N° d'ordre : 4324

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Mathieu CHANAUD**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Conception d'un solveur haute performance de systèmes linéaires creux couplant des méthodes multigrilles et directes pour la résolution des équations de Maxwell 3D en régime harmonique discrétisées par éléments finis

---

Soutenue le : 18 Octobre 2011

Après avis des rapporteurs :

Daniel BOUCHE	Directeur de recherche CEA ..	CEA/DAM/DIF
Yousef SAAD	Professeur	University of Minnesota

Devant la commission d'examen composée de :

Daniel BOUCHE	Directeur de recherche CEA .	Examinateur
Luc GIRAUD	Directeur de recherche INRIA	Examinateur
David GOUDIN	Ingénieur-chercheur CEA	Examinateur
Stéphane LANTÉRI ..	Directeur de recherche INRIA	Examinateur
Jean Jacques PESQUÉ	Chercheur senior CEA	Examinateur
Jean ROMAN	Professeur IPB et INRIA	Directeur de thèse
Xavier VASSEUR	Chercheur senior CERFACS .	Examinateur

– 2011 –

Remerciements

Je souhaite tout d'abord remercier mon directeur de thèse Jean Roman ainsi que David Goudin qui m'ont encadré pendant ces 3 années et demi ainsi que Luc Giraud et Jean-Jacques Pesqué pour leurs précieux conseils. Je remercie aussi messieurs Daniel Bouche et Yousef Saad d'avoir accepté d'être rapporteurs sur ce manuscrit et de d'avoir consacré une partie de leur mois d'août à la relecture de celui-ci. Je remercie également les membres du jury non cités jusqu'alors Stéphane Lantéri et Xavier Vasseur de faire le déplacement jusqu'à Bordeaux pour juger mon travail lors de ma soutenance.

Je remercie tous les gens du CEA qui m'ont accepté et côtoyé durant ces quelques années avec une pensée particulière pour Sébastien et Thibault qui soutiendront leur thèse d'ici peu et pour les autres doctorants Cyril, François x2, Manuel et Pascal : n'oubliez pas que tout vient à point à qui sait attendre (le dernier moment). Je remercie aussi Xavier pour avoir répondu à mes diverses questions concernant PaStiX et pour les dernières semaines où il a été mes yeux et mains pour lancer les ultimes runs et dépouiller les résultats. Je lui souhaite bon courage et beaucoup de réussite pour la future thèse qu'il a acceptée d'entreprendre et ce en toute connaissance de cause.

Je remercie également toutes les personnes de l'INRIA pour l'intérêt porté à mon travail et pour m'avoir accueilli en début et fin de thèse.

Je remercie finalement ma famille et mes proches pour leur soutien quotidien, avec une mention spéciale décernée à Cyrielle qui a su me supporter pendant ces 3++ années et pour sa tendance à stresser à ma place : je promet de faire de mon mieux pour égaler cette performance pendant les trois années qui l'attendent ;)

Mathieu

Table des matières

Introduction générale

Le service simulation des phénomènes physiques (SSPP) du CEA/DAM/CESTA réalise la simulation d'objets soumis à des ondes électromagnétiques. À partir de la géométrie de l'objet et des matériaux le constituant, ce type d'étude permet d'en prévoir leur réponse radar. Ces simulations sont une composante essentielle du développement d'engins furtifs. Pour réaliser ces études, des codes de calcul spécifiques sont développés et employés par le CEA. La complexité des calculs à réaliser ainsi que la taille des problèmes considérés nécessitent l'utilisation de machines parallèles et les méthodes employées reposent sur des techniques de calcul hautes performances. Les phénomènes électromagnétiques se modélisent par le biais des équations de Maxwell [39]. Elles décrivent le comportement des champs électriques et magnétiques autour de l'objet d'étude. Du fait de la complexité de ces équations, la résolution analytique est impossible et nécessite alors une discrétisation de l'espace d'étude. La discrétisation dont nous discuterons dans cette thèse repose sur les éléments finis d'arêtes de Nédélec [40, 41]. L'objet est alors représenté par un maillage non structuré sur lequel est calculée la valeur des champs électromagnétiques. Une fois discrétisées, les équations de Maxwell sont représentées par une matrice creuse à coefficients complexes. Le calcul des champs électromagnétiques requiert alors la résolution d'un système linéaire creux. Diverses méthodes permettent de résoudre de tels systèmes linéaires ; parmi elles on peut compter les méthodes directes et itératives toutes deux couramment utilisées en simulation numérique. Dans le cadre des équations de Maxwell, le comportement des méthodes itératives n'est pas satisfaisant [45] et la majorité des résolutions se base sur les méthodes directes.

La taille des objets simulés impose l'utilisation de maillages plus ou moins volumineux. Les études actuellement menées au CEA concernent des objets 3D de taille importante requérant des maillages composés de millions de mailles. Les systèmes linéaires comptant des dizaines de millions d'inconnues sont fréquents et, dans un futur proche, il faudra envisager la résolution de systèmes de plusieurs milliards d'inconnues. La méthode directe utilisée pour les résolutions volumiques induit des coûts CPU et mémoire très importants. Pour les domaines de grande taille, le remplissage induit par la factorisation rend le stockage de ces données difficilement réalisable sur les machines actuelles d'autant plus que la tendance pour les machines futures est de réduire la quantité de mémoire par cœur. Une solution consiste à réduire la taille des systèmes en réalisant une décomposition de domaine où chaque sous-domaine est résolu de manière indépendante. Une seconde solution, consistant à utiliser plus de processeurs pour la résolution directe, arrive elle aussi à ses limites : les méthodes directes parallèles nécessitent un ratio calcul/communication suffisant pour être performantes. Plus on distribue le système, plus les communications augmentent, rendant les méthodes directes peu scalables sur des milliers de processeurs. Pour les systèmes linéaires de grande taille et pour les méthodes massivement parallèles l'utilisation des méthodes directes est donc remise en question. Avec

l'arrivée des machines massivement parallèles comportant plusieurs centaines de milliers de processeurs il est nécessaire de disposer de solveurs hautement parallèles. Malgré leur robustesse, leur prévisibilité et la qualité des solutions calculées, les solveurs directs ont du mal à passer à l'échelle lorsqu'il s'agit de milliers de cœurs de calcul. Le remplaçant de la méthode directe devra donc permettre l'utilisation d'un très grand nombre de cœurs.

En plus de la taille de l'objet étudié, la fréquence a elle aussi une grande influence sur la nature du maillage : pour une simulation précise et répondant aux besoins du CEA, un nombre suffisant de mailles par longueur d'onde (NMLO) est nécessaire [46]. Cette contrainte permet d'éviter tout phénomène d'*aliasing*. En électromagnétisme, il est d'usage d'utiliser au moins 7 mailles par longueur d'onde (10 pour les hautes fréquences). La longueur d'onde étant inversement proportionnelle à la fréquence, plus la fréquence est élevée, plus le maillage doit être fin. Dans le cadre des simulations menées au CEA, les objets sont testés sur de larges gammes de fréquences. Il est alors nécessaire de fournir un maillage suffisant pour chacune de ces fréquences. Fournir autant de maillages nécessite un travail conséquent de CAO en amont de la simulation (génération des maillages et prétraitements). Nous proposons alors une solution se basant un maillage *minimal* représentant correctement la géométrie de l'objet et l'emploi de raffinements automatiques. Le maillage utilisé lors des calculs et respectant la contrainte du NMLO sera généré automatiquement par raffinements successifs. Ce raffinement automatique permettra aux utilisateurs de calculer de larges gammes de fréquences en ne fournissant qu'un seul maillage qui sera le plus grossier. De ces constats est née l'idée de remplacer le solveur direct actuellement employé par une méthode multigrille géométrique.

Développée dans les années 70 par A.Brandt [7], la méthode multigrille repose sur plusieurs niveaux de maillages pour effectuer la résolution. À partir d'une solution calculée sur un niveau de maillage grossier (fourni par l'utilisateur) la méthode multigrille est capable de calculer une solution sur des maillages fins par méthodes itératives. L'utilisation de la méthode multigrille géométrique réduit alors à son strict minimum le besoin en CAO introduit plus tôt : le maillage "minimum" fourni par l'utilisateur est raffiné de manière automatique par le solveur multigrille ; le respect des contraintes en termes de nombre de mailles par longueur d'onde est alors délégué à la technique de raffinement (sous la condition que le maillage d'origine représente de manière précise l'objet d'étude).

En plus de cet avantage, les méthodes multigrilles sont réputées robustes et performantes. Alors que les méthodes itératives classiques nécessitent des préconditionneurs très performants pour résoudre les problèmes électromagnétiques [37, 42, 45], la méthode multigrille repose sur un solveur direct. L'utilisation de la solution "exacte" sur le niveau grossier permet alors de garantir une convergence rapide et la robustesse de la méthode pour un nombre de niveaux raisonnable (2 à 3 niveaux de raffinement). Finalement, la méthode multigrille est scalable. Alors que les préconditionneurs performants des méthode itératives classiques ne sont pas parallèles (ILU), la méthode multigrille passe à l'échelle facilement et peut s'adapter aux nouvelles architectures de machines. Par contre, le parallélisme de la méthode est contraint par la méthode directe car la quantité de calculs sur le niveau grossier doit être suffisante.

Du fait de ces complexités introduites précédemment et de l'arrivée des machines massivement parallèles, les méthodes itératives connaissent un regain d'attention pour la résolution des équations de Maxwell. Le but de cette thèse est de fournir au CEA un solveur de systèmes linéaires creux capable de passer à l'échelle et de calculer de façon satisfaisante la solution des systèmes linéaires. Les méthodes itératives de Krylov telles que GMRES [51] et le gra-

dient conjugué [29] ne sont pas satisfaisantes pour la résolution des équations de Maxwell ou nécessitent des préconditionneurs peu scalables. Ce comportement est principalement dû au fait que les opérateurs de Maxwell et de Helmholtz ne sont pas définis positifs, leur *near null space* est de grande taille et les coefficients complexes ajoutent à la difficulté. L'utilisation d'un solveur direct dans la méthode multigrille permet de réduire les effets négatifs de ces propriétés.

Cette thèse se divise en trois parties. La première d'entre elles décrit le contexte général de l'étude et fait un état de l'art des méthodes et outils employés. La seconde partie détaille les choix de conception et la réalisation du solveur multigrille géométrique. Finalement, nous prouvons dans la troisième partie la validité et l'utilité de la méthode au point de vue numérique et l'étude de ses performances parallèles.

Dans le chapitre 1, les méthode multigrilles sont présentées dans le cadre général. Du fait de l'utilisation combinée des solveurs directs et itératifs, une brève introduction à ces techniques de résolution est faite et, par un exemple simple, nous introduisons les bases de la méthode multigrille [7]. Nous complétons ensuite cette description de manière plus rigoureuse par l'expression algébrique de la méthode. Une fois la méthode multigrille introduite, nous nous intéressons aux équations de Maxwell dans le chapitre 2. Dans ce chapitre, nous présenterons les équations de Maxwell ainsi que leur discrétisation par éléments finis de Nédélec. À la fin de ce chapitre est décrite la méthode de décomposition de domaine dans laquelle s'inscrit notre solveur multigrille. Finalement, dans le chapitre 3 nous décrivons et justifions les choix faits pour la conception du solveur multigrille. En s'appuyant sur des références bibliographiques récentes, nous déterminerons quelles sont les méthodes actuellement employées pour de tels problèmes ainsi que les difficultés liées à l'utilisation des méthodes multigrilles géométriques et algébriques dans le cadre des équations de Maxwell.

Le seconde partie débute par la description de l'environnement logiciel dans lequel s'inscrivent les travaux de cette thèse (chapitre 4). En se basant sur l'existant, nous déterminons la façon dont est conçu et développé le solveur. En suivant ce qui a été décidé dans le chapitre 4, le chapitre 5 présente la réalisation du solveur en décrivant certains algorithmes et techniques de parallélisation employées. Cette partie est une description plus complète du solveur développé et met en évidence les points complexes de la résolution. Pour ne pas trop obscurcir le discours, cette partie ne présente pas les détails d'implémentation mais plutôt les concepts et les techniques employés.

Finalement le chapitre 6 donne lieu à la validation du solveur multigrille sur divers cas tests. Ce chapitre démontre la capacité du solveur à résoudre le problème des équations de Maxwell. Du fait de la complexité de la résolution mise en évidence lors de la revue bibliographique, de nombreux tests sont réalisés : nous tentons de montrer le bon comportement de la méthode multigrille dans des cas à hautes fréquences ou à indices de matériau élevés qui intéressent le CEA. En dernier lieu, nous montrons les performances parallèles de la méthode développée en exécutant le code sur des milliers de cœurs de calcul, permettant de traiter des problèmes de taille importante que l'on ne pourrait pas résoudre par la méthode actuellement en utilisation au CEA/CESTA. Alors que le plus grand système résolu aujourd'hui par un solveur direct parallèle pour un problème 3D a une taille de 83 millions d'inconnues (en complexe double précision), nous montrons comment, à partir d'un maillage grossier à 21 millions d'inconnues, atteindre une résolution sur un maillage fin à 1.3 milliards d'inconnues avec une précision suffisante sur 1024 cœurs de calcul de TERA100, qui est le dernier calculateur du CEA/DAM.

Première partie

Contexte - État de l'art

Introduction

Dans les chapitres de cette première partie, nous présenterons de façon générale la méthode multigrille ainsi que les équations de Maxwell. Chacune de ces présentations sera suffisamment généraliste pour permettre au lecteur de se familiariser avec les concepts utilisés et de décrire au mieux le contexte dans lequel s'inscrivent les travaux de cette thèse.

Dans la partie multigrille, nous nous plaçons dans le cadre général et nous introduirons les outils mis à notre disposition. Tout ce qui sera présenté ne sera pas utile à la thèse, mais permet de familiariser le lecteur avec les méthodes multigrilles géométriques et algébriques. En ce qui concerne la partie sur les équations de Maxwell, nous verrons aussi la méthode de décomposition de domaine qui conduit à un algorithme itératif de résolution de ces équations dans lequel le solveur multigrille développé ici est destiné à être couplé.

Finalement, en utilisant les divers concepts décrits, nous énoncerons le positionnement des travaux de cette thèse, en basant nos choix par rapport à la littérature scientifique sur le sujet et aux travaux effectués actuellement au sein du service simulation des phénomènes physiques du CEA/CESTA.

Chapitre 1

Les Méthodes multigrilles

Dans cette partie, nous introduirons les concepts à la base de la méthode multigrille introduite dans les années 1970 par Brandt [7]. Avec l'arrivée des machines massivement parallèles et du fait de leur complexité calculatoire optimale, ces méthodes connaissent un regain d'attention. De nombreuses études sont menées afin d'appliquer ces méthodes à une gamme de problèmes de plus en plus large.

