϕ_j) au vecteur (ξ_k) se fera à l'aide d'une des méthodes de montée décrites aux sections 1.3.1.6 et 2.3. Il suffira ensuite de sommer les contributions de chaque cellule $\mathcal{C}^{(d)}$ translatée en O pour obtenir $\vec{A}(\vec{\xi})$:

$$\vec{A}(\vec{\xi}) = i\omega\mu_0 \sum_{\mathcal{C}^{(d)}} e^{ikM^{(d)}O} \vec{\mathcal{F}_{\mathcal{C}^{(d)}}}(\vec{\xi})$$

Il suffirait alors de choisir le niveau d pour minimiser le temps d'exécution. Par rapport à la première approche, les avantages et les inconvénients sont globalement inversés. La mise au point serait un peu plus compliquée puisqu'elle nécessiterait de créer un nouveau type d'opération pour cette FMM. Pour ce qui est de la vitesse de calcul, il est difficile de prévoir laquelle des deux approches sera la plus rapide. Cela dépendra du nombre de directions ξ, de la forme de l'objet, du niveau plafond, du niveau d choisi, et sans doute d'autres paramètres encore.

7.3 Matériaux diélectriques

7.3.1 Modèle physique considéré

On s'intéresse maintenant au cas d'un objet Ω_i régulier de frontière Γ composé d'un matériau diélectrique isotrope non-absorbant caractérisé par sa permittivité électrique ϵ_i et sa perméabilité magnétique μ_i. ϵ_i et μ_i sont réels, le cas complexe sera évoqué à la section 7.4.4. La vitesse de propagation des ondes dans ce milieu est $c_i = 1/\sqrt{\epsilon_i\mu_i}$, leur longueur d'onde est $\lambda_i = c_i/f$ (f étant bien sûr la fréquence), leur nombre d'onde est $k_i = 2\pi/\lambda_i$, le noyau de Green associé est $G_i(r) = e^{ik_ir}/4\pi r$. On note Ω_e le complémentaire de Ω_i, et ϵ_e, μ_e, c_e, λ_e, k_e, G_e les grandeurs qui s'y rattachent. On note ν la normale sortante sur Γ. Ω est illuminé par une onde incidente $(\vec{E}_{inc},\vec{H}_{inc})$.

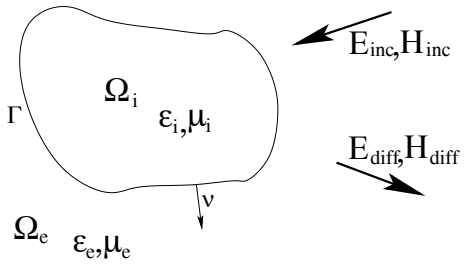


FIG. 7.2 – *Cas de matériaux diélectriques*

7.3.2 Modélisation mathématique

7.3.2.1 Mise en équations

Dans Ω_e, on prend pour inconnues les champs diffractés. Les équations sont :

$$(P_e)\ \begin{cases} \vec{rot}\vec{E_e} - i\omega\mu_e\vec{H_e} = 0, \\ \vec{rot}\vec{H_e} + i\omega\epsilon_e\vec{E_e} = 0, \\ \lim\limits_{r\to+\infty} r\left|\sqrt{\epsilon_e}\vec{E_e} - \sqrt{\mu_e}\vec{H_e}\wedge\dfrac{\vec{r}}{r}\right| = 0 \end{cases} \qquad dans\ \Omega_e \tag{7.9}$$

Dans Ω_i, les inconnues sont les champs totaux et les équations s'écrivent :

$$(P_i)\ \begin{cases} \vec{rot}\vec{E_i} - i\omega\mu_i\vec{H_i} = 0, \\ \vec{rot}\vec{H_i} + i\omega\epsilon_i\vec{E_i} = 0 \end{cases} \qquad dans\ \Omega_i \tag{7.10}$$

Pour avoir des conditions au bord, on écrit la continuité sur Γ des traces tangentielles des champs $\vec{E}$ et $\vec{H}$ totaux :

$$\begin{cases} \vec{E_i}\wedge\vec{\nu} = (\vec{E_e} + \vec{E_{inc}})\wedge\vec{\nu}, \\ \vec{H_i}\wedge\vec{\nu} = (\vec{H_e} + \vec{H_{inc}})\wedge\vec{\nu} \end{cases} \qquad sur\ \Gamma \tag{7.11}$$

On va tenter d'écrire une formulation intégrale afin de résoudre le problème (7.9)+(7.10)+(7.11). Nous allons d'abord présenter un résultat fondamental.

7.3.2.2 Théorème de représentation

Nous rappelons rapidement ici le théorème de représentation intégrale des solutions des équations de Maxwell. Un exposé complet peut être trouvé dans [29]. Dans un premier temps, on se place dans un cas général, indépendant du problème diélectrique présenté auparavant. Nous verrons par la suite comment faire le rapprochement.

On se donne Ω un ouvert borné régulier, de frontière Γ. On note ν la normale sortante à Ω. ϵ et μ sont les caractéristiques électromagnétiques du milieu, que l'on supposent être les mêmes à l'intérieur et à l'extérieur de Ω. Soulignons que ce problème est *fictif*: il n'a rien à voir pour l'instant avec le problème physique représenté sur le figure 7.2. La preuve, les coefficients ϵ et μ sont ici les mêmes dans tout l'espace, et la frontière Γ sépare deux domaines composé du même matériau (!!).

On note $(\vec{E},\vec{H})$ les solutions du problème (7.12) posé dans Ω et dans son complémentaire $\complement\Omega$.

$$\begin{array}{ll} dans\ \Omega & \left\{\begin{array}{l} \vec{rot}\vec{E}_i - i\omega\mu\vec{H}_i = 0, \\ \vec{rot}\vec{H}_i + i\omega\epsilon\vec{E}_i = 0 \end{array}\right. \\ dans\ \complement\Omega & \left\{\begin{array}{l} \vec{rot}\vec{E}_e - i\omega\mu\vec{H}_e = 0, \\ \vec{rot}\vec{H}_e + i\omega\epsilon\vec{E}_e = 0, \\ \lim\limits_{r\to+\infty} r\left|\sqrt{\epsilon}\vec{E}_e - \sqrt{\mu}\vec{H}_e \wedge \dfrac{\vec{r}}{r}\right| = 0 \end{array}\right. \end{array} \tag{7.12}$$

On note $\vec{j}$ et $\vec{m}$ les sauts des traces tangentielles des champs $(\vec{E},\vec{H})$ solutions de ce système.

$$\left\{\begin{array}{l} \vec{j} = \vec{H}_i \wedge \vec{\nu} - \vec{H}_e \wedge \vec{\nu}, \\ \vec{m} = \vec{E}_i \wedge \vec{\nu} - \vec{E}_e \wedge \vec{\nu} \end{array}\right. \qquad sur\ \Gamma \tag{7.13}$$

Alors les champs $\vec{E}(y)$ et $\vec{H}(y)$ pour tout $y \notin \Gamma$ se représentent sous la forme :

$$\left\{\begin{array}{l} \vec{E}(y) = i\omega\mu \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx + \frac{i}{\omega\epsilon}\vec{grad}_y \int_\Gamma G(|y-x|)div_\Gamma\vec{j}(x)dx, \\ \qquad\qquad +\vec{rot}_y \displaystyle\int_\Gamma G(|y-x|)\vec{m}(x)dx \\ \vec{H}(y) = -i\omega\epsilon \displaystyle\int_\Gamma G(|y-x|)\vec{m}(x)dx - \frac{i}{\omega\mu}\vec{grad}_y \int_\Gamma G(|y-x|)div_\Gamma\vec{m}(x)dx, \\ \qquad\qquad +\vec{rot}_y \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx \end{array}\right.$$

Ce sont ces formules qui servent à calculer les champs proches et lointains à partir des courants surfaciques. Pour $y \in \Gamma$, on a par ailleurs les formules suivantes qui donnent les valeurs intérieures (indicées $-$) et extérieures (indicées $+$) des traces tangentielles $\vec{E} \wedge \vec{\nu}$ et $\vec{H} \wedge \vec{\nu}$:

$$\left\{\begin{array}{l} (\vec{E} \wedge \vec{\nu})_\pm(y) = \mp\dfrac{\vec{m}(y)}{2} + \displaystyle\int_\Gamma \vec{grad}_y G(|y-x|) \wedge \vec{m}(x)dx \wedge \vec{\nu}(y) \\ \qquad\qquad +i\omega\mu \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x)dx \wedge \vec{\nu}(y) \\ \qquad\qquad +\dfrac{i}{\omega\epsilon}\displaystyle\int_\Gamma \vec{grad}_y G(|y-x|)div_\Gamma\vec{j}(x)dx \wedge \vec{\nu}(y), \\ (\vec{H} \wedge \vec{\nu})_\pm(y) = \mp\dfrac{\vec{j}(y)}{2} + \displaystyle\int_\Gamma \vec{grad}_y G(|y-x|) \wedge \vec{j}(x)dx \wedge \vec{\nu}(y) \\ \qquad\qquad -i\omega\epsilon \displaystyle\int_\Gamma G(|y-x|)\vec{m}(x)dx \wedge \vec{\nu}(y) \\ \qquad\qquad -\dfrac{i}{\omega\mu}\displaystyle\int_\Gamma \vec{grad}_y G(|y-x|)div_\Gamma\vec{m}(x)dx \wedge \vec{\nu}(y) \end{array}\right. \tag{7.14}$$

Pour alléger un peu les notations, on définit les deux opérateurs suivants :

$$\left\{\begin{array}{l} \vec{R}j(y) = \displaystyle\int_\Gamma \vec{grad}_y G(|y-x|) \wedge \vec{j}(x)dx, \\ \vec{S}j(y) = \displaystyle\int_\Gamma G(|y-x|)\vec{j}(x) + \frac{1}{k^2}\vec{grad}_y G(|y-x|)div_\Gamma\vec{j}(x)dx \end{array}\right. \tag{7.15}$$

R et S s'appliquent bien sûr à $\vec{j}$ comme à $\vec{m}$. Les traces tangentielles (7.14) s'écrivent alors (la dépendance de tous les termes en $y \in \Gamma$ est implicite) :

$$\left\{\begin{array}{l}(\vec{E}\wedge\vec{\nu})_{\pm}=\mp\dfrac{\vec{m}}{2}+R\vec{m}\wedge\vec{\nu}+i\omega\mu S\vec{j}\wedge\vec{\nu},\\(\vec{H}\wedge\vec{\nu})_{\pm}=\mp\dfrac{\vec{j}}{2}+R\vec{j}\wedge\vec{\nu}-i\omega\epsilon S\vec{m}\wedge\vec{\nu}\end{array}\right. \tag{7.16}$$

7.3.2.3 Formulation intégrale

Pour obtenir une équation sur les courants $\vec{j}$ et $\vec{m}$, nous allons appliquer deux fois le théorème de représentation.

7.3.2.3.1 Dans le domaine extérieur Dans Ω_e, on considère le problème (P_e) (équation (7.9)) sur les champs diffractés $\vec{E}_{diff}$ et $\vec{H}_{diff}$. On complète dans Ω_i avec le système :

$$\left\{\begin{array}{l}\vec{rot}\vec{E}_i-i\omega\mu_e\vec{H}_i=0,\\\vec{rot}\vec{H}_i+i\omega\epsilon_e\vec{E}_i=0\end{array}\right.\qquad dans\ \Omega_i$$

qui utilise ϵ_e, μ_e, tout comme (P_e). On prend pour solution l'opposé du champ incident $(-\vec{E}_{inc},-\vec{H}_{inc})$ (Cette donnée incidente définie dans Ω_e vérifie forcément ces équations de Maxwell). La définition (7.13) devient :

$$\left\{\begin{array}{l}\vec{j}=(-\vec{H}_{inc})\wedge\vec{\nu}-\vec{H}_{diff}\wedge\vec{\nu}=\vec{\nu}\wedge\vec{H}_{tot},\\\vec{m}=(-\vec{E}_{inc})\wedge\vec{\nu}-\vec{E}_{diff}\wedge\vec{\nu}=\vec{\nu}\wedge\vec{E}_{tot}\end{array}\right. \tag{7.17}$$

On a alors le droit d'écrire la représentation (7.16) des traces tangentielles dans Ω_e (avec les paramètres ϵ_e, μ_e, le noyau G_e, et les opérateurs R_e et S_e).

7.3.2.3.2 Dans le domaine intérieur Dans Ω_i, on considère le problème (P_i) (équation (7.10)) sur les champs totaux $\vec{E}_{tot}$ et $\vec{H}_{tot}$. On complète dans Ω_e avec le système :

$$\left\{\begin{array}{l}\vec{rot}\vec{E}_e-i\omega\mu_i\vec{H}_e=0,\\\vec{rot}\vec{H}_e+i\omega\epsilon_i\vec{E}_e=0,\\\lim\limits_{r\to+\infty}r\left|\sqrt{\epsilon_i}\vec{E}_e-\sqrt{\mu_i}\vec{H}_e\wedge\dfrac{\vec{r}}{r}\right|=0\end{array}\right.\qquad dans\ \Omega_e$$

qui utilise ϵ_i, μ_i, tout comme (P_i). On prend pour solution les champs nuls, solution triviale. La définition (7.13) devient :

$$\left\{\begin{array}{l}\vec{j}=(\vec{H}_{tot})\wedge\vec{\nu}-\vec{0}\wedge\vec{\nu}=-\vec{\nu}\wedge\vec{H}_{tot},\\\vec{m}=(\vec{E}_{tot})\wedge\vec{\nu}-\vec{0}\wedge\vec{\nu}=-\vec{\nu}\wedge\vec{E}_{tot}\end{array}\right.$$

On voit que les courants changent de signes par rapport au cas extérieur. On a alors le droit d'écrire la représentation (7.16) des traces tangentielles dans Ω_i (avec les paramètres ϵ_i, μ_i, le noyau G_i, et les opérateurs R_i et S_i).

7.3.2.3.3 Dans les deux domaines On note $\vec{j}$ et $\vec{m}$ les courants surfaciques électriques et magnétiques définis de la manière suivante :

$$\left\{ \begin{array}{l} \vec{j} = \vec{\nu} \wedge \vec{H}_{tot}, \\ \vec{m} = \vec{\nu} \wedge \vec{E}_{tot} \end{array} \right. \qquad sur\ \Gamma \tag{7.18}$$

Cette définition est légitime, car les traces tangentielles des champs totaux sont continues à la traversée de Γ (cf. équation (7.11)). Il n'y a donc pas d'ambiguïté sur le côté par lequel on approche Γ. Par rapport aux deux sections précédentes, on voit que l'on a adopté la définition de $(\vec{j},\vec{m})$ du problème extérieur. On devra donc écrire la représentation des traces tangentielles dans Ω_i avec $-\vec{j}$ et $-\vec{m}$.

Précisément, dans le domaine extérieur, on a :

$$\left\{ \begin{array}{l} (\vec{E}_e \wedge \vec{\nu}) = -\dfrac{\vec{m}}{2} + \vec{R_e m} \wedge \vec{\nu} + i\omega\mu_e \vec{S_e j} \wedge \vec{\nu}, \\ (\vec{H}_e \wedge \vec{\nu}) = -\dfrac{\vec{j}}{2} + \vec{R_e j} \wedge \vec{\nu} - i\omega\epsilon_e \vec{S_e m} \wedge \vec{\nu} \end{array} \right. \tag{7.19}$$

Dans le domaine intérieur, compte tenu du changement de signe souligné (intégré aux membres de gauche), on a :

$$\left\{ \begin{array}{l} -(\vec{E}_i \wedge \vec{\nu}) = +\dfrac{\vec{m}}{2} + \vec{R_i m} \wedge \vec{\nu} + i\omega\mu_i \vec{S_i j} \wedge \vec{\nu}, \\ -(\vec{H}_i \wedge \vec{\nu}) = +\dfrac{\vec{j}}{2} + \vec{R_i j} \wedge \vec{\nu} - i\omega\epsilon_i \vec{S_i m} \wedge \vec{\nu} \end{array} \right.$$

Ces deux derniers systèmes sont tous les deux définis sur Γ, on peut donc les sommer. En utilisant la condition de continuité (7.11), on obtient :

$$\left\{ \begin{array}{l} -(\vec{E}_{inc} \wedge \vec{\nu}) = (\vec{R_e m} + \vec{R_i m}) \wedge \vec{\nu} + i\omega(\mu_e \vec{S_e j} + \mu_i \vec{S_i j}) \wedge \vec{\nu}, \\ -(\vec{H}_{inc} \wedge \vec{\nu}) = (\vec{R_e j} + \vec{R_i j}) \wedge \vec{\nu} - i\omega(\epsilon_e \vec{S_e m} + \epsilon_i \vec{S_i m}) \wedge \vec{\nu} \end{array} \right.$$

En testant cela sur des fonctions tangentes $(\vec{j}^t,\vec{m}^t)$, on peut ôter le $\wedge\vec{\nu}$. On obtient la formulation intégrale : Trouver $(\vec{j},\vec{m})$ tels $\forall\vec{j}^t$ et $\forall\vec{m}^t$, on ait :

$$\left\{ \begin{array}{l} -\displaystyle\int_\Gamma \vec{E}_{inc}.\vec{j}^t = \int_\Gamma (\vec{R_e m} + \vec{R_i m}).\vec{j}^t + i\omega \int_\Gamma (\mu_e \vec{S_e j} + \mu_i \vec{S_i j}).\vec{j}^t, \\ -\displaystyle\int_\Gamma \vec{H}_{inc}.\vec{m}^t = \int_\Gamma (\vec{R_e j} + \vec{R_i j}).\vec{m}^t - i\omega \int_\Gamma (\epsilon_e \vec{S_e m} + \epsilon_i \vec{S_i m}).\vec{m}^t \end{array} \right.$$

Pour homogénéiser le système, on utilise des courants magnétiques normalisés notés :

$$\vec{p} = \frac{\vec{m}}{Z_0} \quad \text{avec} \quad Z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}}$$

$$\left\{ \begin{array}{l} \dfrac{1}{Z_0}\displaystyle\int_\Gamma \vec{E}_{inc}.\vec{j}^t = -\int_\Gamma (\vec{R_e p} + \vec{R_i p}).\vec{j}^t - \frac{i\omega}{Z_0}\int_\Gamma (\mu_e \vec{S_e j} + \mu_i \vec{S_i j}).\vec{j}^t, \\ \displaystyle\int_\Gamma \vec{H}_{inc}.\vec{p}^t = -\int_\Gamma (\vec{R_e j} + \vec{R_i j}).\vec{p}^t + i\omega Z_0 \int_\Gamma (\epsilon_e \vec{S_e p} + \epsilon_i \vec{S_i p}).\vec{p}^t \end{array} \right. \tag{7.20}$$

Sous cette forme, les deux équations sont homogènes : les courants $\vec{j}$ et $\vec{p}$ s'expriment en $A.m^{-1}$. Elles expriment le principe de réaction de Rumsey [35] : La réaction du champ total ($\vec{E}_{tot}$,$\vec{H}_{tot}$) sur un quelconque courant test ($\vec{j}^t$,$\vec{m}^t$) est nulle. Cela s'écrit :

$$\int_\Gamma \left(\vec{E}_{tot}(x).\vec{j}^t(x) + \vec{H}_{tot}(x).\vec{m}^t(x)\right) dx = 0$$

7.3.2.3.4 Cas parfaitement conducteur Avant de passer à la résolution numérique de ces équations, nous allons revenir rapidement sur le cas d'un matériau parfaitement conducteur et montrer comment ce qui précède permet d'aboutir aux formulations EFIE et MFIE de la section 1.1.2.2.