En guise d'introduction aux méthodes multigrilles, nous présenterons les méthodes de résolution dites "classiques", comme les méthodes directes et itératives. Nous aborderons ensuite le cœur des méthodes multigrilles en décrivant l'idée sous-jacente à l'origine de ces méthodes, ainsi que les différentes variations algorithmiques de celles-ci. Finalement, les avantages de la méthode multigrille en termes de mémoire, de complexité et de parallélisme seront étudiés.

Cette partie ne se veut qu'une description suffisamment générale des méthodes multigrilles afin de montrer au lecteur toute l'étendue des possibilités qui s'offrent aux développeurs. Les choix effectués dans cette thèse ne seront présentés que dans le chapitre 3 où nous les justifierons aussi par des références bibliographiques.

1.1 Méthodes de résolution classiques

Soit A une matrice carrée de dimension n et $\mathbf{x}$ et $\mathbf{b}$ deux vecteurs. Pour A et y donnés, on cherche à calculer la solution x du système $A\mathbf{x} = \mathbf{b}$. Deux grandes familles de méthodes existent pour résoudre ce type de problème :

- les méthodes directes qui calculent de manière "exacte" la solution ;
- les méthode itératives qui construisent une suite d'approximations convergeant vers la solution.

Dans ce chapitre, nous allons discuter de ces deux types de méthodes, en mettant en avant pour chacune d'entre elles leurs avantages et leurs inconvénients.

1.1.1 Méthodes directes

Les méthodes directes reposent sur la factorisation de la matrice A. D'un point de vue générique, on cherche à exprimer une matrice inversible A sous la forme d'un produit $A =$

$M_1 \cdots M_k$. Pour des matrices M de structures triangulaires ou diagonales, la résolution du système A s'en trouve simplifiée. En considérant L, D, et U des matrices respectivement triangulaire inférieure, diagonale et triangulaire supérieure, la factorisation de A conduit à la résolution de

$$A\mathbf{x} = LDU\mathbf{x} = \mathbf{b}, \tag{1.1}$$

et cette résolution se divise en trois étapes

$$L\mathbf{y} = \mathbf{b}, \tag{1.2}$$
$$D\mathbf{z} = \mathbf{y}, \tag{1.3}$$
$$U\mathbf{x} = \mathbf{z}. \tag{1.4}$$

Du fait de la structure des matrices L, D et U, la résolution de ces trois systèmes linéaires se fait simplement (voir sous-section titrée "Résolution directe", page 19). Cette factorisation générique peut se décliner sous diverses formes :

- la décomposition $A = LU$ (Crout), avec L une matrice triangulaire inférieure et U une matrice triangulaire supérieure à diagonale unité. Cette méthode de factorisation est la plus courante car elle permet de factoriser des matrices non hermitiennes ;
- la factorisation LL^T (Cholesky) est applicable lorsque A est hermitienne. Le principal avantage de cette méthode réside dans l'utilisation de la structure de la matrice ce qui limite la quantité de mémoire nécessaire au stockage lors de la factorisation. En comparaison de la méthode LU, seulement la moitié de la matrice est stockée. La factorisation LDL^T est une variante de cette méthode.

Par la suite, l'inversion directe d'un système ne sera pas énoncée en tant que telle, mais plutôt décrite par les deux étapes qui la composent : la *factorisation* et la *descente-remontée*. Le principal avantage de la méthode directe repose dans le fait que la factorisation de A ne dépend pas du second membre. Une fois calculée, la factorisation de A est peut être réutilisée pour tout second membre. Le coût de p résolutions impliquant la matrice A se réduit alors au coût d'une seule factorisation et de p descentes-remontées qui peuvent par ailleurs être réalisées "en même temps" algorithmiquement.

Factorisation de matrices denses

Bien que ce travail s'inscrive dans le cadre des matrices creuses, la factorisation dense nous sert à poser les bases algorithmiques de la méthode. En effet, la factorisation par blocs de matrices creuses suit le même algorithme que la version dense et se base sur des factorisations de blocs denses. Afin de mieux appréhender le cas creux, nous nous plaçons dans le cadre de la factorisation LU dense

$$\begin{pmatrix} a_{11} & \mathbf{a_{12}}^\mathsf{T} \\ \mathbf{a_{21}} & A_{22} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 \\ \mathbf{l_{21}} & L_{22} \end{pmatrix} \begin{pmatrix} u_{11} & \mathbf{u_{12}}^\mathsf{T} \\ 0 & U_{22} \end{pmatrix} \tag{1.5}$$

avec en minuscules les scalaires, en minuscules grasses les vecteurs et en majuscules les matrices. De cette notation et des contraintes posées sur L et U, l'algorithme de factorisation

procède ainsi : calcul du terme diagonal u_{11} ($l_{11} = 1$), calcul des vecteurs $\mathbf{l_{21}}$ et $\mathbf{u_{12}}$, mise à jour du bloc diagonal $S = A_{22} - \mathbf{l_{21}}.\mathbf{u_{12}}^{\mathsf{T}}$ et enfin factorisation de S. En notant $C(n)$ la complexité de la factorisation d'un système de dimension $n \times n$, on peut exprimer la complexité de manière récursive

$$C(n) = 1 + \underbrace{2(n-1)}_{l_{21} \text{ et } u_{12}} + \underbrace{2(n-1)^2}_{\text{màj. } S} + \underbrace{C(n-1)}_{\text{facto. } S}. \tag{1.6}$$

La complexité algorithmique de la factorisation est de l'ordre de $O(n^3)$.

Factorisation de matrices creuses

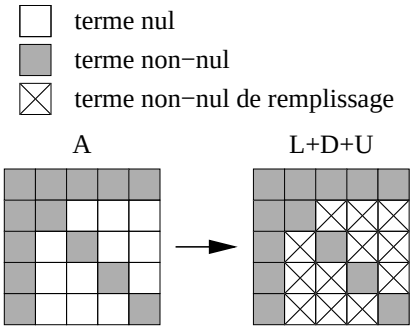
L'algorithme de factorisation des matrices creuses est analogue à la factorisation de matrices denses. Après factorisation, les termes non-nuls de la matrice $L+U$ sont constitués des termes non-nuls de la matrice A, plus des termes de *remplissage* apparus lors de la factorisation. Ce phénomène implique que la quantité de mémoire nécessaire au stockage de la factorisation est supérieure ou égale à celle du stockage de la matrice initiale. Outre l'augmentation de la consommation mémoire, plus le remplissage est important, plus la quantité de calculs nécessaires augmente. Pour limiter le remplissage et ses effets négatifs, une permutation symétrique des lignes et colonnes est effectuée. Cette permutation est en fait une *renumérotation* des inconnues. Un exemple classique de renumérotation réduisant le remplissage est présenté dans la figure 1.1. En l'absence de renumérotation (figure 1.1(a)), le résultat de la factorisation de la matrice creuse conduit à stocker un système dense. Au contraire, après renumérotation (permutation des inconnues 1 et 5) le remplissage est nul (figure 1.1(b)) : seuls les non-zéros du système initial sont des termes non-nuls de la matrice factorisée.

Bien qu'extrême, cet exemple montre bien que le remplissage a d'importantes répercussions sur le coût mémoire et calculatoire de la factorisation. D'une manière générale, la renumérotation est calculée à l'aide d'un partitionneur de graphes utilisant des algorithmes de type dissections emboîtées [44]. Cette technique de renumérotation garantit la non-apparition de termes non-nuls entre inconnues de certaines zones de la matrice. Cette renumérotation conduit à une réécriture par blocs de l'algorithme de factorisation scalaire où toutes les opérations de base se font sur des blocs denses de la matrice creuse [28, 44].

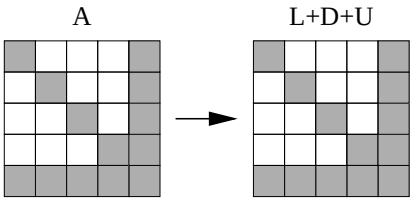
Dans le cas de la factorisation parallèle de matrices creuses, la renumérotation impacte aussi sur la qualité du parallélisme sous-jacent et sur la quantité de communications nécessaires.

Descente-Remontée

Le principe de la résolution est simple : la factorisation du système initial conduit à des matrices triangulaires ou diagonales. La résolution d'un système triangulaire dense est très simple. À chaque étape, un terme du vecteur solution est calculé en utilisant les valeurs précédemment calculées. Le sens de parcours de ces calculs fait que les résolutions de ce type sont appelées *descentes-remontées*. Une résolution dense a une complexité en $O(n^2)$. À l'instar de la factorisation creuse, l'étape de descente-remontée creuse parallèle dépend fortement de la renumérotation et conduit à un réécriture par blocs pour la dissection emboîtée. En général,



(a) sans renumérotation.



(b) avec renumérotation.

FIGURE 1.1: Effet de la renumérotation sur le remplissage.

la qualité du parallélisme de l'étape de résolution est moins bonne que pour la factorisation à cause d'un ratio calcul/communication moins favorable.

1.1.2 Méthodes itératives

Contrairement aux méthodes directes qui calculent "en une seule fois" la solution d'un système linéaire, les méthodes itératives travaillent par étapes. À partir d'une estimation initiale de la solution, les méthodes itératives construisent une suite convergente d'approximations. Parmi ces méthodes, les méthodes de *splitting* sont très largement utilisées par la communauté multigrille et sont présentées dans la partie suivante. Pour d'autres techniques itératives de résolution telles que les méthodes de Krylov [51], se référer à la partie 1.5.1, page 32.

Méthodes de splitting

Les algorithmes de splitting tels que Jacobi et Gauss-Seidel reposent sur la décomposition de la matrice A sous la forme $A = M - N$. On notera $\mathbf{x}^{[k]}$ l'estimation de la solution à l'itération k, on a :

$$\mathbf{x}^{[k+1]} = M^{-1}(\mathbf{b} + N\mathbf{x}^{[k]}) \tag{1.7}$$

$$\iff \mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + M^{-1}(\mathbf{b} - A\mathbf{x}^{[k]}). \tag{1.8}$$

Pour que cet algorithme soit valide, il est impératif que M soit inversible. Tout comme lors de la factorisation, le choix de la structure de la matrice M est primordial : la résolution du système M à chaque itération impose une structure triangulaire ou diagonale à cette matrice.

En considérant $A = L + D + U$ avec U et L triangulaires et D diagonale, la décomposition de Jacobi s'obtient en posant $M = D$ et $N = -(L + U)$. L'itération s'exprime alors

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + D^{-1}(\mathbf{b} - A\mathbf{x}^{[k]}). \tag{1.9}$$

De manière similaire à la méthode de Jacobi, l'algorithme de Gauss-Seidel se base sur la décomposition $M = L + D$ et $N = -U$, ce qui conduit à

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + (L + D)^{-1}(\mathbf{b} - A\mathbf{x}^{[k]}) \tag{1.10}$$

qui peut se réécrire :

$$\mathbf{x}^{[k+1]} = D^{-1}(\mathbf{b} - L\mathbf{x}^{[k+1]} - U\mathbf{x}^{[k]}). \tag{1.11}$$

Bien que très similaires, ces deux méthodes n'ont pas le même comportement. Si l'on regarde de plus près l'algorithme de Gauss-Seidel, on peut voir qu'il est intrinsèquement séquentiel. Si l'on se concentre sur la composante i de $\mathbf{x}^{[k+1]}$, on a la relation suivante :

$$\mathbf{x}_i^{[k+1]} = \frac{1}{D_{ii}} \left[\mathbf{b}_i - \sum_{j=1}^{i-1} (L_{ij}\, \mathbf{x}_j^{[k+1]}) - \sum_{j=i+1}^{n} (U_{ij}\, \mathbf{x}_j^{[k]}) \right]. \tag{1.12}$$

Il est alors évident que le calcul de la i-ième valeur de $\mathbf{x}^{[k+1]}$ nécessite le calcul préliminaire des termes $\mathbf{x}_j^{[k+1]}$ avec $j < i$. Par contre, un avantage de cette méthode est que le calcul peut se réaliser sans avoir recours à des tableaux temporaires. En effet, la seconde somme de l'équation (1.12) n'utilise du vecteur $\mathbf{x}^{[k]}$ que les termes d'indices $j > i$. Ainsi, tous les termes d'indices inférieurs peuvent être remplacés par les valeurs calculées du vecteur $\mathbf{x}^{[k+1]}$. Exprimé ainsi, la méthode de Gauss-Seidel a un problème lié au sens de parcours des vecteurs : le premier terme du vecteur $\mathbf{x}$ est toujours mis à jour à partir des valeurs de l'itération précédente. Pour remédier à ce problème, des méthodes similaires alternant les sens de parcours existent.

Par ailleurs, d'autres méthodes permettent de rendre la méthode parallèle. Parmi ces méthodes, le *red-black* Gauss-Seidel, est particulièrement populaire : au lieu de se baser sur les matrices triangulaires initiales, elle se base sur une renumérotation des inconnues.

Cette méthode repose sur une bi-coloration des inconnues, de telle sorte que deux inconnues de même couleur ne soient pas adjacentes. La structure par blocs de la matrice renumérotée comporte alors les matrices diagonales A_{RR} et A_{BB} d'interactions entre inconnues d'une même couleur, et les matrices de couplage A_{RB} et A_{BR} (figure 1.2). Le fait que les matrices A_{RR} et A_{BB} soient diagonales fait que la méthode est désormais parallèle : dans (1.11), les multiplications $L\mathbf{x}^{[k+1]}$ et $U\mathbf{x}^{[k]}$ mettent à jour les inconnues noires à partir des rouges et inversement. Lors de chacune de ces opérations, le calcul des termes peut se faire en parallèle. Tout comme la version standard du Gauss-Seidel, il est aussi possible d'alterner le sens de parcours pour de meilleurs résultats.

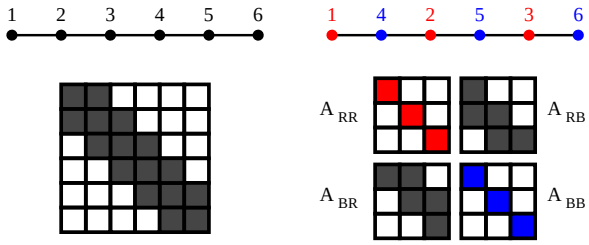


FIGURE 1.2: Red-Black Gauss-Seidel : structure de la matrice avant et après coloration.