A l'intérieur d'un objet Ω parfaitement conducteur, les champs totaux $\vec{E}_{tot}$ et $\vec{H}_{tot}$ sont nuls. Sur la surface Γ de cet objet, seule la trace tangentielle du champ $\vec{E}_{tot}$ est continue (et nulle). Avec les définitions (7.17) pour $\vec{j}$ et $\vec{m}$, on trouve :

$$\begin{cases} \vec{m} = 0, \\ \vec{E}_e \wedge \vec{\nu} = -\vec{E}_{inc} \wedge \vec{\nu} \end{cases}$$

Les équations (7.19) deviennent :

$$\begin{cases} -(\vec{E}_{inc} \wedge \vec{\nu}) = i\omega\mu_e \vec{S_e j} \wedge \vec{\nu}, \\ -(\vec{H}_{inc} \wedge \vec{\nu}) = -\dfrac{\vec{j}}{2} + \vec{R_e j} \wedge \vec{\nu} \end{cases}$$

De ces deux équations, on déduit très simplement les formulations EFIE et MFIE, mais une seule suffit à résoudre le problème.

7.3.3 Résolution numérique

7.3.3.1 Discrétisation

On utilise les mêmes éléments finis pour les courants $\vec{j}$ et $\vec{p}$, à savoir ceux présentés dans le cas parfaitement conducteur 1.1.3. On note $\vec{\varphi}_k$ les fonctions de base. On note λ_k les flux de courant électrique $\vec{j}$ à travers les arêtes du maillages, et ν_k les flux de courant magnétique $\vec{p}$ à travers ces mêmes arêtes. On a les décompositions suivantes :

$$\begin{cases} \vec{j}(x) = \displaystyle\sum_{1\leq k\leq n_{dl}} \lambda_k \vec{\varphi}_k(x), \\ \vec{p}(x) = \displaystyle\sum_{1\leq k\leq n_{dl}} \nu_k \vec{\varphi}_k(x) \end{cases} \tag{7.21}$$

La mise sous forme matricielle est obtenue en insérant (7.21) dans (7.20), et en prenant pour fonction test chacune des fonctions de base $\vec{\varphi}_j$. La matrice obtenue est symétrique. La résolution itérative de ce type de système conduit à la réalisation de produit matrice vecteur. Nous allons voir comment les réaliser de manière rapide grâce à la FMM.

7.3.3.2 Algorithme multipôle

7.3.3.2.1 Adaptation à la FMM Etant donné des courants surfaciques $\vec{j}$ et $\vec{p}$, la réalisation d'un produit matrice vecteur revient à calculer les deux quantités suivantes $\forall \vec{\varphi}_k$:

$$\left\{ \begin{array}{l} -\displaystyle\int_\Gamma (\vec{R_e p} + \vec{R_i p}).\vec{\varphi}_k - \frac{i\omega}{Z_0}\int_\Gamma (\mu_e \vec{S_e j} + \mu_i \vec{S_i j}).\vec{\varphi}_k, \\ -\displaystyle\int_\Gamma (\vec{R_e j} + \vec{R_i j}).\vec{\varphi}_k + i\omega Z_0 \int_\Gamma (\epsilon_e \vec{S_e p} + \epsilon_i \vec{S_i p}).\vec{\varphi}_k \end{array} \right. \tag{7.22}$$

La présence de deux milieux différents fait apparaître toutes les grandeurs « en double » : deux nombres d'onde k_e et k_i, deux noyaux de Green G_e et G_i, etc. La FMM étant intrinsèquement liée à ces données, la méthode la plus naturelle et la plus simple pour calculer (7.22) est de réaliser deux produits multipôles, un dans chaque domaine. Par exemple, dans le domaine extérieur, on calcule $\forall \vec{\varphi}_k$:

$$\left\{ \begin{array}{l} -\displaystyle\int_\Gamma (\vec{R_e p}).\vec{\varphi}_k - \frac{i\omega}{Z_0}\int_\Gamma (\mu_e \vec{S_e j}).\vec{\varphi}_k, \\ -\displaystyle\int_\Gamma (\vec{R_e j}).\vec{\varphi}_k + i\omega Z_0 \int_\Gamma (\epsilon_e \vec{S_e p}).\vec{\varphi}_k \end{array} \right. \tag{7.23}$$

Techniquement, cela implique de manipuler deux produits FMM s'appuyant sur le même maillage (de Γ). Ceci dit, avec des valeurs de k, G, c et λ différentes, ces deux FMM n'auront pas la même taille de feuille, pas le même nombre de niveaux dans l'octree, pas la même matrice d'interactions proches, pas les mêmes matrices de transfert et de translation, etc. Autrement dit, même en reposant sur le même maillage, ces deux FMM seront totalement différentes.

Pour la suite de cette section, nous allons supprimer les indices « $_e$ » et « $_i$ » afin de simplifier les formules. Il faut bien sûr garder à l'esprit que la plupart des constantes utilisées dépendent désormais du milieu considéré.

7.3.3.2.2 Nouvelles formulations Reprenons les expressions (7.15) des opérateurs R et S. On souhaite donc calculer les deux termes suivants :

$$\left\{ \begin{array}{l} -\displaystyle\int_{\Gamma\times\Gamma} (\vec{grad}_y G(|y-x|) \wedge \vec{p}(x)).\vec{\varphi}_k(y)dxdy \\ \qquad -\displaystyle\frac{i\omega\mu}{Z_0}\int_{\Gamma\times\Gamma} G(|y-x|)\left[\vec{j}(x).\vec{\varphi}_k(y) - \frac{1}{k^2} div_\Gamma \vec{j}(x) div_\Gamma \vec{\varphi}_k(y)\right] dxdy, \\ -\displaystyle\int_{\Gamma\times\Gamma} (\vec{grad}_y G(|y-x|) \wedge \vec{j}(x)).\vec{\varphi}_k(y)dxdy \\ \qquad +i\omega\epsilon Z_0 \displaystyle\int_{\Gamma\times\Gamma} G(|y-x|)\left[\vec{p}(x).\vec{\varphi}_k(y) - \frac{1}{k^2} div_\Gamma \vec{p}(x) div_\Gamma \vec{\varphi}_k(y)\right] dxdy \end{array} \right.$$

Sans surprise, on retrouve des choses déjà vues lors de l'écriture de l'algorithme multipôle en EFIE et en MFIE (cf. 1.2.2). On se place donc dans le contexte habituel : on crée un octree autour de Γ dans lequel on distribue toutes les inconnues (électriques et magnétiques). Les interactions entre feuilles proches sont traitées classiquement. On cherche à calculer le terme d'interaction entre deux feuilles lointaines $\mathcal{C}$ et $\mathcal{C}'$, centrées en M et M'. Soient deux points $x \in \mathcal{C}$ et $y \in \mathcal{C}'$. On décompose le noyau de Green avec (1.4) :

$$G(|y-x|) = \frac{ik}{16\pi^2} \lim_{L\to+\infty} \int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}} T^L_{M\vec{M}'}(\vec{s}) e^{ik\vec{s}.\vec{M'}y} d\vec{s}$$

En pratique, le nombre de pôles L est fixé en fonction de la précision voulue et de la taille des cellules au niveau de l'octree concerné, le terme « $\lim_{L\to+\infty}$ » peut donc être ôté.

Le terme $\int_{\Gamma\cap\mathcal{C}\times\Gamma\cap\mathcal{C}'} G(|y-x|)\left[\vec{j}.\vec{\varphi}_k - 1/k^2 div_\Gamma\vec{j} div_\Gamma\vec{\varphi}_k\right]$ a déjà été rencontré dans le cas EFIE, on a la formulation FMM à trois composantes suivantes :

Initialisation : on calcule la fonction de radiation

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{j}(x) - (\vec{s}.\vec{j}(x))\vec{s})dx \tag{7.24}$$

Intégration : pour toute fonction de base $\vec{\varphi}_k$ localisée dans $\mathcal{C}'$, on intègre :

$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s}).\vec{\varphi}_k(y)d\vec{s}dy$$

Les montées, transferts, et descentes restent inchangés. On sait qu'il existe également une formulation multipôle à deux composantes pour ce calcul.

Regardons maintenant l'autre terme en $\vec{j}$: $\int_{\Gamma\cap\mathcal{C}\times\Gamma\cap\mathcal{C}'}(\vec{grad}_y G(|y-x|)\wedge\vec{j}(x)).\vec{\varphi}_k(y)dxdy$. On peut le réécrire sous la forme :

$$\int_{\Gamma\cap\mathcal{C}\times\Gamma\cap\mathcal{C}'}(\vec{\varphi}_k(y)\wedge\vec{grad}_y G(|y-x|)).\vec{j}(x))dxdy \tag{7.25}$$

On a alors la décomposition suivante pour le gradient de G :

$$\vec{grad}_y G(|y-x|) = \frac{ik}{16\pi^2}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.x\vec{M}} T^L_{M\vec{M}'}(\vec{s})\ ik\vec{s}\ e^{ik\vec{s}.\vec{M'}y}d\vec{s} \tag{7.26}$$

En insérant (7.26) dans (7.25), on obtient la formulation FMM suivante :

Initialisation : on calcule la fonction de radiation

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\vec{j}(x)dx \tag{7.27}$$

Intégration : pour toute fonction de base $\vec{\varphi}_k$ localisée dans $\mathcal{C}'$, on intègre :

$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y} ik\vec{\mathcal{G}}_{\mathcal{C}'}(\vec{s}).(\vec{\varphi}_k(y)\wedge\vec{s})d\vec{s}dy$$

La présence du « $\wedge\vec{s}$ » implique que la composante des fonctions de radiation dirigée selon $\vec{s}$ ne nous intéresse pas (comme dans le cas de la MFIE, cf. 1.2.2.1.3). On peut donc remplacer l'étape d'initialisation (7.27) par (7.24). Enfin, on a écrit les équations pour les termes en $\vec{j}$, il faut faire de même pour les termes en $\vec{p}$. On obtient alors l'algorithme suivant :

Initialisation : on calcule les deux fonctions de radiation :

$$\left\{\begin{array}{l} \vec{\mathcal{F}}^j_{\mathcal{C}}(\vec{s}) = \displaystyle\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{j}(x) - (\vec{s}.\vec{j}(x))\vec{s})dx, \\ \vec{\mathcal{F}}^p_{\mathcal{C}}(\vec{s}) = \displaystyle\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}(\vec{p}(x) - (\vec{s}.\vec{p}(x))\vec{s})dx \end{array}\right. \tag{7.28}$$

Intégration : pour toute fonction de base $\vec{\varphi}_k$ localisée dans $\mathcal{C}'$, on intègre :

$$\left\{\begin{array}{l} \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\left[-ik\vec{\mathcal{G}^p_{\mathcal{C}'}}(\vec{s}).(\vec{\varphi}_k(y)\wedge\vec{s})-\dfrac{i\omega\mu}{Z_0}\vec{\mathcal{G}^j_{\mathcal{C}'}}(\vec{s}).\vec{\varphi}_k(y)\right]d\vec{s}dy, \\ \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}\left[-ik\vec{\mathcal{G}^j_{\mathcal{C}'}}(\vec{s}).(\vec{\varphi}_k(y)\wedge\vec{s})+i\omega\epsilon Z_0\vec{\mathcal{G}^p_{\mathcal{C}'}}(\vec{s}).\vec{\varphi}_k(y)\right]d\vec{s}dy \end{array}\right.$$

Dans la première de ces deux intégrales, $\vec{\varphi}_k$ agit en tant que fonction test $\vec{j}^t$, et en tant que $\vec{p}^t$ dans la seconde. On obtient ainsi une formulation multipôle à 6 termes, qui peut être transformée en formulation à quatre termes grâce à la méthode de la section 1.2.2.1.2, qui s'applique ici aux fonctions de radiation j et p séparément. Il convient juste, dans ce cas, de penser à ajuster les opérations de montée/descente en conséquence (cf. section 1.3.1.6.4).

Remarquons que les deux équations de l'étape d'intégration peuvent s'écrire :

$$\left\{\begin{array}{l} \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}(-\dfrac{i\omega\mu}{Z_0})\left[\vec{\mathcal{G}^j_{\mathcal{C}'}}(\vec{s})+\dfrac{Z_0}{c\mu}\vec{s}\wedge\vec{\mathcal{G}^p_{\mathcal{C}'}}(\vec{s})\right].\vec{\varphi}_k(y)d\vec{s}dy, \\ \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}(-ik)\left[\vec{\mathcal{G}^j_{\mathcal{C}'}}(\vec{s})+\dfrac{\omega\epsilon Z_0}{k}\vec{s}\wedge\vec{\mathcal{G}^p_{\mathcal{C}'}}(\vec{s})\right].(\vec{\varphi}_k(y)\wedge\vec{s})d\vec{s}dy \end{array}\right.$$

Le terme entre crochets est le même dans ces deux équations. On peut donc faire le produit multipôle complet avec une seule fonction de radiation définie à partir de 7.28 par :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})=\vec{\mathcal{F}^j_{\mathcal{C}}}(\vec{s})+\frac{Z_0}{c\mu}\vec{s}\wedge\vec{\mathcal{F}^p_{\mathcal{C}}}(\vec{s})$$

On obtient ainsi un algorithme multipôle à seulement deux termes dont les étapes initiales et finales s'écrivent :

Initialisation : on calcule la fonction de radiation :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s})=\int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\left[(\vec{j}(x)-(\vec{s}.\vec{j}(x))\vec{s})+\frac{Z_0}{c\mu}\vec{s}\wedge\vec{p}(x)\right]dx$$

Le « $\vec{p}(x)-(\vec{s}.\vec{p}(x))\vec{s}$ » est redevenu un « $\vec{p}(x)$ » car le produit vectoriel par $\vec{s}$ enlève automatiquement la composante de $\vec{p}(x)$ colinéaire à $\vec{s}$.

Intégration : pour toute fonction de base $\vec{\varphi}_k$ localisée dans $\mathcal{C}'$, on intègre :

$$\left\{\begin{array}{l} \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}(-\dfrac{i\omega\mu}{Z_0})\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}).\vec{\varphi}_k(y)d\vec{s}dy, \\ \dfrac{ik}{16\pi^2}\displaystyle\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'}y}(-ik)\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s}).(\vec{\varphi}_k(y)\wedge\vec{s})d\vec{s}dy \end{array}\right.$$

7.4 Objets hétérogènes

On va maintenant étudier le cas d'objets hétérogènes, composés de plusieurs matériaux diélectriques ou parfaitement conducteurs. Nous allons voir comment ce genre de problème peut être géré en formulation classique puis multipôle. Il s'agit donc de généraliser ce qui vient d'être vu dans le cas de deux milieux (un objet Ω dans le vide).

7.4.1 Cas traité

L'obstacle Ω, de forme quelconque, est composé de plusieurs domaines volumiques connexes Ω_k constitués chacun d'un matériau homogène soit diélectrique (de caractéristiques ϵ_k et μ_k) soit parfaitement conducteur (noté p.c.). On note D le nombre de sous-domaines, l'indice k variant de 1 à D. Deux sous-domaines disjoints peuvent être composés du même matériau. Cet objet est placé dans un milieu lui-même diélectrique Ω_0 , dont on notera ϵ_0 et μ_0 les constantes. Dans chaque domaine (hors parfaitement conducteur), on définit la célérité des ondes $c_k = 1/\sqrt{\epsilon_k \mu_k}$, la longueur d'onde $\lambda_k = c_k/f$, le nombre d'onde $k_k = 2\pi/\lambda_k$, le noyau de Green $G_k(r) = e^{ik_k r}/4\pi r$. Toutes les surfaces de discontinuité du matériau sont maillées.

Cette obstacle est soumis à une excitation de fréquence f qui peut être une onde incidente (plane ou sphérique), une source de tension, ou toute combinaison linéaire de celles-ci.

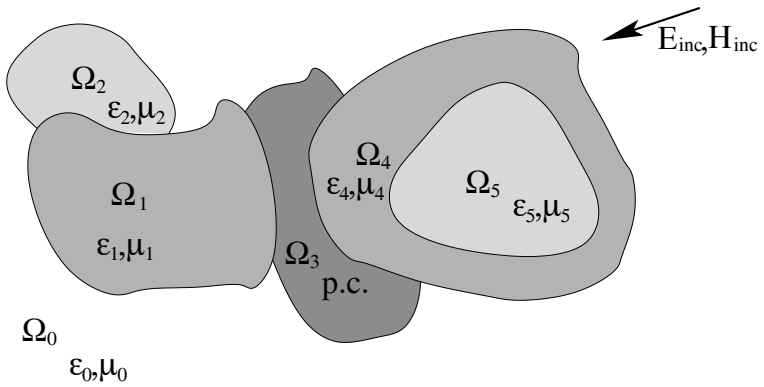


FIG. 7.3 – *Cas de plusieurs matériaux*

7.4.2 Mise en équations

On va appliquer le théorème de représentation une fois pour chaque domaine diélectrique, puis sommer toutes les contributions ainsi écrites.

7.4.2.1 Dans un domaine

Dans le domaine Ω_k diélectrique, le problème (réel) que l'on cherche à résoudre s'écrit :

$$(P_k) \quad \begin{cases} \vec{rot}\vec{E}_{diff} - i\omega\mu_k\vec{H}_{diff} = 0, \\ \vec{rot}\vec{H}_{diff} + i\omega\epsilon_k\vec{E}_{diff} = 0, \\ \lim\limits_{r\to+\infty} r\left|\sqrt{\epsilon_k}\vec{E}_{diff} - \sqrt{\mu_k}\vec{H}_{diff} \wedge \dfrac{\vec{r}}{r}\right| = 0 \end{cases} \qquad dans\ \Omega_k$$

La condition à l'infini, dite condition de radiation de Silver-Muller, n'a bien sûr de sens que si Ω_k est non borné (dans le cas contraire, cette condition s'appliquera au problème (P'_k) qui suit). Les inconnues sont les champs diffractés par les courants $\vec{j}$ et $\vec{m}$ portés par la frontière fermée de ce domaine $\partial\Omega_k$. Le champ total $(\vec{E}_{tot},\vec{H}_{tot})$ à l'intérieur de Ω_k est la somme du champ incident $(\vec{E}_{inc},\vec{H}_{inc})$ dû aux sources éventuelles d'excitation contenues dans ce domaine, et du champ diffracté $(\vec{E}_{diff},\vec{H}_{diff})$

Pour appliquer le théorème de représentation dans Ω_k, on complète ce problème (P_k) par un problème fictif (P'_k) posé dans le complémentaire de Ω_k :

$$(P'_k) \quad \begin{cases} \vec{rot}\vec{E} - i\omega\mu_k\vec{H} = 0, \\ \vec{rot}\vec{H} + i\omega\epsilon_k\vec{E} = 0, \\ \lim\limits_{r\to+\infty} r\left|\sqrt{\epsilon_k}\vec{E} - \sqrt{\mu_k}\vec{H}\wedge\dfrac{\vec{r}}{r}\right| = 0 \end{cases} \qquad dans\ \complement\Omega_k$$

Pour solution de (P'_k), on prend l'opposé des champs incidents $(-\vec{E}_{inc}, -\vec{H}_{inc})$ (dûs aux sources d'excitation contenues dans Ω_k). On note ν la normale sortante, Ω_k sera alors considéré comme étant le domaine intérieur. Les sauts des traces tangentielles s'écrivent :

$$\begin{cases} \vec{j} = (\vec{H}_{diff})\wedge\vec{\nu} - (-\vec{H}_{inc})\wedge\vec{\nu} = \vec{H}_{tot}\wedge\vec{\nu}, \\ \vec{m} = (\vec{E}_{diff})\wedge\vec{\nu} - (-\vec{E}_{inc})\wedge\vec{\nu} = \vec{E}_{tot}\wedge\vec{\nu} \end{cases} \tag{7.29}$$

On utilise alors l'expression (7.16) pour obtenir :

$$\begin{cases} (\vec{E}_{diff}\wedge\vec{\nu})_k = +\dfrac{\vec{m}}{2} + \vec{R_k m}\wedge\vec{\nu} + i\omega\mu_k\vec{S_k j}\wedge\vec{\nu}, \\ (\vec{H}_{diff}\wedge\vec{\nu})_k = +\dfrac{\vec{j}}{2} + \vec{R_k j}\wedge\vec{\nu} - i\omega\epsilon_k\vec{S_k m}\wedge\vec{\nu} \end{cases} \tag{7.30}$$

Dans un domaine parfaitement conducteur, tous les champs sont nuls, et rien de tout cela ne s'applique.