D'une manière générale, on peut étudier les méthodes de splitting en exprimant la matrice d'itération à partir de l'erreur commise $\mathbf{e}^{[k]} = \mathbf{x} - \mathbf{x}^{[k]}$ avec la solution exacte $\mathbf{x}$

$$\mathbf{e}^{[k+1]} = M^{-1}N\mathbf{e}^{[k]} \tag{1.13}$$

ou encore

$$\mathbf{e}^{[k+1]} = (I - M^{-1}A)\mathbf{e}^{[k]}. \tag{1.14}$$

Pour assurer la convergence de ces méthodes pour tout vecteur initial $\mathbf{x}^{[0]}$, il faut que la matrice $K = M^{-1}N$ ait un rayon spectral $\rho(K) < 1$. Dans le cas contraire, pour la décomposition du vecteur $\mathbf{e}^{[0]} = \alpha_n\mathbf{v}_n + \cdots + \alpha_0\mathbf{v}_0$, avec $\mathbf{v}_n$ le vecteur propre associé à la plus grande valeur propre $\lambda_n > 1$ on a :

$$\lim_{k\to\infty} \|\mathbf{e}^{[k]}\| = \lim_{k\to\infty} \|\lambda_n^k\mathbf{e}^{[0]}\| = \infty. \tag{1.15}$$

Pour toute matrice K de rayon spectral supérieur à 1 et pour toute estimation initiale non orthogonale à $\mathbf{v}_n$, la méthode de splitting est *divergente*. Pour remédier à ce problème, des méthodes de réduction de rayon spectral existent [4]. La plus simple de ces méthodes consiste à utiliser un paramètre de sous-relaxation ω rendant la méthode convergente (voir section 1.4.1).

1.2 Bases de la méthode multigrille

La méthode multigrille a été développée afin de résoudre des systèmes issus de la discrétisation de problèmes elliptiques [7]. Parmi ces problèmes on trouve par exemple l'équation de Poisson

$$-\nabla \cdot (\nabla \mathbf{u}) = \mathbf{b}. \tag{1.16}$$

Si l'on tente de résoudre ce système avec les méthodes de splitting, on converge très lentement vers la solution. Pour les systèmes de grande taille, cette convergence lente pose problème car elle implique un nombre de produits matrice-vecteur de l'ordre de la dimension du système. Dans de tels cas, le recours aux méthodes itératives n'est plus un choix judicieux en terme

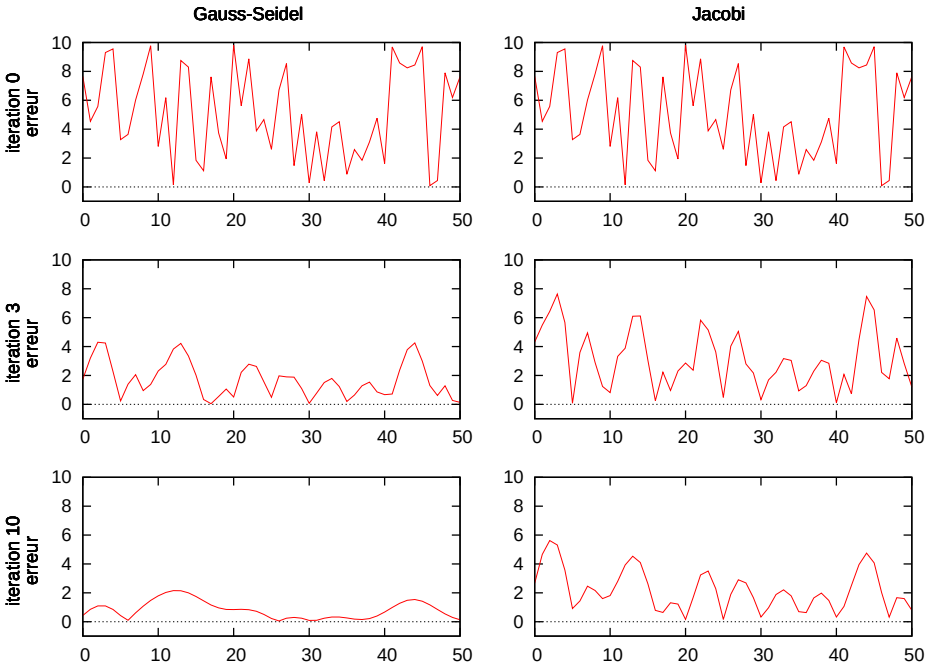


FIGURE 1.3: Équation de Poisson à une dimension. Effet des algorithmes de GS et de Jacobi sur l'erreur. L'axe de abscisses représente le domaine d'étude.

de complexité. Les méthodes de Jacobi et de Gauss-Seidel ont un comportement remarquable sur l'erreur. Si l'on regarde $\mathbf{e}^{[k]} = \mathbf{x} - \mathbf{x}^{[k]}$, on peut voir que plus k est grand moins l'erreur oscille. Cet comportement est illustré dans la figure 1.3.

Cette figure est obtenue en discrétisant le problème de Poisson sur un domaine à une dimension. Le domaine est découpé en intervalles de tailles identiques et l'estimation initiale de la solution est aléatoire. Sur les deux figures du haut, on voit l'erreur initiale $\mathbf{e}^{[0]}$ pour les algorithmes de Gauss-Seidel et de Jacobi. Comme on s'est placé dans les mêmes conditions pour les deux algorithmes, ces courbes sont identiques. On présente ensuite sur les seconde et troisième lignes l'erreur $\mathbf{e}^{[3]}$ et $\mathbf{e}^{[10]}$. Il est évident que ces deux erreurs sont beaucoup moins oscillantes que $\mathbf{e}^{[0]}$. Afin de s'en assurer, la figure 1.4 présente pour les mêmes vecteurs d'erreur la transformée de Fourier discrète calculée par la bibliothèque FFTW. On peut voir que les hautes fréquences (partie centrale du spectre) décroissent plus rapidement que toutes les autres fréquences.

Le comportement mis en évidence par ces deux graphiques est intéressant : on vient de montrer que les méthodes de splitting ont une action efficace contre les composantes hautes fréquences de l'erreur. Suite à ce constat et en remarquant que le vecteur erreur paraît de plus en plus lisse au cours des itérations, nous appellerons *lisseur* tout solveur ayant ce comportement.

La résolution précise de tels systèmes linéaires nécessite alors de réduire de manière efficace les composantes basses fréquences de l'erreur. Dans le cas présent, la fréquence de l'erreur est directement liée au nombre de points utilisés pour la discrétisation du domaine. Toujours dans le cadre de la discrétisation de l'équation de Poisson en 1D (figure 1.3), si l'on réduit le nombre de points de discrétisation alors un vecteur "lisse" devient oscillant sur le nouveau maillage [8]. En prenant un point sur deux, la fréquence relative des composantes du vecteur

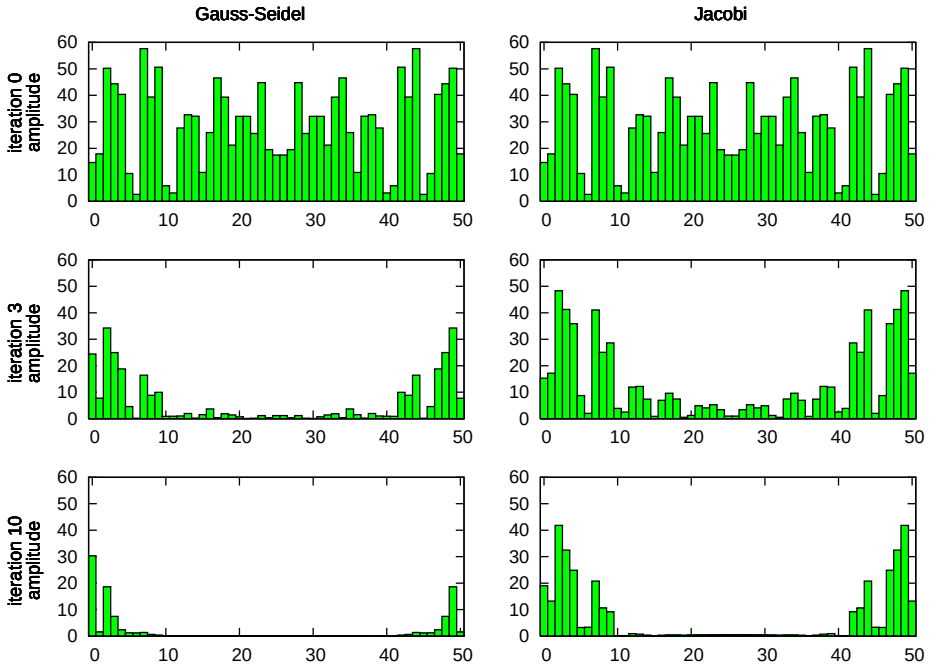


FIGURE 1.4: Effet des algorithmes de GS et de Jacobi sur les modes oscillatoires de l'erreur. En abscisse est représentée la transformée de Fourier discrète de l'erreur.

erreur est multipliée par deux. Ainsi, le déraffinement du maillage va provoquer un glissement du spectre vers les hautes fréquences. Par déraffinement, on peut donc faire apparaître des composantes hautes fréquences dans un vecteur qui ne comportait que des composantes basses fréquences. Un lisseur appliqué sur ce nouveau vecteur aura donc plus d'effet que ce même lisseur sur le vecteur non raffiné. De ce constant naît l'idée principale du multigrille qui est d'utiliser différents niveaux de raffinement pour résoudre un problème.

De façon plus rigoureuse, on suppose désormais que l'on a des pas de discrétisation de longueur h pour le *maillage fin* et de $2h$ pour le *maillage grossier*. On note A_f le système issu de la discrétisation du problème sur le maillage fin et A_g le système sur le niveau grossier. On veut alors résoudre, sur le niveau fin

$$A_f \mathbf{x}_f = \mathbf{b}_f. \tag{1.17}$$

On suppose désormais qu'à une étape k quelconque de la résolution, on ait calculé une estimation de la solution $\mathbf{x}_f^{[k]}$. On peut alors exprimer le résidu $\mathbf{r}_f^{[k]}$ et l'erreur $\mathbf{e}_f^{[k]}$ par les relations suivantes :

$$\mathbf{r}_f^{[k]} = \mathbf{b}_f - A_f \mathbf{x}_f^{[k]} \tag{1.18}$$

$$\mathbf{e}_f^{[k]} = \mathbf{x}_f - \mathbf{x}_f^{[k]} \tag{1.19}$$

avec $\mathbf{x}_f$ la solution exacte du système (1.17). Comme nous ne connaissons pas cette solution exacte, on exprime plutôt l'erreur en fonction du résidu

$$\mathbf{r}_f^{[k]} = A\mathbf{e}_f^{[k]}. \tag{1.20}$$

Cette expression, liant les vecteurs résidu et erreur permet de calculer l'erreur commise sur le niveau grossier

$$A_g\mathbf{e}_g^{[k]} = \mathbf{r}_g^{[k]}. \tag{1.21}$$

Il ne reste plus que la définition des opérateurs de transfert pour obtenir la méthode multigrille à deux niveaux. En notant R l'opérateur de *restriction*, transformant un vecteur fin en vecteur grossier, et P l'opération inverse dite de *prolongation*, on a pour tout vecteur $\mathbf{v}$:

$$\mathbf{v}_g = R\mathbf{v}_f \tag{1.22}$$
$$\mathbf{v}_f = P\mathbf{v}_g. \tag{1.23}$$

Sans avoir explicitement exprimé en quoi consistent ces opérateurs, cette définition nous permet de mettre en place l'algorithme multigrille à deux niveaux :

1. Appliquer ν_1 itérations de lisseur sur le niveau fin ;
2. Calculer le résidu et le transmettre (par l'opérateur de restriction) sur la grille grossière ;
3. Résoudre le système $A\mathbf{e} = \mathbf{r}$ sur le niveau grossier ;
4. Calculer l'erreur sur le niveau fin et corriger l'estimation (par l'opérateur de prolongation) ;
5. Lisser à nouveau en appliquant ν_2 itérations pour réduire les composantes hautes fréquences introduites par l'interpolation.

Plus particulièrement, en omettant les étapes de lissage et en considérant la résolution grossière comme exacte, on exprime la correction grossière sous la forme

$$\mathbf{r}_f = \mathbf{b}_f - A_f\mathbf{x}_f \tag{1.24}$$
$$\mathbf{r}_g = R\mathbf{r}_f \tag{1.25}$$
$$\mathbf{e}_g = A_g^{-1}\mathbf{r}_g \tag{1.26}$$
$$\mathbf{e}_f = P\mathbf{e}_g \tag{1.27}$$
$$\mathbf{x}_f = \mathbf{x}_f + \mathbf{e}_f. \tag{1.28}$$

Finalement, en introduisant le lisseur sous la forme d'une matrice S (de l'anglais *smoother*) et en ne considérant que le terme d'erreur sur le maillage fin, on obtient la matrice d'itération suivante

$$\mathbf{e}_f^{[k+1]} = S^{\nu_2}(I - PA_g^{-1}RA_f)S^{\nu_1}\mathbf{e}_f^{[k]}. \tag{1.29}$$

Jusqu'à présent, la méthode multigrille considérée ne se basait que sur deux niveaux. Dans les expressions précédentes, la résolution grossière (1.26) peut être remplacée par une méthode multigrille. On définit alors le cycle multigrille en V à nombre de niveaux quelconque.

Nous avons exprimé ici les bases de la méthode multigrille sans définir exactement les composantes de cette méthode. Afin d'obtenir une méthode performante, la relation (1.29) composée de ces briques de base doit avoir certaines propriétés. Dans la suite de ce chapitre, nous verrons un peu plus dans le détail quelles sont les options qui s'offrent à nous en termes de choix de lisseurs, types de prolongation et en quoi consiste la résolution des systèmes impliquant A_g.

1.2.1 Raffinement et déraffinement

Comme nous l'avons vu dans les paragraphes précédents, le raffinement n'intervient pas en tant que tel dans les relations du calcul. Néanmoins, dans le cadre de l'approximation par éléments finis, ce raffinement est primordial : c'est le contexte des éléments finis qui définit l'interpolation utilisée. Contrairement au cas géométrique qui nous intéresse ici, toute une branche de la communauté scientifique travaillant sur le multigrille développe actuellement des solveurs multigrilles algébriques [8, 20] (AMG), faisant abstraction de la géométrie. Cela permet non seulement d'éviter la construction de maillages, mais aussi de fournir une méthode multigrille de type boîte noire. Comme nous nous inscrivons dans le cadre des méthodes géométriques (cf. 3), il faut que le raffinement et le déraffinement aient un sens géométrique. La difficulté de cette étape repose sur la capacité de calculer un maillage grossier à partir d'un maillage fin. En effet, même dans le cas simple d'une grille cartésienne que l'on veut déraffiner en groupant les cellules par quatre (figure 1.5), on peut voir que plusieurs maillages grossiers correspondent à ce même traitement. Cette non unicité de déraffinement pose problème dans le cadre d'un solveur parallèle. Sans concertation au préalable, chaque processeur peut potentiellement déraffiner de manière différente, rendant le domaine de calcul inconsistant entre les processeurs (maillages non superposables à l'interface).

La méthode la plus simple et la plus rapide est d'effectuer l'opération inverse : partir d'un maillage grossier pour calculer un maillage fin. Avec un algorithme permettant de calculer de façon déterministe un maillage grossier à partir d'un maillage fin (par stockage d'information), cette étape ne pose plus de problème. L'information peut être explicite par stockage simultané des deux maillages, mais elle peut aussi être implicite en conservant les informations relatives au maillage grossier dans le maillage fin. Cette dernière méthode est plus contraignante que la précédente, mais a l'avantage de limiter la mémoire nécessaire.