7.4.2.2 A l'interface entre deux domaines

On se place en un point y situé sur la frontière entre deux domaines Ω_k et Ω_j tous deux diélectriques. On suppose la normale ν orientée de Ω_k vers Ω_j (cf. figure 7.4). Dans Ω_k, la normale ν est sortante, la formule (7.30) peut s'écrire telle quelle. Dans Ω_j par contre, la normale sortante est $-\nu$, les courants sont donc $-\vec{j}$ et $-\vec{m}$, et (7.30) devient :

$$\begin{cases} -(\vec{E}_{diff}\wedge\vec{\nu})_j = -\dfrac{\vec{m}}{2} + \vec{R_j m}\wedge\vec{\nu} + i\omega\mu_j\vec{S_j j}\wedge\vec{\nu}, \\ -(\vec{H}_{diff}\wedge\vec{\nu})_j = -\dfrac{\vec{j}}{2} + \vec{R_j j}\wedge\vec{\nu} - i\omega\epsilon_j\vec{S_j m}\wedge\vec{\nu} \end{cases} \tag{7.31}$$

On somme (7.30) et (7.31). Les traces tangentielles des champs totaux étant continues, le saut des champs diffractés est l'opposé du saut des champs incidents. On obtient le système :

$$\begin{cases} (\vec{E}_{inc}\wedge\vec{\nu})_j - (\vec{E}_{inc}\wedge\vec{\nu})_k = \left(\vec{R_j m} + \vec{R_k m} + i\omega\mu_j\vec{S_j j} + i\omega\mu_k\vec{S_k j}\right)\wedge\vec{\nu}, \\ (\vec{H}_{inc}\wedge\vec{\nu})_j - (\vec{H}_{inc}\wedge\vec{\nu})_k = \left(\vec{R_j j} + \vec{R_k j} - i\omega\epsilon_j\vec{S_j m} - i\omega\epsilon_k\vec{S_k m}\right)\wedge\vec{\nu} \end{cases}$$

Cette équation est valable en tout point $y \in \partial\Omega_k \cap \partial\Omega_j$. On note $\Gamma_{kj} = \partial\Omega_k \cap \partial\Omega_j$. Si on teste par des fonctions tangentes $(\vec{j}^t, \vec{m}^t)$, on peut supprimer le produit vectoriel par ν. On obtient :

$$\begin{cases} \displaystyle\int_{y\in\Gamma_{kj}} \left(\vec{R_j m} + \vec{R_k m} + i\omega\mu_j\vec{S_j j} + i\omega\mu_k\vec{S_k j}\right).\vec{j}^t dy = \int_{y\in\Gamma_{kj}} \left((\vec{E}_{inc})_j - (\vec{E}_{inc})_k\right).\vec{j}^t dy, \\ \displaystyle\int_{y\in\Gamma_{kj}} \left(\vec{R_j j} + \vec{R_k j} - i\omega\epsilon_j\vec{S_j m} - i\omega\epsilon_k\vec{S_k m}\right).\vec{m}^t dy = \int_{y\in\Gamma_{kj}} \left((\vec{H}_{inc})_j - (\vec{H}_{inc})_k\right).\vec{m}^t dy \end{cases} \tag{7.32}$$

On a choisi de noter $(\vec{E}_{inc})_j$ le champ créé par les sources d'excitation contenues dans Ω_j. Ainsi, les membres de droite sont des données du problème. Il ne faut pas perdre de vue que les opérateurs S_j et R_j font intervenir des intégrales surfaciques sur $\partial\Omega_j$ tout entier. Les membres de gauche font donc intervenir les courants $\vec{j}$ et $\vec{m}$ sur tout $\partial\Omega_k \cup \partial\Omega_j$.

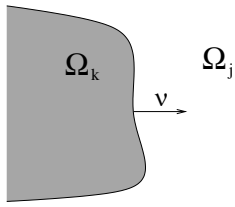


FIG. 7.4 – *Frontière entre Ω_k et Ω_j*

Terminons par une remarque sur le rôle de l'orientation de la normale. Dans le membre de droite, le saut $(\vec{E}_{inc})_j - (\vec{E}_{inc})_k$ est calculé dans le sens « extérieur moins intérieur » c'est-à-dire dans le sens opposé à la normale sortante ν (cf. figure 7.4). Si on change le sens de la normale, on change le signe du second membre. En résolvant notre système, on trouvera donc des courants $\vec{j}$ et $\vec{m}$ opposés, ce qui avec (7.29) nous redonnera les mêmes traces tangentielles pour les champs totaux. Le choix de l'orientation de la normale est donc bien libre.

Si l'un des deux domaines est parfaitement conducteur, disons Ω_k, alors $\vec{m}$ est nul sur Γ_{kj}. On n'a plus besoin de $\vec{m}^t$, et (7.32) se réduit à une seule équation :

$$\int_{y\in\Gamma_{kj}} i\omega\mu_j \vec{S_j j}.\vec{j}^t dy = \int_{y\in\Gamma_{kj}} (\vec{E}_{inc})_j.\vec{j}^t dy$$

7.4.2.3 Sur l'ensemble des frontières

On va sommer les équations (7.32) sur l'ensemble des frontières des domaines Ω_k diélectriques avec $0 \leq k \leq D$. On introduit les courants magnétiques normalisés $\vec{p} = \vec{m}/Z_0$. On obtient :

$$\left\{\begin{array}{l} \displaystyle\sum_{0\leq k\leq D} \int_{y\in\partial\Omega_k} \left(\vec{R_k p} + \frac{i\omega\mu_k}{Z_0}\vec{S_k j}\right).\vec{j}^t dy = \frac{1}{Z_0}\sum_{0\leq k\leq D}\int_{y\in\partial\Omega_k}\left(-\nu.\nu_k(\vec{E}_{inc})_k\right).\vec{j}^t dy, \\ \displaystyle\sum_{0\leq k\leq D} \int_{y\in\partial\Omega_k} \left(\vec{R_k j} - i\omega\epsilon_k Z_0 \vec{S_k p}\right).\vec{p}^t dy = \sum_{0\leq k\leq D}\int_{y\in\partial\Omega_k}\left(-\nu.\nu_k(\vec{H}_{inc})_k\right).\vec{p}^t dy \end{array}\right. \tag{7.33}$$

On note ν_k la normale sortante à Ω_k, et ν la normale définie globalement en tout point des surfaces considérées. Le coefficient $\nu.\nu_k$ vaut donc $+1$ si ν sort de Ω_k, et -1 sinon. Rappelons que $(\vec{E}_{inc})_k$ représente le champ incident créé par les sources d'excitation contenues dans Ω_k. Dans le cas d'ondes planes incidentes, elles sont censées être contenues dans le domaine extérieur infini Ω_0.

Dans (7.33), la somme ne se fait que sur les domaines diélectriques. La présence éventuelle de domaines parfaitement conducteurs est prise en compte en interdisant la présence de fonction $\vec{p}$ et $\vec{p}^t$ sur leurs frontières.

7.4.3 Aspects numériques

7.4.3.1 Assemblage de la matrice

Comme on le voit dans l'équation (7.33), seuls les fonctions de base $\vec{j}$ et $\vec{j}^t$ se trouvant sur les frontières d'un milieu Ω_k commun vont pouvoir interagir. Prenons l'exemple de la figure 7.5 : L'objet est composé de quatre milieux Ω_0 à Ω_3, séparés par des interfaces baptisées Γ_a à Γ_f.

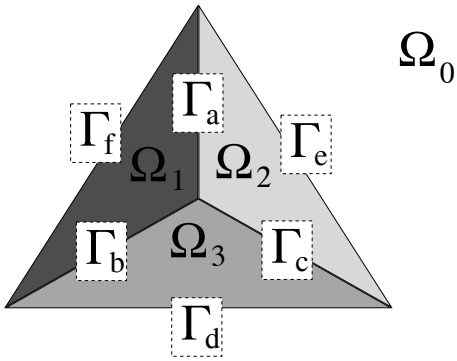


FIG. 7.5 – *Exemple composé de 4 matériaux*

Les interfaces Γ_a et Γ_b interagiront via le milieu Ω_1, Γ_c agira sur elle-même via les milieux Ω_2 et Ω_3, Γ_a et Γ_d n'interagiront pas (du moins, pas directement). Dans le tableau 7.1, on donne pour chaque couple d'interface le ou les milieux susceptibles de les faire interagir.

	Γ_a	Γ_b	Γ_c	Γ_d	Γ_e	Γ_f
Γ_a	1,2	1	2		2	1
Γ_b	1	1, 3	3	3		1
Γ_c	2	3	2,3	3	2	
Γ_d		3	3	0,3	0	0
Γ_e	2		2	0	0,2	0
Γ_f	1	1		0	0	0,1

TAB. 7.1 – *Matrice d'interaction entre les interfaces*

On voit que cette matrice n'est pas pleine, car il existe des degrés de liberté qui ne vont pas interagir directement. Elle est bien évidemment symétrique. Si un des milieux était parfaitement conducteur, il ne serait le cadre d'aucune interaction, et son indice serait tout simplement supprimé du tableau.