1.2.2 Restriction et prolongation

Comme on a pu le voir jusqu'ici, la méthode multigrille doit disposer de méthodes d'interpolation inter-niveaux performantes et précises : performantes car elles sont très souvent utilisées, précises car il faut qu'un vecteur grossier soit une représentation "fidèle" du vecteur fin. La définition de ces opérateurs est alors une phase importante de la conception de la méthode.

Idéalement, on voudrait que les opérations de restriction et de prolongation aient la propriété suivante :

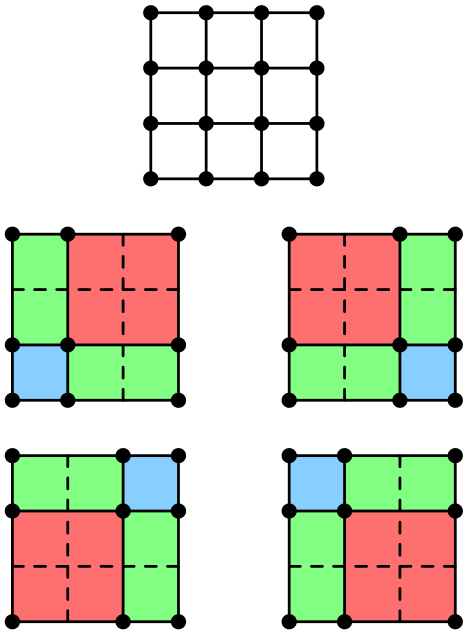


FIGURE 1.5: Grille initiale et les quatre grilles grossières correspondantes.

$$PR = I_f \text{ et } RP = I_g \tag{1.30}$$

avec P et R des matrices rectangulaires de dimensions respectives $\mathbb{R}^{\mathcal{F}\times\mathcal{G}}$ et $\mathbb{R}^{\mathcal{G}\times\mathcal{F}}$ avec $\mathcal{F}$ et $\mathcal{G}$ le nombre d'inconnues des maillages fin et grossier. Cette hypothèse permet de passer d'un niveau à l'autre de manière transparente ($PR\mathbf{x} = \mathbf{x}$) et nous permettra par la suite d'exprimer l'opérateur grossier en fonction de l'opérateur fin et inversement.

De plus, pour les problèmes elliptiques, il est souvent imposé que les lignes de P soient une partition de l'unité (pour toute ligne i on a $\sum_k P_{ik} = 1$). Cette restriction implique qu'un champ constant sur le niveau fin le sera aussi sur le niveau grossier. Cette propriété est utile car, comme on l'a dit précédemment, la composante constante de l'erreur (de fréquence nulle) n'est pas réduite par le lisseur. Il est alors indispensable d'annuler cette composante sur la grille grossière. La partition de l'unité permet alors de transférer exactement cette composante jusqu'au niveau grossier et ainsi favoriser son annulation.

Dans de nombreux cas, la méthode multigrille est utilisée en tant que préconditionneur. Dans ces conditions, une dernière contrainte peut exister. En regardant l'opérateur de réduction d'erreur (1.29), on peut remarquer que cette relation n'est pas forcément symétrique. Certaines méthodes itératives que l'on préconditionne par la méthode multigrille nécessitent un préconditionneur symétrique (cas du Gradient Conjugué). Dans ce cas, il faut imposer la symétrie de (1.29). Très souvent, A_g et A_f sont symétriques ainsi que le lisseur. Il faut alors que $R = P^\mathsf{T}$ afin de conserver la symétrie du système.

Toutes les propriétés présentées ici sont des contraintes théoriques. Dans le cadre de maillages cartésiens avec une discrétisation différences finies, il est relativement aisé de les respecter. Par contre avec des maillages irréguliers (cas typique des éléments finis), imposer le respect de ces règles nécessite des calculs de complexité algorithmique élevée : multiples parcours de la

géométrie et accès à des données non contiguës. Dans le cadre de l'AMG, Il est alors fréquent d'avoir des temps de prétraitement (calcul des opérateurs de restriction et de prolongation) supérieurs au temps de résolution.

1.2.3 Calcul de l'opérateur grossier

Le point fort de la méthode multigrille réside dans le fait que la résolution (1.26) concerne désormais le système grossier au lieu du système fin comme dans (1.17). Afin de tirer parti de ce fait, il faut que la complexité de la résolution et du calcul du système grossier soit (nettement) inférieure à la complexité de la résolution du système fin. Plusieurs méthodes sont disponibles pour calculer l'opérateur sur le niveau grossier :

– rediscrétisation du problème avec un pas de discrétisation inférieur ;
– calcul d'un opérateur grossier en utilisant les opérateurs de prolongation et de restriction, aussi appelé méthode de Galerkin.

Rediscrétisation du problème

La rediscrétisation du problème est la manière la plus directe et la plus simple pour obtenir le nouvel opérateur. Comme pour une discrétisation classique, l'opérateur est calculé à partir de la géométrie et des éléments finis utilisés. Par contre, cette rediscrétisation demande un couplage fort entre le code de calcul et le solveur : la discrétisation se base sur une géométrie raffinée calculée pour le solveur multigrille. De plus, le raffinement est intrinsèquement lié au problème à traiter. Le raffinement ne peut donc être calculé ni par le code de calcul, dont le but n'est pas d'effectuer des raffinements, ni par le solveur multigrille qui n'a pas une connaissance approfondie du problème à traiter (éléments finis employés, conventions de numérotation, ...). Ainsi, les solveurs multigrilles géométriques utilisant la méthode de rediscrétisation sont souvent développés "sur mesure" pour une gamme de problèmes bien précise. Contrairement à cela, les méthodes algébriques se basent sur la définition même des opérateurs de restriction et de prolongation afin d'estimer l'opérateur grossier.

La méthode de Galerkin

La méthode de calcul de l'opérateur par l'utilisation de la restriction et de la prolongation est appelée méthode de Galerkin. En considérant les opérateurs idéaux de prolongation et de restriction, on peut écrire les relations suivantes :

$$P(A_g(R\mathbf{x}_f)) = P\mathbf{b}_g \Longleftrightarrow PA_gR = A_f \tag{1.31}$$
$$R(A_f(P\mathbf{x}_g)) = R\mathbf{b}_f \Longleftrightarrow RA_fP = A_g. \tag{1.32}$$

Pour que cette méthode soit performante la contrainte sur les opérateurs P et R est importante : ils doivent à la fois interpoler correctement les vecteurs, mais aussi permettre de calculer le nouvel opérateur. L'hypothèse des opérateurs idéaux utilisée ici est que la solution fine est égale à la prolongation de la solution grossière, ce qui n'est pas vrai dans la majeure partie des cas.

1.3 Les cycles multigrilles

Comme indiqué plus haut, la résolution effectuée sur le niveau grossier est laissée au choix de l'utilisateur. En utilisant les divers outils à notre disposition, nous pouvons définir ce qui est appelé des *cycles multigrilles*. Nous parlerons plus particulièrement des *Cycles en V*, des *Cycles en W* et le *full-multigrid* (FMG) [8].

1.3.1 Le cycle en V

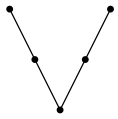


FIGURE 1.6: Cycle en V.

Le cycle en V, présenté dans la figure 1.6, est le cycle multigrille le plus classique. L'algorithme est obtenu en remplaçant la résolution grossière de l'algorithme à deux niveaux par une méthode multigrille. On obtient l'algorithme présenté dans la procédure 1.1.

Procédure 1.1 : vcycle(A_f, x, b)

Entrées : A_f, x, b
Sorties : x

```
1 x := lisseur(A_f, x, b)
2 r := restriction(b - A_f x)
  /* ... calcul de A_g                                   */
3 e := vcycle(A_g, 0, r)
4 x := x + prolongation(e)
5 x := lisseur(A_f, x, b)
```

Dans cet algorithme, nous supposons disposer d'une fonction *lisseur* prenant A, x, et b et retournant un vecteur y de telle sorte que $Ay \simeq b$ avec y géométriquement lisse comparé à x.

En poussant le concept jusqu'au bout, on peut voir que cet algorithme manipule des niveaux de plus en plus petits en termes de nombre d'inconnues. Inévitablement, il arrive un moment où le système ne possède plus qu'une seule inconnue. Cette résolution se fait directement. Le test d'arrêt sur le niveau, non présenté dans l'algorithme, est alors indispensable. De plus, ce dernier permet à l'utilisateur de stopper l'algorithme à un niveau donné, là où il considère que le lisseur est suffisamment précis ou que la résolution directe est possible à moindre coût.

1.3.2 Le cycle en W

Le cycle en W est une extension du cycle en V. Le principe est semblable au cycle en V, mis à part que la récurrence est plus complexe : en effet, au lieu de n'utiliser qu'une seule correction grossière, cette méthode repose sur deux corrections séparées par une étape de lissage. La figure 1.7 présente les changements de niveaux effectués. Tout comme le cycle en V, il arrive

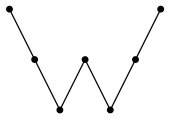


FIGURE 1.7: Cycle en W.

un moment où le système ne peut plus être déraffiné. Ici aussi la résolution sera directe, ou l'algorithme sera stoppé avant d'arriver à ce niveau.

L'avantage du cycle en W par rapport au cycle en V réside dans la qualité de la correction calculée et remontée sur le niveau fin. Toutefois, ce gain se fait au détriment de la complexité algorithmique ; Alors que l'algorithme du cycle en V peut s'exprimer de façon récursive et non récursive, le cycle en W est plus complexe à exprimer de façon non récursive. De ce fait, l'implémentation du cycle en W nécessite plus de travail afin de garantir l'efficacité de la méthode.

1.3.3 Le cycle full-multigrid

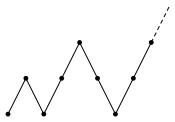


FIGURE 1.8: Full-multigrid.

Le FMG, à l'inverse des deux algorithmes précédents, n'est pas un cycle à proprement parler. Cette méthode, à la différence des deux autres, retourne un résultat sur un niveau plus fin que le niveau d'entrée. La figure 1.8 représente assez fidèlement le traitement effectué. La description algorithmique du FMG est présentée dans la procédure 1.2. Cette description algorithmique ne met pas en évidence les tests d'arrêt ni les raffinements de maillages. Malgré tout, cette description algorithmique met en évidence la principale différence entre les cycles précédents et le FMG : alors que les cycles en V et en W ne restreignent et ne prolongent que des résidus et erreurs, le FMG interpole une solution. Pour chaque niveau atteint un nouveau cycle en V est débuté avec sa propre estimation initiale et son propre second membre. Pour garantir la validité physique de la solution, le second membre de chaque niveau ne peut être calculé par prolongation mais doit être calculé en fonction de la physique du problème et de sa discrétisation.

Procédure 1.2 : FMG(A_g, x_g, b_g) algorithme simplifié

Entrées : A_g, x_g, b_g
Sorties : x_f
1 x_g := `lisseur(`A_g, x_g, b_g`)`
2 x_f := `prolongation` (x_g)
`/* ... calculs de` A_f `et` b_f `*/`
3 x_f := `vcycle(`A_f, x_f, b_f`)`

1.4 Le lisseur

Beaucoup de méthodes sont à notre disposition pour effectuer le lissage. Les plus simples et les plus utilisées d'entre elles sont les méthodes de Jacobi et de Gauss-Seidel présentées plus tôt dans ce chapitre. Malgré tout, d'autres méthodes sont aussi disponibles comme les lisseurs polynômiaux de type Chebyshev [1]. De nombreux articles [38, 56] proposent des modifications améliorant la convergence. Du fait de leur complexité (évaluation de valeurs propres de la matrice), leur utilisation est peu répandue en comparaison des solveurs de type splitting.

1.4.1 Méthodes de splitting

Les méthodes présentées en début de chapitre (Jacobi, Gauss-Seidel) sont celles utilisées dans le cas général. Selon le type d'équation que l'on étudie, certaines modifications doivent être apportées à ces méthodes pour en améliorer la convergence. La plus simple d'entre elles consiste à utiliser un facteur de relaxation ω de telle sorte que l'itération (1.7) se transforme en

$$\mathbf{x}^{[k+1]} = (1-\omega)\mathbf{x}^{[k]} + \omega\mathbf{x}^{[k+\frac{1}{2}]} \tag{1.33}$$

avec

$$\mathbf{x}^{[k+\frac{1}{2}]} = M^{-1}(\mathbf{b} + N\mathbf{x}^{[k]}), \tag{1.34}$$

ou plus simplement

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \omega M^{-1}(\mathbf{b} - A\mathbf{x}^{[k]}) \tag{1.35}$$

avec comme itération relative à l'erreur

$$\mathbf{e}^{[k+1]} = ((1-\omega)I + \omega M^{-1}N)\ \mathbf{e}^{[k]}. \tag{1.36}$$

En notant $R_d = M^{-1}N$ l'opérateur de réduction d'erreur de l'équation (1.13) et R_ω le nouvel opérateur obtenu, on a alors :

$$R_\omega = (1-\omega)I + \omega R_d. \tag{1.37}$$

Dans le cas où $\omega = 1$, on retrouve l'itération de l'algorithme de splitting original. Si $\omega = 0$, par contre, l'algorithme ne converge plus ($R_\omega = I$). Une sous-relaxation, utilisant un facteur $\omega < 1$, permet de réduire le rayon spectral de l'opérateur d'itération et résout ainsi le problème des itérations divergentes ($\rho(K) > 1$). La méthode de Jacobi utilisant un facteur de sous-relaxation est aussi appelée *damped Jacobi*.

1.4.2 Autres méthodes

Le lisseur tient une place importante dans la méthode multigrille. Pour cette raison, de nombreux travaux ont porté sur l'amélioration des méthodes employées. Parmi ces lisseurs, on trouve :

- les lisseurs polynomiaux et lisseurs de Chebyshev dont la qualité de lissage repose sur une estimation fine des valeurs propres de la matrice ;
- le *Block Gauss-Seidel*, constitué d'un Jacobi par blocs avec résolution des blocs diagonaux par Gauss-Seidel. Le potentiel de lissage de cette méthode se situe entre le Jacobi et le Gauss-Seidel classiques pour des tailles de blocs suffisantes [4] ;
- l'*Overlapping Schwarz* [53], basé sur une méthode de Schwarz additif avec recouvrement. Chaque sous-domaine de la méthode de Schwarz est constitué des éléments adjacents à un nœud du maillage. Pour les équations de Maxwell, ce type de lisseur est performant, mais nécessite beaucoup de communications en parallèle.