En pratique, le calcul se fait de la manière suivante : pour tout couple de triangles T et T' du maillage, il faut pouvoir retrouver les numéros des milieux séparés par chacun de deux. Pour chaque milieu commun trouvé, on calcule la matrice élémentaire d'interaction entre ces deux triangles dans ce milieu. Cette matrice est de taille 6×6 en général, et seulement 3×3 si un des deux triangles borde un milieu parfaitement conducteur. La somme de toutes ces matrices élémentaires produit la matrice entière du système linéaire à résoudre.

7.4.3.2 Formulation multipôle

Dans le cas d'une résolution itérative avec FMM, on procédera ici comme on l'a fait à la section 7.3.3.2. Chaque milieu (hors conducteur parfait) donnera lieu à un produit multipôle visant à calculer les deux termes suivants :

$$\begin{cases} \displaystyle\int_{y\in\partial\Omega_k} \left(\vec{R_k}p + \frac{i\omega\mu_k}{Z_0}\vec{S_k}j\right).\vec{j}^t dy, \\ \displaystyle\int_{y\in\partial\Omega_k} \left(\vec{R_k}j - i\omega\epsilon_k Z_0 \vec{S_k}p\right).\vec{p}^t dy \end{cases}$$

On retrouve, au signe près, exactement la même chose que dans l'équation (7.23). La différence de signe provient des choix opposés faits pour définir $\vec{j}$ et $\vec{m}$ (équation (7.18) et (7.29) respectivement). Ainsi, le cas de la figure 7.5 sera traité avec quatre FMM (trois s'il y a un conducteur parfait), et celui de la figure 7.3 en 6 produits multipôles (moins le nombre de domaines parfaitement conducteurs). Il est important de souligner que si deux domaines disjoints sont composés du même matériau, on ne peut pour autant pas faire une FMM commune au deux. Ce n'est pas le concept de matériau qui est déterminant, mais le concept de milieu diélectrique homogène connexe. Par exemple, sur la figure 7.3, si les domaines Ω_1 et Ω_4 sont de même composition, ça ne change rien à la manière de traiter le problème en mode FMM.

7.4.4 Matériaux absorbants

Terminons cette étude des matériaux diélectriques par le cas des matériaux absorbants, c'est-à-dire pour lesquels le nombre d'onde k a une partie imaginaire non-nulle. Lorsque les constantes électromagnétiques ϵ et μ sont complexes, le nombre d'onde $k = \omega\sqrt{\epsilon\mu}$ s'écrit alors $k = k_r + ik_i$ avec $k_i > 0$ (seul cas physiquement acceptable). Le noyau de Green devient :

$$G_k(r) = \frac{e^{ikr}}{4\pi r} = e^{-k_i r}\frac{e^{ik_r r}}{4\pi r}$$

La partie réelle de k détermine la composante ondulatoire de la fonction de Green, tandis que sa partie imaginaire traduit le caractère absorbant du matériaux. Ce cas est étudié dans [8], nous nous contentons de souligner les principaux résultats. Les formules fondamentales de la FMM restent valides en théorie. En pratique, la présence de termes de la forme $e^{|k_i r|}$ les rend inutilisables dès que $|k_i r|$ grandit. On est amené à fixer un seuil s tel que au-delà d'une distance $r = s/k_i$ toutes les interactions sont négligées. Ainsi on est assuré que $|k_i r|$ restera borné. En pratique, s est de l'ordre de 3. Dans l'algorithme multipôle, aucun transfert n'aura lieu pour des distances supérieures à s/k_i, ce qui conduit à plafonner le calcul à un certain niveau donné par s (tout en continuant à ne faire les transferts à ce niveau qu'entre cellules banlieues, contrairement au cas habituel du niveau plafond où l'on fait les transferts entre toutes les cellules non-voisines). Bien sûr, cette nouveauté méritera d'être étudiée plus en détail lors de son implémentation.

Les principales difficultés liées à la mise en œuvre de la FMM dans le cas de matériaux absorbants ne viennent pas de l'algorithme lui-même, qui varie assez peu, mais de la nécessité de manipuler des nombres complexes à la place de réels à certains endroits clés (nombre d'onde, ϵ, μ, fonctions de Hankel h_l et de Bessel j_l) qui requerront des réécritures de code parfois fastidieuses.

7.5 Structures filaires minces

7.5.1 Approximation des fils minces

Nous nous intéressons dans cette section aux structures filaires minces dans le cas d'un objet Ω métallique parfaitement conducteur placé dans le vide, de paramètres ϵ_0 et μ_0. Les courants électriques $\vec{j}$ sont répartis sur une couche infiniment mince à la surface extérieure de Ω. Ils sont solutions de l'équation EFIE (1.1).

Pour les structures filaires, l'adjectif « mince » signifie que la section des fils est très petite par rapport à leur longueur et par rapport à la longueur d'onde de l'onde incidente. On suppose en outre que la section des fils est circulaire, et que le rayon de courbure de l'axe du fil est grand devant la longueur d'onde. Dans ces conditions, le courant $\vec{j}$ peut être considéré comme parallèle à l'axe du fil, en négligeant sa composante située dans un plan perpendiculaire à cet axe. De plus, $\vec{j}$ ne dépend que de l'abscisse curviligne sur le fil, et est constant sur toute section droite du fil.

On note r_f le rayon du fil, l l'abscisse curviligne sur le fil, $\vec{\tau}(l)$ la tangente à l'axe du fil et $I(l)$ l'intensité du courant au point d'abscisse l. On a :

$$\vec{j}(l) = \frac{I(l)\vec{\tau}(l)}{2\pi r_f}$$

Dans la formulation EFIE, on souhaite transformer l'intégrale sur la surface du fil en une intégrale linéique sur l'axe du fil. Le but de l'approximation des fils minces est donc d'ôter l'intégration sur le périmètre de la section du fil.

7.5.2 Discrétisation

On utilise l'élément de Raviart-Thomas dans sa restriction 2D. Le fil est découpé en segments qui constituent le maillage. On note M_i les sommets de ce maillage. Les fonctions de base sont les fonctions « chapeaux » de l'abscisse curviligne affines par morceaux, valant 1 en un point du maillage et 0 sur les autres. On note φ_i ces fonctions. Le support d'une telle fonction est constitué d'au plus deux segments consécutifs. L'inconnue est l'intensité du courant en chaque point du maillage. On note I_i ces degrés de liberté. Comme les inconnues λ_i sur les triangles, les I_i sont homogènes à des ampères. En tout point d'abscisse l sur le fil, l'intensité et le courant de surface s'écrivent :

$$\begin{cases} I(l) = \displaystyle\sum_i I_i \varphi_i(l), \\ \vec{j}(l) = \dfrac{1}{2\pi r_f} \displaystyle\sum_i I_i \varphi_i(l) \vec{\tau}(l) \end{cases}$$

Nous ne rentrerons pas dans les détails de l'implémentation de cette méthode. Cela nous entraînerait trop loin dans des développements numériques et analytiques. Soulignons simplement que par rapport au cas purement surfacique (évoqué à la section 1.1.4.1), il faut ici en plus savoir traiter les interactions fil-triangle et fil-fil, en distinguant les interactions proches (traitées avec des formules analytiques) et lointaines (traitées plus simplement avec des intégrations par points de Gauss). Pour de plus amples informations, voir par exemple [31] et [36].

7.5.3 Formulation multipôle

Reprenons la formulation multipôle à deux composantes de la section 1.2.2.1.2. Pour toute feuille $\mathcal{C}$ de centre M, l'étape d'initialisation consiste à calculer :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{x\in\Gamma\cap\mathcal{C}} e^{ik\vec{s}.x\vec{M}}\vec{j}(x)dx \tag{7.34}$$

On passe sous silence les phases de montée, transfert et descente qui resteront inchangées (elles ne dépendent que du noyau) ainsi que le passage de trois à deux composantes et son inverse. Pour toute feuille $\mathcal{C}'$ de centre M', l'intégration consiste à calculer le terme :

$$\frac{ik}{16\pi^2}\int_{y\in\Gamma\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} e^{ik\vec{s}.\vec{M'y}}\vec{\varphi_j}(y).\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s})d\vec{s}dy \tag{7.35}$$

Nous allons regarder comment l'approximation des fils minces modifie ces équations.

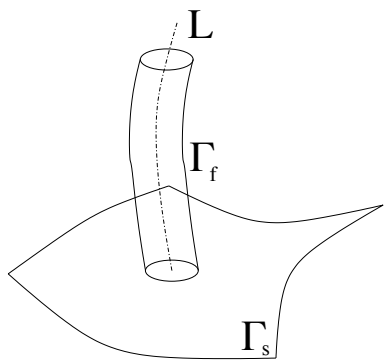


FIG. 7.6 – *Cas d'une structure comportant un fil*

Pour ce faire, conformément à la figure 7.6, on note Γ_f la surface des structures filaires, et Γ_s le reste de Γ, c'est-à-dire la partie véritablement surfacique. On note $\mathcal{L}$ l'axe des fils. Le but de l'approximation des fils minces est de passer d'une intégration sur $\Gamma = \Gamma_f \cup \Gamma_s$ à une intégration simplifiée sur $\mathcal{L}\cup\Gamma_s$. Dans les deux formules (7.34) et (7.35), on va mettre de côté l'intégration sur $\Gamma_s \cap \mathcal{C}$ – qui va rester inchangée – pour ne conserver que l'intégrale sur la surface du fil $\Gamma_f \cap \mathcal{C}$.