1.5 La méthode multigrille comme préconditionneur

La méthode multigrille utilisée à ses origines comme un solveur nécessite un réglage parfois délicat de l'ensemble des paramètres qui gouvernent ses performances. Pour cette raison, le multigrille est alors souvent utilisé en tant que préconditionneur pour des méthodes de Krylov. Son faible coût en calcul et en mémoire en font un candidat idéal pour le préconditionnement. Les méthodes pour lesquelles le multigrille s'est avéré être un préconditionneur efficace font principalement partie de la famille des méthodes de Krylov, telles que le gradient conjugué [29] (GC), et le GMRES [51].

1.5.1 Méthodes de Krylov

Outre les méthodes de splitting présentées plus tôt, une autre gamme de méthodes de résolution existe. Cette méthode utilise des opérations de projection. Soit $V = [\mathbf{v}_1, \cdots, \mathbf{v}_m]$ une matrice de dimension $n \times m$ dont les $\mathbf{v}_i$ forment une base d'un espace $\mathcal{K}$ et $W = [\mathbf{w}_1, \cdots \mathbf{w}_m]$ une autre matrice de dimension $n \times m$ dont les vecteurs forment une base d'un espace $\mathcal{L}$. L'opération de projection consiste, à partir d'une estimation initiale $\mathbf{x}_0$, de trouver une approximation $\hat{\mathbf{x}}$ de la solution de telle sorte que $\hat{\mathbf{x}} \in \mathbf{x}_0 + \mathcal{K}$ et $(\mathbf{b} - A\hat{\mathbf{x}}) \perp \mathcal{L}$. Nous appellerons $\mathcal{K}$ l'espace d'expansion et $\mathcal{L}$ l'espace de projection. En utilisant la base de l'espace $\mathcal{K}$, l'approximation $\hat{\mathbf{x}}$ devient

$$\hat{\mathbf{x}} = \mathbf{x}_0 + V\mathbf{y}. \tag{1.38}$$

En utilisant l'orthogonalité avec l'espace de projection et la définition de $\hat{x}$, on obtient

$$W^{\mathsf{T}}(\mathbf{b} - A\mathbf{x}_0 - AV\mathbf{y}) = 0. \tag{1.39}$$

En notant $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ le résidu associé à la première approximation, il vient

$$W^{\mathsf{T}}\mathbf{r}_0 = W^{\mathsf{T}}AV\mathbf{y} \Longleftrightarrow \mathbf{y} = (W^{\mathsf{T}}AV)^{-1}W^{\mathsf{T}}\mathbf{r}_0. \tag{1.40}$$

En introduisant la définition de $\mathbf{y}$ dans la relation (1.38), on obtient finalement

$$\mathbf{x} = \mathbf{x}_0 + V(W^{\mathsf{T}}AV)^{-1}W^{\mathsf{T}}\mathbf{r}_0. \tag{1.41}$$

La matrice $W^{\mathsf{T}}AV$, nécessaire au calcul de la solution, est de dimension m. Pour que cette méthode soit performante, il faut que cette dimension soit la plus petite possible. Les espaces d'expansion et de projection doivent quant à eux être suffisamment "riches" pour représenter correctement la correction de x_0. On a alors un compromis à faire entre la taille du système projeté à résoudre et la qualité de la solution.

Parmi les méthodes non-stationnaires, on retrouve les méthodes de Krylov. Ces méthodes utilisent les espaces de Krylov pour construire l'espace d'expansion, soit

$$K_m(A, \mathbf{r}_0) = [\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \cdots, A^m\mathbf{r}_0]. \tag{1.42}$$

Le gradient conjugué, MINRES [43] et GMRES [51] font parti de cette classe de solveurs. Afin de mieux cerner en quoi consistent ces méthodes, nous présenterons rapidement en quoi consiste le GMRES.

1.5.2 La méthode GMRES

La méthode GMRES (*Generalized Minimum RESidual*) se base sur la minimisation de la norme du résidu $\mathbf{r} = \mathbf{b} - A\hat{\mathbf{x}}$; on a

$$\|\mathbf{r}\| = \|\mathbf{b} - A\hat{\mathbf{x}}\| = \|\mathbf{b} - A\mathbf{x}_0 - AV\mathbf{y}\| = \|\mathbf{r}_0 - AV\mathbf{y}\| \tag{1.43}$$

avec r_0 le résidu associé à l'approximation initiale x_0. La minimisation de $\|r\|$ revient à trouver y de telle sorte que

$$(\mathbf{r}_0 - AV\mathbf{y}) \perp AV. \tag{1.44}$$

En comparant cette relation et la définition des méthodes non-stationnaires, on peut conclure que l'espace d'expansion est relié à l'espace de projection par $W = AV$, avec V l'espace de Krylov de dimension m.

La méthode GMRES se décompose alors en deux principales étapes :
– la construction d'une base de l'espace de Krylov ;
– la minimisation du résidu.

La construction d'une base de l'espace de Krylov implique le calcul d'une base orthonormale de dimension m du sous espace $K_m(A, \mathbf{r}_0)$. Les orthogonalisations sont calculées par l'algorithme de Gram-Schmidt [24]. Le résultat de cette opération est la factorisation de la matrice AV_m sous la forme

$$AV_m = V_{m+1}H_{m+1,m} \tag{1.45}$$

avec V_m l'espace d'expansion de dimension $n \times m$ et $H_{m,m}$ une matrice de Hessenberg supérieure de dimension $m \times m$. La matrice $H_{m+1,m}$ est la matrice $H_{m,m}$ augmentée d'une ligne. En posant $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|$ la première colonne de V_m et en notant que $\forall i \neq j \mathbf{v}_i \perp \mathbf{v}_j$, on a la relation suivante, relative au résidu

$$\|\mathbf{r}\| = \|\mathbf{r}_0 - AV\mathbf{y}\| = \|\mathbf{r}_0 - V_{m+1}H_{m+1,m}\mathbf{y}\|. \tag{1.46}$$

La seconde partie de l'algorithme, consistant à minimiser le résidu, se réduit à un problème de moindres carrés. La méthode employée repose sur une suite de rotations de Givens dont nous ne donnerons pas le détail ici. L'algorithme GMRES réalise ces deux opérations en même temps ; pendant la construction de l'espace de Krylov, la minimisation est effectuée.

Le GMRES est une méthode robuste de résolution. Par contre, plus le nombre d'itérations augmente, plus la base orthonormale est grande. De plus, chaque nouveau vecteur qu'on y ajoute doit être orthogonal aux précédents. Pour un nombre important d'itérations, les besoins en mémoire et en temps de calcul deviennent alors importants. Afin d'améliorer la convergence et ainsi réduire le nombre d'itérations nécessaires, il est possible de préconditionner le système avant la résolution. Ce préconditionnement permet aussi dans le cas où la matrice A est mal conditionnée de faciliter l'étape d'orthogonalisation. Le système préconditionné que l'on traite alors s'écrit sous la forme

$$M_1^{-1}AM_2^{-1}\mathbf{z} = M_1^{-1}\mathbf{b} \tag{1.47}$$

avec $\mathbf{x} = M_2^{-1}\mathbf{z}$ et $M_1M_2 \simeq A$. Lorsque $M_2 = I$ ou $M_1 = I$, on a alors un préconditionnement respectivement à gauche ou à droite. Une restriction au préconditionnement à gauche existe car il faut faire attention à ce que M_1 soit constant au cours des itérations. En effet, si l'inversion de M_1 est réalisée par une méthode non déterministe telle qu'un solveur itératif, alors la factorisation (1.45) n'est plus valide. Dans de tels cas, seul le préconditionnement à droite est autorisé.

1.6 Avantages de la méthode multigrille

1.6.1 Complexité mémoire

La méthode multigrille, ne mettant en jeu que des opérateurs simples, a l'avantage de consommer très peu de mémoire. Si l'on fait le bilan des données potentiellement stockées, on retrouve pour chaque niveau :

– la géométrie ;
– l'opérateur A ;
– le second membre ;
– la solution.

Stockage de la géométrie

Tout d'abord, le stockage de la géométrie n'a de sens que dans la méthode multigrille géométrique. On peut se poser la question de l'importance du stockage du maillage sur chaque niveau. Si l'on considère le cas du V-cycle sur une grille cartésienne simple, il est alors possible de calculer pour chaque niveau le maillage. Cette possibilité permet de ne stocker le maillage que sur le niveau fin. De telles techniques de raffinement peuvent aussi être développées pour les maillages non structurés, mais sont plus complexes à mettre en place.

De plus, si on considère désormais le FMG, on peut facilement calculer un maillage fin contenant le maillage grossier, les opérations de raffinement et de déraffinement s'en trouvant ainsi facilitées. Pour des cas plus complexes, il sera par contre indispensable de stocker cette géométrie sur chaque niveau.

Finalement, même si les maillages de tous les niveaux doivent être stockés, on peut considérer asymptotiquement que la quantité de mémoire nécessaire au stockage de la géométrie est de l'ordre de la taille de la géométrie fine. Pour le multigrille algébrique, cette question du stockage de la géométrie n'a pas lieu d'être.

Opérateur A

La mémoire nécessaire au stockage de l'opérateur dépend de la méthode de calcul employée. Si l'opérateur est calculé par rediscrétisation, le stockage nécessaire se limite à la taille de l'opérateur sur le niveau fin. Dans le cas des méthodes algébriques, où l'opérateur est calculé par méthode de Galerkin on peut oublier l'opérateur à chaque niveau pour le recalculer par les opérations de Galerkin :

$$A_f = PA_gR, \tag{1.48}$$
$$A_g = RA_fP. \tag{1.49}$$

Une seconde solution consiste à conserver pour chaque niveau la matrice de l'opérateur. Bien que coûteuse en mémoire cette technique a pour avantage de limiter la quantité de calculs à effectuer.

La première de ces solutions repose sur la qualité de l'approximation effectuée par la méthode de Galerkin. En effet, ces relations ont été établies sur la base d'opérateurs de prolongation et de restriction idéaux ($x = PRx = RPx$). Dans la pratique, ces égalités ne sont pas vraies. Plus précisément, on a

$$PR\mathbf{x} = \mathbf{x} + \delta \tag{1.50}$$

avec δ un bruit d'interpolation. En enchaînant les calculs par méthode de Galerkin, les erreurs d'interpolation s'accumulent. La seule solution acceptable est alors le stockage de l'opérateur à chaque niveau. Tout comme dans le cas de la géométrie, on peut en conclure que la mémoire nécessaire au stockage de tous les opérateurs est du même ordre de grandeur que le stockage de l'opérateur du niveau fin.

Le second membre

Le stockage du second membre est différent selon le cycle qui est utilisé. Il faut remarquer que les seconds membres utilisés sont de deux types : ceux du problème physique, qui ne sont manipulés que sur le niveau fin, et les résidus utilisés dans les niveaux grossiers. Le second membre "physique" peut être considéré comme une donnée calculable au même titre que l'opérateur A. Dans la majeure partie des cas, les codes de simulation permettent de calculer les seconds membres facilement. Les résidus sont quant à eux intrinsèquement liés à la résolution. Il est alors impératif de les stocker. Pour plus d'homogénéité, on stockera tout second membre, qu'il soit physique ou de type résidu. Finalement, pour chaque niveau de grille c'est un vecteur à stocker. Comme toujours, ce stockage sera du même ordre de grandeur que celui du niveau fin.

1.6.2 Complexité calculatoire de la méthode multigrille

Très souvent dans la littérature, la méthode multigrille est présentée comme étant une méthode optimale [8]. Cette complexité linéaire est obtenue en considérant la méthode multigrille sur un maillage cartésien. Une discrétisation par différences finies de l'équation de Poisson sur ce type de maillage se traduit par une matrice bande. Une méthode de splitting telle que la méthode de Jacobi utilisée en tant que lisseur est alors de complexité linéaire. Dans ces conditions, le produit matrice-vecteur de la relation (1.13) est en réalité une combinaison linéaire d'une inconnue avec son voisinage. Le nombre d'opérations nécessaires à ce produit est de l'ordre de $3n$ sur un problème 1D, $5n$ en 2D et $7n$ en 3D. En supposant le raffinement et les opérateurs inter-niveaux de complexité linéaire, on a une complexité de l'ordre de

$$c_l = kn_l + c_{l-1} \tag{1.51}$$

où c_l est la complexité de résolution au niveau l et k une constante. Pour évaluer la complexité de c_{l-1} en 2D, il suffit de constater que le nombre d'inconnues n_{l-1} au niveau $l-1$ est quatre fois moindre que le nombre d'inconnues sur le niveau l, soit

$$n_{l-1} = \frac{1}{4} n_l. \tag{1.52}$$

On a alors par récurrence

$$c_l = \frac{k}{4^0} n_l + \frac{k}{4^1} n_l + \cdots + \frac{k}{4^{l-1}} n_l + c_0 \;\Leftrightarrow\; c_l = c_0 + kn_l \sum_{i=0}^{l-1} (\frac{1}{4^i}) \tag{1.53}$$

avec c_0 la complexité de la résolution grossière. Dans le cas extrême, le système du niveau zéro se réduit à un scalaire. De fait, $c_0 = 1$. Après simplification, on obtient une complexité asymptotique en $O(n)$. Le parti pris de raffiner jusqu'à obtenir un système de dimension 1 est purement théorique. Dans la pratique, on stoppe le déraffinement bien plus tôt. La résolution grossière, bien que non linéaire, reste négligeable par rapport au coût des itérations de lisseur.

1.6.3 Aspects liés au parallélisme

Dans la définition de la méthode multigrille, il n'est nullement fait allusion au parallélisme. Du fait de la modularité de l'algorithme (différents lisseurs, résolution grossière au choix de l'utilisateur, etc.), on peut obtenir facilement une méthode parallèle. En prenant Jacobi comme lisseur, il est alors évident que toutes les étapes se situant au dessus du niveau le plus grossier sont parallèles. En ce qui concerne le niveau grossier, toute méthode de résolution parallèle est utilisable.

Pour avoir une méthode parallèle performante, il faut garantir un équilibrage des charges. Si l'on pousse le concept multigrille jusqu'à obtenir un système de dimension 1 sur le niveau grossier, alors l'équilibrage de charge est mauvais : tous les processeurs sauf un sont en attente. Sans aller jusqu'à cette extrémité et pour garantir une efficacité parallèle et un bon équilibrage de charge, il faut que tous les processeurs aient suffisamment de travail sur le niveau grossier. Cela implique une dimension minimum au système que l'on traite. La taille de ce système est déterminée par la méthode de résolution utilisée sur le niveau grossier.

Chapitre 2

Modélisation électromagnétique

Le but de la modélisation des phénomènes électromagnétiques est de calculer le comportement d'objets soumis à une onde électromagnétique incidente. Lorsqu'un objet est "éclairé" par une onde électromagnétique, cette onde peut être soit diffractée, réfléchie, ou absorbée par l'objet. Le développement d'objets furtifs et de radars nécessite une connaissance approfondie de ces phénomènes. Les simulations réalisées dans ce contexte consistent alors à calculer la Surface Équivalente Radar (SER) des objets étudiés.