On note l l'abscisse curviligne sur $\mathcal{L}$, et $\mathcal{S}(l)$ le périmètre de la section droite du fil à l'abscisse l (cf. figure 7.7). Cette section est circulaire (par hypothèse), de centre $O(l)$, de rayon $r_f(l)$ (constant le long d'un fil, mais pouvant varier d'un fil à l'autre). Le courant $\vec{j}$ sur le fil ne dépend que de l. L'initialisation devient alors :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{l\in\mathcal{L}\cap\mathcal{C}}\int_{x\in\mathcal{S}(l)} e^{ik\vec{s}.x\vec{M}}\vec{j}(l)dldx$$

En décomposant $x\vec{M} = x\vec{O}(l) + \vec{O(l)M}$, on obtient :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{l\in\mathcal{L}\cap\mathcal{C}}\left[\int_{x\in\mathcal{S}(l)} e^{ik\vec{s}.x\vec{O}(l)}dx\right] e^{ik\vec{s}.\vec{O(l)M}}\vec{j}(l)dl \tag{7.36}$$

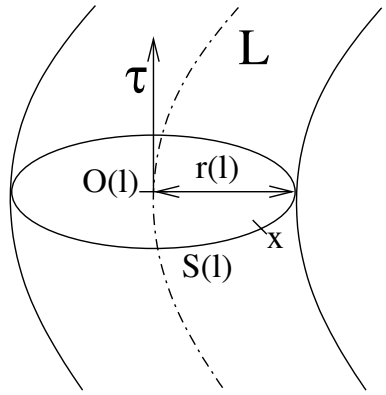


FIG. 7.7 – *Section d'un fil*

Pour calculer l'intégrale entre crochet, on se place dans le plan du disque $\mathcal{S}(l)$. Dans ce plan, on prend un repère d'origine $O(l)$. Le point x a pour coordonnées $(r_f(l)\cos\theta, r_f(l)\sin\theta, 0)$ avec $\theta \in [0,2\pi]$. On note (s_x, s_y, s_z) les coordonnées de $\vec{s}$ dans ce nouveau repère. On fait un développement limité à l'ordre 4 en $kr_f(l)$ de cette intégrale (pour alléger les écritures, on va provisoirement ôter les dépendances des termes en l) :

$$\begin{aligned}\int_{x\in\mathcal{S}(l)} e^{ik\vec{s}.\vec{xO}}dx &= \int_{\theta=0}^{2\pi} e^{-ikr_f(s_x\cos\theta+s_y\sin\theta)} r_f d\theta \\ &= \int_{\theta=0}^{2\pi}\left[\sum_{n=0}^{4}\frac{(-ikr_f(s_x\cos\theta+s_y\sin\theta))^n}{n!} + o(k^4r_f^4)\right] r_f d\theta\end{aligned} \tag{7.37}$$

L'intégrale et le signe somme pouvant permuter, on voit apparaître les quantités suivantes :

$$\begin{cases}\displaystyle\int_{\theta=0}^{2\pi}(s_x\cos\theta+s_y\sin\theta)^n d\theta = 0 \qquad \forall n \text{ impair}, \\ \displaystyle\int_{\theta=0}^{2\pi}(s_x\cos\theta+s_y\sin\theta)^0 d\theta = 2\pi, \\ \displaystyle\int_{\theta=0}^{2\pi}(s_x\cos\theta+s_y\sin\theta)^2 d\theta = \pi(s_x^2+s_y^2), \\ \displaystyle\int_{\theta=0}^{2\pi}(s_x\cos\theta+s_y\sin\theta)^4 d\theta = \frac{3\pi}{4}(s_x^2+s_y^2)^2, \\ \displaystyle\int_{\theta=0}^{2\pi}(s_x\cos\theta+s_y\sin\theta)^6 d\theta = \frac{5\pi}{8}(s_x^2+s_y^2)^3,\end{cases}$$

(7.37) devient :

$$\int_{x\in\mathcal{S}(l)} e^{ik\vec{s}.\vec{xO}}dx = 2\pi r_f - \frac{k^2r_f^3}{2}\pi(s_x^2+s_y^2) + \frac{k^4r_f^5}{32}\pi(s_x^2+s_y^2)^2 + r_f o(k^4r_f^4) \tag{7.38}$$

On ne connaît pas s_x et s_y, en revanche on sait que $s_z = \pm\vec{s}.\vec{\tau}$ (où $\vec{\tau}$ est le vecteur tangent à l'axe du fil, donc normal à sa section). Le vecteur $\vec{s}$ étant unitaire, on a $s_x^2+s_y^2 = 1-s_z^2 = 1-(\vec{s}.\vec{\tau})^2$. On utilise (7.38) et l'équation $\vec{j}(l) = I(l)\vec{\tau}(l)/2\pi r_f$ dans (7.36) :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{l\in\mathcal{L}\cap\mathcal{C}} \left[1 - \frac{k^2 r_f^2}{4}(1-(\vec{s}.\vec{\tau})^2) + \frac{k^4 r_f^4}{64}(1-(\vec{s}.\vec{\tau})^2)^2 + o(k^4 r_f^4)\right] e^{ik\vec{s}.O(\vec{l})M} I(l)\vec{\tau}(l)dl$$

Avec $k = 2\pi/\lambda$ et $r_f < \lambda/10$ (par hypothèse), on trouve $kr_f < 2\pi/10 \approx 0{,}63$, et on déduit $k^2 r_f^2/4 < 0{,}10$ et $k^4 r_f^4/64 < 2{,}4.10^{-3}$. Le terme suivant du développement limité s'écrira $k^6 r_f^6/2304$ et sera inférieur à $2{,}7.10^{-5}$. En prenant seulement deux termes, on a une erreur inférieure à 1 %, en prenant trois termes elle est inférieur a 10^{-4} ce qui sera toujours suffisant dans le cadre d'une FMM. Remarquons que les trois premiers termes se factorisent :

$$\left[1 - \frac{k^2 r_f^2}{4}(1-(\vec{s}.\vec{\tau})^2) + \frac{k^4 r_f^4}{64}(1-(\vec{s}.\vec{\tau})^2)^2\right] = \left[1 - \frac{k^2 r_f^2}{8}(1-(\vec{s}.\vec{\tau})^2)\right]^2$$

Dans ce cas, l'étape d'initialisation pour un fil s'écrit :

$$\vec{\mathcal{F}}_{\mathcal{C}}(\vec{s}) = \int_{l\in\mathcal{L}\cap\mathcal{C}} \left[1 - \frac{k^2 r_f(l)^2}{8}(1-(\vec{s}.\vec{\tau}(l))^2)\right]^2 e^{ik\vec{s}.O(\vec{l})M} I(l)\vec{\tau}(l)dl$$

De la même manière, l'intégration d'une fonction de radiation $\vec{\mathcal{G}_{\mathcal{C}'}}$ sur une portion filaire $\Gamma_f \cap \mathcal{C}$ s'écrit :

$$\frac{ik}{16\pi^2}\int_{l\in\mathcal{L}\cap\mathcal{C}'}\int_{\vec{s}\in\mathcal{S}} \left[1 - \frac{k^2 r_f(l)^2}{8}(1-(\vec{s}.\vec{\tau}(l))^2)\right]^2 e^{ik\vec{s}.M'\vec{O}(l)}\vec{\varphi_j}(l).\vec{\mathcal{G}_{\mathcal{C}'}}(\vec{s})d\vec{s}dl$$

On voit que l'approximation des fils minces s'intègre sans difficulté dans la méthode multipôle rapide. Il y a juste un terme dépendant de $r_f(l)$, $\vec{s}$ et $\vec{\tau}(l)$ à rajouter pour les intégrales sur les fils $\mathcal{L}$.

Conclusion

Nous avons présenté dans ce chapitre différentes applications de la méthode multipôle rapide. Nous avons pu traiter différents types de calculs (des post-traitements, des résolutions itératives), appliqués à des problèmes physiques plus ou moins élaborés (plusieurs milieux, matériaux multiples, fils, antennes), mais basés à chaque fois sur le même noyau de Green $G(r) = e^{ikr}/4\pi r$. Les formulations multipôles présentées ici n'ont pas encore été mises en œuvre en pratique, mais elles le seront prochainement en fonction des besoins manifestés par notre partenaire industriel.

Au demeurant, il existe une méthode simple pour ne pas être bridé par les seules possibilités de la FMM. Il suffit de savoir séparer au démarrage du calcul les interactions qui pourront être traitées via une FMM, et les autres (qui seront alors traitées classiquement). Si ces interactions hors-FMM sont peu nombreuses, leur matrice sera creuse. C'est ce que nous avons fait pour les éléments filaires, et cela nous permet de traiter rapidement des cas de calcul de grande taille comportant quelques antennes.

Conclusion

Nous avons réalisé l'implémentation de la méthode multipôle rapide pour les équations de Maxwell et de Helmholtz dans les codes d'éléments finis de frontière d'EADS. Les résultats obtenus se sont révélés très satisfaisants, tant en terme de précision que de rapidité. Sur machine séquentielle, nous pouvons désormais traiter de manière anodine des problèmes comportant jusqu'à 10^5 inconnues, voire 10^6 sur des machines haut de gamme. Nous avons parallélisé ce code, et obtenu sur machines parallèles des résultats très probants en terme de vitesse et d'efficacité parallèle, établissant au passage de nouveaux records en terme de nombre d'inconnues pour ce type de problème.

Pour autant, toutes les difficultés n'ont pas été balayées par la FMM. La méthode multipôle hérite des inconvénients associés aux solveurs itératifs et aux formulations intégrales surfaciques. Pour les premiers, le principal problème est de ne pas maîtriser la vitesse de convergence. De nombreuses voies sont explorées pour tenter d'y remédier, parmi lesquelles les solveurs multi seconds membres, les préconditionneurs flexibles ([22]) et ceux basés sur les projecteurs de Calderon ([11]). Quant aux formulations intégrales de frontières, le principal reproche à leur faire est de ne pas s'appliquer aux objets complexes, comme ceux comportant des matériaux hétérogènes. La solution ici serait de réaliser un couplage entre la FMM et des méthodes fréquentielles volumiques plus à même de gérer ces difficultés.