Les phénomènes électromagnétiques sont régis par les équations de Maxwell et leurs lois constitutives. La simulation de ces phénomènes nécessite la représentation discrète des champs électriques et magnétiques aux alentours des objets étudiés. Dans le cas d'objets de grande taille, cette simulation nécessite une décomposition de domaine. Cette décomposition permet de découper le problème global en plusieurs problèmes de petite taille moins coûteux à résoudre.

Dans cette partie, nous présenterons les équations de Maxwell dans le cadre général en posant les équations de comportement et les lois constitutives. Nous nous placerons ensuite dans le cadre harmonique où nous établirons ces mêmes équations et lois. Une fois le problème posé, la discrétisation des équations par les éléments finis de Nédélec sera introduite. Finalement, la méthode de décomposition de domaine utilisée dans le code de calcul électromagnétique du CEA sera présentée.

2.1 Équations de Maxwell

2.1.1 Équations de Maxwell temporelles

Un champ électromagnétique est généré par la présence et le déplacement de charges électriques. Il est décrit par quatre fonctions associant un vecteur à tout point de l'espace $x \in \mathbb{R}^3$ et à tout instant $t \in \mathbb{R}$. Nous noterons respectivement $\vec{E}(x,t)$, $\vec{H}(x,t)$ les champs électriques et magnétiques et $\vec{D}(x,t)$, $\vec{B}(x,t)$ les inductions électriques et magnétiques. Pour plus de clarté dans les notations et ne pas surcharger le discours, toutes les fonctions vectorielles seront notées $E = \vec{E}(x,t)$. Dans le système international (SI), E est exprimé en Volt par mètre, H en Ampère par mètre, D en Coulomb par mètre carré, et B en Weber par mètre carré.

En plus de ces fonctions, d'autres grandeurs physiques entrent en jeu : la densité volumique

de charge ρ (exprimée en Coulomb par mètre cube) et la densité de courant $\mathcal{J}$ (Ampère par mètre carré). La densité de charge ρ, à l'inverse de toutes les autres grandeurs physiques précédentes, est une fonction scalaire. En utilisant ces notations, les équations de Maxwell modélisant les phénomènes électromagnétiques se présentent sous la forme suivante :

$$\nabla \times E + \frac{\partial B}{\partial t} = 0, \tag{2.1a}$$

$$\nabla \times H - \frac{\partial D}{\partial t} = \mathcal{J}, \tag{2.1b}$$

$$\nabla \cdot D = \rho, \tag{2.1c}$$

$$\nabla \cdot B = 0. \tag{2.1d}$$

La loi d'induction de Faraday (2.1a) relie le champ électrique à l'induction magnétique. L'équation (2.1b) représente la loi de Maxwell-Ampère. La loi de Gauss (2.1c) relie l'induction électrique (ou flux électrique) à la densité de charge. L'équation (2.1d) est la loi de Gauss magnétique et impose un divergence nulle à l'induction magnétique. Dans le contexte qui nous intéresse, l'absence de charges ($\rho = 0$) dans le milieu impose une divergence nulle au champ D. Comme seulement trois de ces équations sont indépendantes, nous n'utiliserons par la suite que les lois de Faraday (2.1a), de Maxwell-Ampère (2.1b) et la loi de Gauss électrique (2.1c).

2.1.2 Équations de Maxwell en régime harmonique

Considérons désormais que l'on veuille étudier le comportement électromagnétique d'un objet soumis à une onde de fréquence unique f et de pulsation $\omega = 2\pi f$. Cette étude en régime harmonique se fait en considérant que tous les champs ont une dépendance en $e^{-i\omega t}$. Ainsi, pour toute fonction temporelle K, on exprimera $K(x,t) = e^{-i\omega t}\, K(x)$ avec $i = \sqrt{-1}$.

En régime harmonique, les fonctions considérées ne dépendent plus que de la position x, contrairement à leur expression temporelle. En les substituant dans les équations (2.1a), (2.1b) et (2.1c), on obtient les équations de Maxwell en régime harmonique :

$$\nabla \times E - i\omega B = 0, \tag{2.2a}$$

$$\nabla \times H + i\omega D = \mathcal{J}, \tag{2.2b}$$

$$\nabla \cdot D = \rho. \tag{2.2c}$$

Cette modification est due au fait que les radars émettent des champs qui ont cette dépendance temporelle. Elle a deux avantages : d'une part les fonctions ne dépendent plus de la variable t, et d'autre part les différentielles partielles en temps se simplifient. À partir des champs harmoniques, il est possible de retrouver les champs dans le domaine temporel en les multipliant par $e^{-i\omega t}$.

2.1.3 Lois constitutives

En plus des équations de comportement décrites précédemment, des lois relient les inductions aux champs. Ces relations, appelées *lois constitutives*, introduisent les notions de permittivité électrique ε, de perméabilité magnétique μ et de conductivité σ d'un matériau. Elles

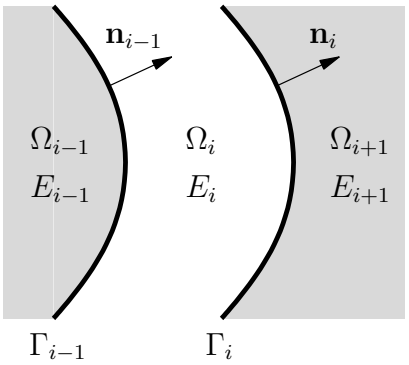


FIGURE 2.1: Couplage entre domaines non homogènes.

s'expriment :

$$D = \varepsilon E, \tag{2.3a}$$
$$B = \mu H, \tag{2.3b}$$
$$\mathcal{J} = \sigma E. \tag{2.3c}$$

Dans le vide, la permittivité et la perméabilité sont des constantes notées ε_0 et μ_0. En notant c la vitesse de la lumière dans le vide, on a $\sqrt{\varepsilon_0\mu_0} = \frac{1}{c}$. On appelle *conducteur* tout matériau de conductivité $\sigma > 0$. Les matériaux avec $\sigma = 0$ et $\varepsilon \neq \varepsilon_0$ seront appelés *diélectriques sans pertes*. L'introduction des lois constitutives dans les équations de Maxwell permet d'éliminer les termes d'induction D et B. Les relations (2.2a) et (2.2b) s'écrivent alors :

$$\nabla \times E - i\omega\mu H = 0, \tag{2.4a}$$
$$\nabla \times H + i\omega(\varepsilon + \frac{i\sigma}{\omega})E = 0. \tag{2.4b}$$

On introduit ε_r et μ_r, les permittivités et perméabilités relatives du matériau, que l'on obtient par les normalisations $\varepsilon + \frac{i\sigma}{\omega} = \varepsilon_r\varepsilon_0$ et $\mu = \mu_r\mu_0$. Dans la suite de cette thèse, nous nous intéresserons uniquement au champ électrique. On exprime alors H par (2.4a) que l'on introduit dans (2.4b) pour obtenir l'équation du champ électrique :

$$\nabla \times (\frac{1}{\mu_r}\nabla \times E) - \kappa_0^2\varepsilon_r E = 0, \tag{2.5}$$

avec $\kappa_0 = \omega\sqrt{\varepsilon_0\mu_0}$ le nombre d'ondes dans le vide. On peut aussi l'exprimer $\kappa_0 = \omega/c$. On appelle nombre d'ondes dans le matériau $\kappa = \kappa_0\sqrt{\varepsilon_r\mu_r} = \kappa_0 n_r$ où n_r est l'*indice* relatif du matériau. Quand κ augmente, les systèmes linéaires obtenus par la discrétisation de (2.5) deviennent non définis positifs. Ce sera l'une des difficultés essentielles de cette étude.

2.1.4 Conditions limites

Interface entre deux matériaux

Les équations précédentes supposaient des champs continus sur des domaines homogènes. Dans les cas où l'on a plusieurs matériaux, la continuité des champs entre les domaines

homogènes n'est pas assurée. Il faut alors poser des conditions de transmission sur les frontières des domaines. Supposons le domaine global $\Omega = \Omega_1 \cup \Omega_2$ avec Γ_1 la frontière entre les domaines Ω_1 et Ω_2 et $\mathbf{n}_1$ la normale à Γ_1, sortante à Ω_1 (Fig. 2.1). On a alors les relations suivantes sur la surface pour les champs électriques :

$$\mathbf{n}_1 \times (E_1 - E_2) = 0, \tag{2.6a}$$
$$\mathbf{n}_1 \cdot (D_1 - D_2) = \rho_s. \tag{2.6b}$$

Dans ce cas ρ_s représente la densité de charge sur la surface Γ_1. La première de ces relations (2.6a), montre la continuité de la partie tangentielle du champ électrique au niveau de la surface. La seconde relation montre que lorsqu'il y a un saut d'indices de permittivité entre deux domaines, la composante normale du champ électrique n'est pas continue. En utilisant (2.3a), le fait que la densité de charge ρ_s soit nulle et l'expression des composantes normales et tangentielles du champ $E = E^n + E^t$, l'équation (2.6b) devient :

$$\mathbf{n}_1 \cdot (\varepsilon_{r,1} E_1^n - \varepsilon_{r,2} E_2^n) = 0. \tag{2.7}$$

qui implique, lorsque les deux domaines ont une permittivité différente, une discontinuité de la composante normale du champ électrique.

Cas du conducteur parfait

Un matériau est appelé conducteur parfait lorsque $\sigma = \infty$. Selon la loi d'Ohm (2.3c), et avec comme hypothèse une densité de courant J bornée, alors E tend vers 0. Dans ce cas, ces conditions limites s'expriment :

$$\mathbf{n} \times E = 0, \tag{2.8}$$
$$\mathbf{n} \cdot B = 0, \tag{2.9}$$

avec E et B les champs du domaine extérieur au conducteur parfait et $\mathbf{n}$ la normale à sa surface. Les deux autres conditions limites s'écrivent :

$$\mathbf{n} \cdot D = \rho_S, \tag{2.10}$$
$$\mathbf{n} \times H = J_S. \tag{2.11}$$

Conditions de transmission et d'impédance

Dans le cadre d'une décomposition de domaine (dont nous parlerons plus tard), il est nécessaire de disposer de fonctions de passage entre deux domaines non homogènes. En considérant un nouvelle fois le schéma Fig. 2.1, on définit l'opérateur suivant pour le domaine Ω_i :

$$T_i^{\pm}(E_{\Omega_i}) = \pm \mathbf{n} \times (\frac{1}{\mu_r} \nabla \times E_{\Omega_i}) - \frac{i\kappa_0}{z} E_{\Omega_i}^{tg}, \tag{2.12}$$

avec $E_{\Omega_i}^{tg} = \mathbf{n} \times (E_{\Omega_i} \times \mathbf{n})$ la composante tangentielle du champ défini sur Ω_i, $z = Z_0 \sqrt{\mu_r / \varepsilon_r}$ l'impédance du matériau et Z_0 l'impédance caractéristique du vide. Cet opérateur est défini pour tout domaine i, et le signe $\pm$ résulte de la convention utilisée pour la normale à la surface.

On utilisera le signe − pour les flux entrants par la frontière interne, et + pour les flux sortants par la frontière externe (éloignée de l'objet). On définit la condition de transmission (CT) pour tous domaines Ω_i et Ω_j adjacents :

$$T_i^{\pm}(E_{\Omega_i}) = T_i^{\pm}(E_{\Omega_j}). \tag{2.13}$$

Note : l'indice i indique pour quel sous-domaine l'opérateur $T^{\pm}$ est défini.

Lors de la décomposition de domaine, cette condition sera très souvent utilisée : les conditions limites de chaque sous-domaine seront calculées par cet opérateur, en fonction des solutions des domaines adjacents.

Le second type de condition limite, la condition d'impédance (CI), concerne la surface du corps Γ_0. Sur cette surface, on a la relation suivante :

$$\mathbf{n} \times (\frac{1}{\mu_r} \nabla \times E_{\Omega_1}) = -\frac{i\kappa_0}{z} E_{\Omega_1}^{tg}. \tag{2.14}$$

En utilisant la précédente définition de l'opérateur $T^{\pm}$, cela équivaut à dire :

$$T_1^{-} E_{\Omega_1} = 0. \tag{2.15}$$

2.2 Résolution des équations de Maxwell

Nous disposons désormais des équations de Maxwell harmoniques. Nous nous intéresserons plus particulièrement au calcul du champ électrique E par l'équation du second ordre :

$$\nabla \times (\frac{1}{\mu_r} \nabla \times E) - \kappa_0^2 \varepsilon_r E = 0. \tag{2.16}$$

Afin de résoudre ce problème, nous allons tout d'abord discrétiser cette équation par la méthode des éléments finis. Cette discrétisation nécessite la mise en place d'un cadre fonctionnel afin de représenter au mieux les opérateurs différentiels continus sur le domaine discret. Nous présenterons dans ce chapitre les éléments d'arêtes de Nédélec [40, 41], qui nous serviront à représenter de façon discrète les équations de Maxwell. Ensuite, nous introduirons les concepts de décomposition de domaine (DDM), méthode permettant de découper le problème global en un ensemble de sous problèmes.

2.2.1 Décomposition de domaine et itération globale

Pour faciliter la résolution de l'équation (2.24), qui exprime le champ électrique sur l'intégralité du domaine d'étude, on considère une décomposition en sous-domaines concentriques (figure 2.2). La Méthode de Décomposition de Domaine (DDM) permet de manipuler des sous-domaines de tailles inférieures au domaine global, sur lesquels les résolutions sont moins coûteuses. En contrepartie, la solution globale est calculée par une seconde résolution utilisant les solutions de chacun des sous-domaines. Le choix fait au CEA est de résoudre par une méthode directe les problèmes dans chaque sous-domaine, et de résoudre le système global à l'aide d'une méthode itérative de type Gauss-Seidel par blocs : à l'itération k, le champ

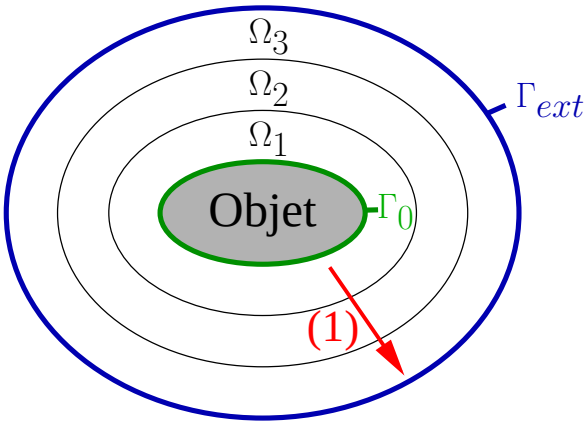


FIGURE 2.2: Décomposition de domaine avec sens de parcours de l'itération globale (1).