La méthode multipôle peut également s'appliquer dans d'autres contextes comme la chimie moléculaire, la biochimie, l'électrostatique. La partie « mathématique » de la FMM doit bien sûr être adaptée au cas par cas (définition des fonctions de radiation et des opérateurs de montée, descente, transfert, ...), mais la partie « algorithmique » de la méthode (création et exploration de l'arbre, création de la liste de tâches) reste inchangée. En particulier, les aspects out-of-core et parallélisation sont conservés. Nous projetons donc de nous associer à des équipes travaillant dans les domaines évoqués ci-dessus afin d'y exploiter au maximum le potentiel de nos développements. Enfin, dans la foulée des travaux de Chew à l'université de l'Illinois, Michielssen a développé une méthode rapide inspirée de la FMM adaptée aux méthodes intégrales temporelles pour l'électromagnétisme ([18], [38]). Adapter cette FMM temporelle à une méthode de potentiels retardés à l'efficacité avérée ([46]) est également un axe de recherche prometteur.

Bibliographie

[1] M. Abramowitz and I. A. Stegun : *Handbook of mathematical functions*, Dover publications, New York, 1972.

[2] G. Alléon, M. Benzi, L. Giraud : *Sparse Approximate Inverse Preconditionning for dense linear systems arising in computational electromagnetics* Numerical Algorithms, vol. 16, pp. 1-15, 1997

[3] A. Bendali : *Approximation par éléments finis de surface de problèmes de diffraction des ondes électromagnétiques*, Thèse de doctorat de l'université Paris 6, 1984.

[4] A. Bendali : *Numerical Analysis of the exterior boundary value problem for the time-harmonic Maxwell equations by a boundary finite element method. I : the continuous problem*, Math. Comp., 43(167):29-46, 1984.

[5] A. Bendali : *Numerical Analysis of the exterior boundary value problem for the time-harmonic Maxwell equations by a boundary finite element method. II : the discrete problem*, Math. Comp., 43(167):47-68, 1984.

[6] J. J. Bowman, T. B. A. Senior, P. L. E. Uslenghi : *Electromagnetic and Acoustic Scattering by Simple Shapes*, 2nb edition, Summa Book, 1987.

[7] W. E. Boyse, A. A. Seidl : *A Block QMR method for computing multiple simultaneous solutions to complex symmetric systems*, SIAM J. Sci. Computing, 17:263-274, 1996.

[8] Q. Carayol : *Développement et analyse d'une méthode multipôle multi-niveau pour l'électromagnétisme*, Thèse de doctorat de l'université Paris 6, Janvier 2002.

[9] B. Carpentieri : *Sparse preconditioners for dense linear systems in electromagnetic applications*, Thèse de l'Institut National Polytechnique de Toulouse, Avril 2002.

[10] V. Frayssé, L. Giraud, S. Gratton : *A Set of GMRES routines for real and complex arithmetics*, technical report TR/PA/97/49, CERFACS, 1997.

[11] S. H. Christiansen et J. C. Nedelec : *Des préconditionneurs pour la résolution numérique des équations intégrales de frontière de l'électromagnétisme*, C.R. Acad. des Sciences de Paris, septembre 2000.

[12] Ecoles des ondes Inria-Rocquencourt : *Aspects récents en méthodes numériques pour les équations de Maxwell*, Editeurs G. Cohen et P. Joly, collection didactique, mars 1998.

[13] Coifman, Rokhlin, Wandzura. *The Fast Multipole Method for the wave equation: A pedestrian description.* , IEEE Trans. on Ant. and Prop., 35(3):7-12, June 1993.

[14] F. Collino et F. Millot : *La méthode multipôle à deux composantes pour l'électromagnétisme*, rapport technique TR/EMC/01/22, CERFACS, 2001.

[15] E. Darve : *Méthodes multipôles rapides : Résolution des équations de Maxwell par formulations intégrales*, Thèse de doctorat de l'université Paris 6, Juin 1999.

[16] E. Darve : *The Fast Multipole Method 1 : Error Analysis and Asymptotic Complexity*, SIAM J. Numer. Anal, vol. 38, No. 1, pp. 98-128, June 2000.

[17] R. Dautray et J. L. Lions : *Analyse mathématique et calcul numérique pour les sciences et techniques*, volume2, Masson, 1985.

[18] A. A. Ergin, B. Shanker and E. Michielssen : *Fast analysis of transient acoustic wave scattering from rigid bodies using the multilevel plane wave time domain algorithm*, J. Acous. Soc. Am. 107(3), March 2000.

[19] R. W. Freund and N. M. Nachtigal : *QMR : A Quasi Minimal Residual Method for Non-Hermitian Linear Systems*, Numerische Mathematik, 60, 1991.

[20] R. W. Freund, M. H. Gutknecht and N. M. Nachtigal : *An implementation of the look-ahead Lanczos algorithm for the non-hermitian matrices*, technical report 91.09, RIACS, NASA Ames research center, Moffett Field, 1991.

[21] M. Frigo and S. G. Johnson : *The Fastest Fourier Transform in the West*, technical report, MIT-LCS-TR-728, sep. 1997.

[22] B. Carpentieri, I. Duff, L. Giraud, G. Sylvand : *Inner-outer Iterations: a Way to Design Robust Linear Solvers for Dense Linear Systems in Electromagnetism*, Latsis Symposium on Iterative Solvers for Large Linear Systems, 18-21 Février 2002, Zurich, Suisse.

[23] L. Greengard and V. Rokhlin : *A Fast Algorithm for Particle Simulations*, J. Comp. Phys., 73:325-348, 1987.

[24] R. H. Hardin and N. J. A. Sloane : *McLaren's Improved Snub Cube and Other New Spherical Designs in Threee Dimension*, Discrete and Comp. Geometry, vol 15, pp. 429-441, 1996.

[25] D. S. Jones : *Acoustic and Electromagnetic Waves*, Clarenton Press, Oxford, 1986.

[26] M. R. Hestenes and E. Stiefel : *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standard, vol. 49, no. 6, pp. 409-436, 1942.

[27] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov and D. C. Sorensen, *Lapack Users' Guide, second version*, Society for Industrial and Applied Mathematics, 1995.

[28] A. D. McLaren : *Optimal Numerical Integration on a Sphere*, Math. of Comp., vol 17, pp. 361-383, 1963.

[29] J. C. Nedelec : *Ondes acoustiques et électromagnétiques - Equations intégrales*, Cours de DEA de l'école polytechnique et de l'université Paris 6, 1996.

[30] W. H. Press [et al.] : *Numerical Recipes in C - The art of scientific computing, second edition*, Cambridge University Press, 1992.

[31] D. Poljak : *Finite Element Integral Equation Modelling of a Thin Wire Loop Antenna*, Communications in Numerical Methods in Engineering, vol. 14, no. 4, pp. 347-354, 1998.

[32] S. M. Rao, D. R. Wilton and A. W. Glisson : *Electromagnetic Scattering by Surfaces of Arbitrary Shape*, IEEE Transactions on Antennas and Propagations, 30(3), pp. 409-418, may 1982.

[33] P. A. Raviart, J. M. Thomas : *A mixed finite element method for 2nd order elliptic problems* in Mathematical aspects of finite element methods, Rome 1975, Eds I. Galligani, E. Magenes, Lectures Notes in Mathematics 606, Springer-Verlag, Berlin, 1975.

[34] V. Rokhlin : *Diagonal forms of translation operators for the Helmholtz equation in three dimension.* Technical report, YALEU/DCS/RR-894, Department of Computer Science, Yale University, 1992.

[35] V. H. Rumsey : *Reaction Concept in Electromagnetic Theory*, Physical Review, 94, pp 1438-1491, (1954).

[36] B. P. Rynne : *Convergence of Galerkin method solutions of the integral equation for thin wire antennas*, Advances in Computational Mathematics, vol. 12, no. 2-3, pp. 251-260, 2000.

[37] Y. Saad, M. Schultz : *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sc. Stat. Comput., 7, pp 856-869, 1986.

[38] B. Shanker, A. A. Ergin and E. Michielssen : *A multilevel plane wave time domain algorithm for the fast analysis of transient scattering phenomena*, Proceedings of IEEE antennas propagat. Soc. Symp., vol. 2, pp. 1342-1345, Orlando, FL, July 11-16, 1999.

[39] X. Q. Sheng, J. M. Jin, J. Song, W. C. Chew and C. C. Lu, *Solution of Combined Field Integral Equation Using Multilevel Fast Multipole Algorithm for Scattering by Homogeneous Bodies* IEEE Ant. Propag., vol. 46, no. 11, pp. 1718-1726, Nov. 1998.

[40] J. M. Song and W. C. Chew : *Fast Multipole Method Solution Using Parametric Geometry*, Micro. Opt. Tech. Lett., vol. 7, no. 16, pp.760-765, November 1994.

[41] J. M. Song and W. C. Chew : *Fast Multipole Method Solution of Combined Field Integral Equation*, Proc. 11th Annual Review of Progress in Applied Computational Electromagnetics, Naval Postgraduate School, Monterey, CA, pp. 629-636, March 20-25, 1995.

[42] J. M. Song, C. C. Lu and W. C. Chew : *Multilevel Fast Multipole Algorithm for Electromagnetic scattering.* IEE Antennas and Propagation, 45:1488-1493, 1997.

[43] J. M. Song, C. C. Lu, W. C. Chew and S. W. Lee : *Fast Illinois Solver Code*, IEEE Antennas and Propagation Magazine, Vol. 40, No. 3, June 1998.

[44] J. M. Song and W. C. Chew : *The Fast Illinois Solver Code : Requirements and Scaling Properties*, IEEE Computationnal Science and Engineering, 5(3):19-23, July-September 1998.

[45] J. A. Stratton : *Electromagnetic Theroy*, McGraw Hill, 1941.

[46] I. Terrasse : *Résolution mathématique et numérique des équations de Maxwell instationnaires par une méthode de potentiels retardés*, Thèse de doctorat de l'école polytechnique, 1993.

[47] L. N. Trefethen and D. Bau : *Numerical Linear Algebra*, SIAM, 1997.

[48] S. Velamparambil and W. C. Chew : *Parallelization of MLFMA on distributed memory computers*, in Roberto D. Graglia, editor, Proceedings of the International Conference on Electromagnetics in Advanced Applications (ICEAA01), pages 141-144, September 10-14 2001.