électrique du sous-domaine Ω_i est calculé à partir du champ dans Ω_i à l'itération précédente et des champs des sous-domaines voisins Ω_{i-1} et Ω_{i+1}. Avec i balayant les sous-domaines de Ω_1 (intérieur) à Ω_n (extérieur), l'*itération globale* s'écrit :

$$E_{\Omega_i}^{[k+1]} = f(E_{\Omega_{i-1}}^{[k+1]}, E_{\Omega_i}^{[k]}, E_{\Omega_{i+1}}^{[k]}). \tag{2.17}$$

Après plusieurs itérations, le champ global converge vers la solution. La relation (2.12) permet alors de transmettre la valeur du champ d'un sous-domaine à l'autre. On distingue trois types de couplages :
– le couplage volumique-volumique entre deux sous-domaines adjacents ;
– le couplage entre un sous-domaine et la surface de l'objet Γ_0 ;
– le couplage entre un sous-domaine et la surface externe du domaine global Γ_{ext}.
En considérant les surfaces Γ_0 et Γ_{ext} comme des sous-domaines à part entière, on parlera de sous-domaines surfaciques et donc de couplages volumiques-surfaciques.

Couplage volumique-volumique : condition de transmission

Le calcul du champ électrique dans le sous-domaine volumique Ω_i à l'itération k (2.17) se base sur les champs dans les sous-domaines adjacents Ω_{i-1} et Ω_{i+1}. Lorsque ces sous-domaines sont volumiques (ni surface du corps, ni surface externe du domaine global), le couplage utilise la condition de transmission (2.12). On peut alors exprimer les relations suivantes :

$$T_i^+ E_{\Omega_i} = T_i^+ E_{\Omega_{i+1}}^{[k]}, \tag{2.18}$$

$$T_i^- E_{\Omega_i} = T_i^- E_{\Omega_{i-1}}^{[k+1]}. \tag{2.19}$$

Les indices d'itérations utilisés ici correspondent au balayage classique, représenté par la flèche (1) dans la figure 2.2. Les valeurs des champs aux frontières ne sont pas imposées par ces conditions et sont relaxées lors de la résolution globale.

Couplage volumique-surfacique : surface de l'objet

Le sous-domaine Ω_1 est un cas particulier : il est couplé avec le sous-domaine surfacique Γ_0 représentant la surface du corps étudié. Dans ce cas, la condition d'impédance (2.15) est utilisée et se substitue à la condition de transmission (2.19) :

$$T_1^- E_{\Omega_1} = T_1^- E_{\Gamma_0} = 0. \tag{2.20}$$

Contrairement aux conditions de transmissions, cette condition n'est pas relaxée lors de l'itération globale : le champ est alors imposé sur la surface de l'objet.

Couplage volumique-surfacique : champ à l'infini et les EID

Pour calculer la SER de l'objet simulé, il faut considérer le champ électrique à l'infini. En pratique, le domaine est tronqué et le champ infini représenté sur la surface externe du domaine global Γ_{ext}. Le calcul du champ lointain est un problème à part entière, résolu par une méthode spécifique. Le problème à l'infini est régi par les équations intégrales (EI) dont la discrétisation donne un système linéaire dense. La résolution de ce problème est effectuée soit par un solveur direct (Scalapack [5]), soit par un solveur itératif accéléré par une méthode *Fast Multipole* (FMM). Pour accélérer la convergence de cette méthode itérative, une formulation particulière des EI est utilisée : les équations intégrales de Després [15] (EID). Contrairement aux EI classiques, la discrétisation des EID donne un système linéaire bien conditionné garantissant une convergence rapide de la méthode itérative.

Malgré cette différence de résolution avec les systèmes volumiques, le couplage entre le sous-domaine volumique externe (Ω_3 dans la figure 2.2) et la surface Γ_{ext} se fait de la même manière que les transmissions volumique-volumique. Si l'on dispose de n sous-domaines, avec Ω_n le plus externe d'entre eux, alors (2.18) devient :

$$T_n^+ E_{\Omega_n} = T_n^+ E_{\Gamma_{ext}}^{[k]}. \tag{2.21}$$

Tout comme la condition de transmission volumique-volumique, cette condition est relaxée lors de l'itération globale

Remarque concernant les conditions limites

Il faut noter que la condition d'impédance est une condition limite de type Dirichlet (non relaxée). La façon dont cette CL est imposée au niveau de la résolution du système linéaire dépend du code de calcul. En considérant le système linéaire $Ax = b$, deux solutions existent pour imposer la valeur v_{cl} à une arête i sur la surface :
– mettre $b_i = v_{cl}$, $a_{ii} = 1$ et $\forall k \neq i : a_{ik} = a_{ki} = 0$;
– ou alors choisir une très grande valeur g et mettre $b_i = g.v_{cl}$ et $a_{ii} = g$.
Contrairement à la première solution, la technique de la grande valeur ne modifie qu'un seul coefficient de la matrice. Dans le cadre d'une résolution parallèle avec matrice distribuée, cette modification locale est un avantage : seul le processus possédant le terme diagonal a_{ii} doit imposer la CL dans la matrice. Par contre, cette modification détériore le conditionnement de la matrice car g devient une valeur propre.

2.2.2 Discrétisation par éléments finis

Formulation faible

Pour obtenir une solution numérique au problème (2.16), nous allons passer par une formulation faible. En considérant un ensemble de fonctions tests $\mathcal{T}$, on définit l'égalité au sens faible de deux fonctions f et g notée $\overset{w}{=}$ (w pour *weak*) :

$$f \overset{w}{=} g \;\Leftrightarrow\; \forall \phi \in \mathcal{T} :\; (f, \phi) = (g, \phi), \tag{2.22}$$

avec $(.,.)$ dénotant un produit scalaire.

Formulation faible des équations de Maxwell

On peut exprimer l'équation de Maxwell (2.16) au sens faible, pour toute fonction test $\phi \in \mathcal{T}$:

$$(\nabla \times (\frac{1}{\mu_r} \nabla \times E), \phi) - (\kappa_0^2 \varepsilon_r E, \phi) = 0. \tag{2.23}$$

En intégrant par parties le premier terme de la relation (2.23), on obtient :

$$(\mu_r^{-1} \nabla \times E, \nabla \times \phi) - (\kappa_0^2 \varepsilon_r E, \phi) = -\int_\Gamma \phi \cdot (\mathbf{n} \times (\mu_r^{-1} \nabla \times E))\, \mathrm{d}S. \tag{2.24}$$

Discrétisation des équations de Maxwell

Après avoir défini toutes les conditions limites que l'on peut rencontrer, on peut transformer (2.24) en exprimant $\mathbf{n} \times (\mu_r^{-1} \nabla \times E_{\Omega_i})$ de la sorte (par (2.12)) :

$$\mathbf{n} \times (\mu_r^{-1} \nabla \times E_{\Omega_i}) = \begin{cases} +T_i^+ E_{\Omega_i} + \frac{i\kappa_0}{z} E_{\Omega_i}^{tg}, \\ -T_i^- E_{\Omega_i} - \frac{i\kappa_0}{z} E_{\Omega_i}^{tg}. \end{cases} \tag{2.25}$$

En exprimant les différents termes de (2.24) relativement au sous-domaine Ω_i de frontières Γ_{i-1} et Γ_i, respectivement frontières interne et externe :

$$(\frac{1}{\mu_r} \nabla \times E_{\Omega_i}, \nabla \times \phi) - (\kappa_0^2 \varepsilon_r E_{\Omega_i}, \phi) + \int_\Gamma \phi \cdot (\frac{i\kappa_0}{z} E_{\Omega_i}^{tg}) = \int_{\Gamma_{i-1}} \phi \cdot T_i^- E_{\Omega_{i-1}}^{[k+1]} - \int_{\Gamma_i} \phi \cdot T_i^+ E_{\Omega_{i+1}}^{[k]}, \tag{2.26}$$

avec $\Gamma = \Gamma_i \cup \Gamma_{i-1}$. Selon le type des sous-domaines adjacents, les opérateurs de transmission T du second membre sont remplacés par les conditions limites présentées précédemment.

En posant $\mathcal{B}$ un ensemble de fonctions de base et $\psi_i \in \mathcal{B}$ la fonction de base associée au degré de liberté (DDL) i, la discrétisation du champ électrique s'exprime :

$$E(x) = \sum_i \alpha_i . \psi_i(x), \tag{2.27}$$

avec α_i la valeur (scalaire) du DDL i. En introduisant cette expression dans l'équation (2.26) et en choisissant $\mathcal{T} = \mathcal{B}$, on obtient la discrétisation par éléments finis (Galerkin) du problème. Pour les espaces fonctionnels, le code de calcul du CEA utilise les éléments finis de Nédélec.

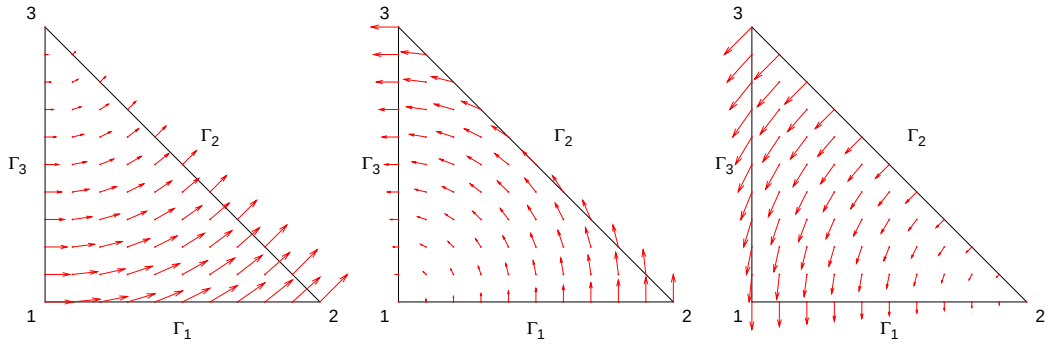


FIGURE 2.3: Fonctions de base pour les éléments de Nédélec de premier ordre en 2D.

2.2.3 Éléments finis de Nédélec

Maintenant que nous avons la formulation faible du problème, il faut choisir un ensemble de fonctions de test (qui par la méthode de Galerkin sera aussi l'ensemble des fonctions de base). Ces fonctions de base se doivent de respecter une certaine continuité entre les éléments. La continuité tangentielle, mise en évidence pour les champs électriques, doit être imposée entre chaque élément par les éléments finis. Pour ces raisons, le code de calcul utilise les éléments finis de Nédélec (de premier ordre) pour discrétiser le problème. Ces éléments associent à chaque arête Γ_i la valeur scalaire α_i définie par :

$$\alpha_i = \int_{\Gamma_i} E(x) \cdot \vec{\tau}_i \, \mathrm{d}\Gamma_i, \tag{2.28}$$

avec $\vec{\tau}_i$ le vecteur unitaire porté par l'arête Γ_i. Dans un domaine discrétisé avec n degrés de liberté, la solution approchée de l'équation (2.16), notée $\bar{E}(x)$, est calculée par la relation :

$$E(x) \simeq \bar{E}(x) = \sum_{i=1}^{n} \alpha_i \vec{p}_i(x), \tag{2.29}$$

avec p_i la fonction de base associée à l'arête Γ_i. En portant (2.29) dans (2.28), on obtient :

$$\alpha_i = \sum_{j=1}^{n} \alpha_j \int_{\Gamma_i} (\vec{p}_j(x) \cdot \vec{\tau}_i) \, \mathrm{d}\Gamma_i. \tag{2.30}$$

Pour toute fonction de base p_j associée au degré de liberté j et pour toute arête Γ_i portant le degré de liberté i, il en suit :

$$\int_{\Gamma_i} (\vec{p}_j(x) \cdot \vec{\tau}_i) \, \mathrm{d}\Gamma_i = \begin{cases} 0 \text{ si } j \neq i \\ 1 \text{ si } i = j \end{cases} \tag{2.31}$$

Convention des tétraèdres

Maintenant que le comportement des fonctions de base est défini, on considère que les p_i sont des fonctions vectorielles à composantes linéaires. En utilisant les conventions de numérotation

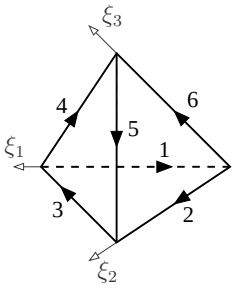


FIGURE 2.4: Convention de numérotation des arêtes du tétraèdre de référence.

et d'orientation des arêtes pour un élément de référence présenté dans la figure 2.4, on obtient les fonctions de base $\hat{p}_i(\xi_1, \xi_2, \xi_3)$ pour l'élément de référence :

$$\hat{p}_1 = \begin{pmatrix} -1+\xi_2+\xi_3 \\ -\xi_1 \\ -\xi_1 \end{pmatrix}, \qquad \hat{p}_2 = \begin{pmatrix} \xi_2 \\ 1-\xi_1-\xi_3 \\ \xi_2 \end{pmatrix},$$

$$\hat{p}_3 = \begin{pmatrix} \xi_2 \\ -\xi_1 \\ 0 \end{pmatrix}, \qquad \hat{p}_4 = \begin{pmatrix} -\xi_3 \\ 0 \\ \xi_1 \end{pmatrix},$$

$$\hat{p}_5 = \begin{pmatrix} 0 \\ \xi_3 \\ -\xi_2 \end{pmatrix}, \qquad \hat{p}_6 = \begin{pmatrix} \xi_3 \\ \xi_3 \\ 1-\xi_1-\xi_2 \end{pmatrix}.$$

L'utilisation de ces fonctions de base – définies ci-dessus pour l'élément de référence – pour un élément quelconque nécessite le calcul de la matrice de transformation B. Cet opérateur de passage du tétraèdre de référence vers le domaine de calcul est une matrice 3×3. En notant $\hat{K}$ le tétraèdre unitaire de référence et K un tétraèdre quelconque de l'espace, on définit $B : \hat{K} \to K$ la relation de passage entre le tétraèdre de référence et le tétraèdre quelconque :

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = B \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \tag{2.32}$$

En utilisant cette notation de changement de base, on définit le changement de base pour les fonctions vectorielles $\hat{p}$:

$$p(x, y, z) = (B^{\mathsf{T}})^{-1}\, \hat{p}(\xi_1, \xi_2, \xi_3). \tag{2.33}$$

Avec la relation suivante pour le rotationnel [39] :

$$\mathrm{rot}(p) = \frac{1}{det(B)}\, B\, \hat{\mathrm{rot}}(\hat{p}). \tag{2.34}$$

Finalement, en utilisant les éléments finis de Nédélec dans la formulation variationnelle des équations des Maxwell (2.26), on obtient le système linéaire $Ax = b$. Du fait des fonctions utilisées, la matrice est creuse, complexe et symétrique (non hermitienne). Le calcul du champ dans chaque sous-domaine volumique se fait par la résolution de ce système.

Chapitre 3

Positionnement

3.1 Problématique globale

3.1.1 Problèmes de grande dimension et passage à l'échelle

La taille des objets simulés impose l'utilisation de maillages plus ou moins volumineux. Les études actuellement menées au CEA concernent des objets 3D de taille importante requérant des maillages composés de millions de mailles. Les systèmes linéaires comptant des dizaines de millions d'inconnues sont fréquents et, dans un futur proche, il faudra envisager la résolution de systèmes de plusieurs milliards d'inconnues. La méthode directe utilisée pour les résolutions volumiques induit des coûts CPU et mémoire très importants. Pour chaque domaine de la décomposition de domaine (DDM) c'est à la fois le système linéaire original et sa factorisation qui doivent être stockés. Pour les domaines de grandes tailles, le remplissage induit par la factorisation rend le stockage de ces données difficilement réalisable sur les machines actuelles d'autant plus que la tendance pour les machines futures est de réduire la quantité de mémoire par cœur. Une solution consiste à réduire la taille des systèmes en augmentant le nombre de sous-domaines de la DDM. Dans ce cas, la méthode qui repose sur une itération globale de type Gauss-Seidel converge plus lentement. De plus, certaines contraintes physiques imposent une taille minimum aux sous-domaines. Une seconde solution consistant à utiliser plus de processeurs pour la résolution directe arrive elle aussi à ses limites : les méthodes directes parallèles nécessitent un ratio calcul/communication suffisant pour être performantes. Plus on distribue le système, plus les communications augmentent, rendant la méthode directe peu scalable sur des milliers de processeurs. Pour les systèmes linéaires de grande taille et pour les méthodes massivement parallèles l'utilisation des méthodes directes est donc remise en question. Avec l'arrivée des machines massivement parallèles comportant plusieurs centaines de milliers de processeurs il est nécessaire de disposer de solveurs hautement parallèles. Malgré leur robustesse, leur prévisibilité et la qualité des solutions calculées, les solveurs directs ont du mal à passer à l'échelle lorsqu'il s'agit de milliers de cœurs de calcul. Le remplaçant de la méthode directe devra donc permettre l'utilisation d'un très grand nombre de cœurs.

Dans l'optique de remplacer les méthodes directes pour les problèmes volumiques de très grande taille, l'utilisation des méthodes itératives a été étudiée. Ces méthodes ont plusieurs avantages : faible consommation mémoire, complexité plus faible et bonne scalabilité en général. Pour la plupart des méthodes itératives, le surcoût mémoire nécessaire à la résolution

est négligeable en comparaison de la taille du système linéaire. De plus, les calculs effectués consistent principalement en des opérations algébriques simples (produits matrice-vecteur, produits scalaires). Malgré tous ces avantages, les méthodes itératives se sont avérées peu performantes. Le conditionnement et le fait que certaines matrices ne soient plus définies positives pour nos problèmes de Maxwell rendent la convergence souvent très lente. Dans des cas extrêmes ces méthodes itératives divergent [35, 45].

3.1.2 Résolutions "hautes" fréquences

Outre la taille de l'objet étudié, la fréquence a elle aussi une grande influence sur la nature du maillage : pour une simulation précise et répondant aux besoins du CEA, un nombre suffisant de mailles par longueur d'onde (NMLO) est nécessaire [46]. Cette contrainte permet d'éviter tout phénomène d'*aliasing*. En électromagnétisme, il est d'usage d'utiliser au moins 10 mailles par longueur d'onde. La longueur d'onde étant inversement proportionnelle à la fréquence, plus la fréquence est élevée, plus le maillage doit être fin.

Dans le cadre des études menées au CEA, les objets sont testés sur de larges gammes de fréquences. Il est alors nécessaire de fournir un maillage suffisant pour chacune de ces fréquences. Deux solutions s'offrent aux utilisateurs : fournir un maillage par fréquence ou un unique maillage suffisamment fin pour la fréquence la plus élevée. Dans le premier cas, le nombre de maillages est important et nécessite un travail conséquent en amont de la simulation (génération des maillages et prétraitements). La seconde solution n'est pas meilleure pour autant : toutes les fréquences, même les plus faibles, seront calculées sur un maillage très fin. En plus du mauvais ratio calculs réalisés / calculs nécessaires, l'utilisation d'un maillage trop fin peut impliquer des systèmes inutilement mal conditionnés et difficiles à résoudre.

Idéalement, la solution proposée devra se baser sur un maillage *minimal* représentant correctement la géométrie de l'objet. Le maillage utilisé lors des calculs et respectant la contrainte du NMLO sera généré automatiquement par raffinements successifs. Ce raffinement automatique permettra aux utilisateurs de calculer de larges gammes de fréquences en ne fournissant qu'un seul maillage qui sera le plus grossier.

3.2 Méthode multigrille et équations de Maxwell

De tout ce qui a été présenté précédemment, il en ressort que la méthode proposée devra avoir les caractéristiques suivantes :

- traiter des problèmes de très grande taille ;
- permettre le raffinement automatique des maillages ;
- passer à l'échelle sur des dizaines de milliers de cœurs ;
- être suffisamment robuste pour être utilisée en production.

Le "mauvais" comportement des méthodes itératives sur cette gamme de problèmes et la question du passage à l'échelle des méthodes directes nous amène à considérer une solution hybride. La solution envisagée, combinant méthodes directe et itérative, sera basée sur le modèle des méthodes multigrilles, et plus particulièrement sur la méthode du full-multigrid (FMG). À partir d'un maillage grossier représentant de manière précise la géométrie de l'objet cette méthode calculera par raffinements automatiques successifs des maillages suffisamment

fins pour les problèmes hautes fréquences. Dans un soucis de robustesse, nous comptons conserver le solveur direct parallèle pour effectuer les résolutions sur le niveau grossier. Cette méthode directe sur le maillage grossier, point d'entrée de l'algorithme multigrille, servira à distribuer le système ; nous réutiliserons donc la distribution des données du solveur direct parallèle PaStiX tout au long de la résolution sur les niveaux fins. Un raffinement homogène du domaine garantira la conservation de l'équilibrage de charge du niveau grossier : si les inconnues sont réparties correctement sur le niveau grossier, elles le seront aussi sur les niveaux fins.

Pour un gain mémoire optimal, nous utiliserons des méthodes *matrix-free* pour toutes les résolutions sur les niveaux fins. Nous nommerons "méthode *matrix-free*" toute méthode de résolution effectuée sans former (assembler) la matrice globale et s'appuyant donc uniquement sur les matrices élémentaires (éléments finis). À partir de ces choix, une étude bibliographique nous a permis de sélectionner les différents composants de la méthode développée dans cette thèse.

3.2.1 Bibliographie : méthode multigrille pour Maxwell et Helmholtz

Dans la littérature, de nombreux articles traitent de la résolution des équations de Maxwell et de Helmholtz par les méthodes multigrilles. Ces deux opérateurs présentent plusieurs difficultés particulières pour les méthodes multigrilles et plus particulièrement les méthodes itératives :
- matrices non définies positives (sauf pour de très petites valeurs du nombre d'onde k) ;
- grande taille de "near null space". Bien que souvent cité pour Maxwell, ce problème est aussi présent pour Helmholtz ;
- matrices à coefficients complexes.

Du fait de ces difficultés, de nombreuses méthodes de résolution ont été étudiées. La majorité d'entre elles concerne le multigrille algébrique et propose des opérateurs inter-niveaux spécifiques, des lisseurs, ou des décompositions de l'opérateur. Bien que nous souhaitions développer une méthode multigrille géométrique, les remarques, problèmes et solutions apportés dans le cadre algébrique sont tout de même intéressants.

Méthodes multigrilles dans le cadre général

La principale source d'informations sur les méthodes multigrilles (algébriques et géométriques) utilisée dans cette thèse est tirée du livre "*A Multigrid Tutorial*" de Briggs, Henson et McCormick [8]. D'autres livres décrivent en détails la méthode multigrille et sont aussi de bons points de départ à toute personne désirant mieux comprendre les méthodes multigrilles. Parmi ceux ci, on peut citer les livres "*Multigrid*" de Trottenberg, Oosterlee et Schüller (consultable en ligne) et le livre "*Multigrid methods and applications*" de Hackbusch [26]. Pour des informations plus spécifiques aux méthodes multigrilles appliquées aux équations de Maxwell, le livre Zhu et Cangellaris [57] peut être, lui aussi, une référence.

Multigrille algébrique pour matrices à coefficients complexes

Un des principaux problèmes de la méthode multigrille algébrique est le calcul des niveaux fins et grossiers. Le critère habituel de ces méthodes repose sur les entrées de la matrice. Plus

particulièrement, on considère deux DDL i et j comme fortement connectés (*strong connection*) en se basant sur la valeur a_{ij} de la matrice. Pour les équations de Maxwell discrétisées par les éléments de Nédélec, les matrices sont complexes et symétriques (non Hermitiennes). Dans ces conditions, l'évaluation des faibles/fortes connexions est plus compliquée que dans le cas réel où une relation d'ordre existe. Dans l'article de Maclachlan et Oosterlee [36], les méthodes permettant de travailler avec des matrices complexes sont abordées. Ils proposent une technique de décomposition de la matrice en parties réelle et imaginaire. L'utilisation de cette technique permet la résolution de systèmes complexes par méthode multigrille algébrique de façon naturelle. Aucune transformation ni connaissance de la matrice ne sont nécessaires pour cela. Néanmoins, cette méthode ne peut être réalisée dans un cadre *matrix-free*.

Ce problème lié aux coefficients complexes de la matrice n'a de sens que pour les méthodes algébriques. Dans le cadre géométrique, on se basera uniquement sur la géométrie pour déterminer quelles sont les inconnues sur les niveaux fins et grossiers.

Décomposition de l'opérateur

Le principal problème des équations de Maxwell discrétisées se situe dans le noyau de l'opérateur $(\mathrm{rot}, \mathrm{rot})$: tous les champs gradients sont dans le noyau de cet opérateur. Les composantes de l'erreur contenues dans ce noyau ralentissent fortement la convergence de la méthode multigrille. Beaucoup d'articles récents traitent de ce sujet, parmi ceux-ci on peut noter les articles de Hiptmair et Xu [30, 31], l'article de Kolev et Vassilevski [33] et Hu [32], tous trois basés sur les travaux de Reitzinger [49]. Ces articles proposent de décomposer l'opérateur $(\mathrm{rot}, \mathrm{rot})$ de Maxwell en deux opérateurs : le noyau du rotationnel (contenant les gradients) et son complémentaire. Les deux parties de l'opérateur sont ensuite traitées de façon indépendante. La construction de l'espace auxiliaire (noyau) dont les inconnues sont aux nœuds se fait par l'utilisation d'un gradient discret. La résolution multigrille s'effectue ensuite principalement sur cet opérateur auxiliaire, la résolution du système principal se faisant par l'emploi d'un solveur classique (Falgout et Kolev préconisent l'utilisation de méthodes de splitting pour la résolution sur l'opérateur principal). Ensuite, les deux solutions obtenues sont combinées afin d'obtenir le résultat final. De plus, Hu [32] introduit des prolongateurs spécifiques à ce type de méthode. Dans notre cas, l'utilisation de cet espace auxiliaire semble difficile à utiliser : on peut facilement obtenir les matrices élémentaires des éléments-arêtes, par contre les éléments-nœuds utilisés dans cette méthode doivent êtres calculés en utilisant le gradient discret. Le gradient discret est facilement calculable car il repose sur la géométrie, chaque ligne de la matrice associée comportant deux entrées non nulles de valeur respective -1 et 1 selon l'orientation de l'arête. Pour une résolution *matrix-free*, cela implique pour chaque matrice élémentaire une quantité de calculs non négligeable et une connaissance de la géométrie. Nous verrons par la suite, dans la partie concernant la conception du solveur, que pour des raisons de généricité nous avons tenté de nous abstraire au maximum de la géométrie : le raffinement de la géométrie est une opération très spécifique au problème et ne peut donc pas être réalisé de façon générique. La méthode du gradient discret va alors à l'encontre de ce choix de conception.

Étude des lisseurs

Le choix du lisseur peut lui aussi s'avérer crucial pour la convergence de la méthode multigrille. Nous présentons ici un panel des propositions et études des lisseurs disponibles tant pour le cas général que pour les équations de Maxwell.

Une étude très intéressante est menée dans l'article [4] qui décrit les performances de divers lisseurs en comparant leurs taux de convergence et leur scalabilité. Parmi les lisseurs testés, on peut trouver les lisseurs polynômiaux (Chebyshev), les versions standard de Gauss-Seidel et de Jacobi et leurs versions hybrides (Block-Gauss-Seidel et Block-Jacobi). Cet article propose une amélioration des algorithmes de splitting appelée "l1-smoothers" et qui force la convergence de ces méthodes. Cette étude met en évidence la supériorité des lisseurs de type Gauss-Seidel. Les lisseurs polynomiaux, qui sont aussi préconisés par Adams [1], montrent des performances intéressantes sur les problèmes du type Maxwell et Helmholtz. Pesqué montre dans [46] que sur Helmholtz, le gain des méthodes de Chebyshev est faible par rapport à la méthode de Jacobi classique. De plus, les réglages nécessaires au bon fonctionnement de la méthode sont assez complexes. Chebyshev ne semble donc pas très utilisable dans un cadre de production.

Un autre type de lisseur, permettant de prendre en compte le noyau de l'opérateur $(\mathrm{rot}, \mathrm{rot})$ est introduit par Arnold, Falk et Winther [3] (AFW). La méthode de lissage est spécifique pour ce type de problème. Elle consiste à résoudre localement les équations. Pour chaque nœud du maillage, le système des DDL associés aux arêtes incidentes est assemblé et résolu. La valeur calculée pour chaque arête est cumulée. Comme chaque arête est incidente à deux nœuds, chacune d'entre elles est mise à jour deux fois. Cette méthode permet de capturer localement les gradients. Bien que performante, cette méthode nécessite un traitement très étroitement lié au maillage. De plus, le calcul pour chaque nœud des arêtes incidentes dans un maillage non structuré a une complexité élevée, rendant la méthode difficilement applicable dans un code 3D et sa parallélisation efficace est difficile.

Une méthode relativement proche est aussi abordée par Lee dans un papier à paraître. Tout comme la méthode AFW, elle repose sur le graphe du système. Au lieu de ne considérer que les arêtes incidentes à un nœud, un arbre couvrant composé d'arêtes est construit. Le lissage est effectué alternativement sur l'arbre couvrant puis sur son complémentaire. Cette partition des arêtes améliore le comportement de la méthode en capturant certaines composantes du noyau de l'opérateur. Dans la thèse de Boonen [6], cette méthode est succinctement décrite. Tout comme pour AFW [3], les résultats semblent intéressants, mais les traitements à effectuer sont relativement complexes et difficilement parallélisables.

Elman, Ernst et O'Leary [17] proposent d'utiliser des méthodes de Krylov (GMRES) en tant que lisseur. Leur méthode globale consiste alors en un GMRES préconditionné par une méthode multigrille utilisant un GMRES en tant que lisseur. Bien que cette méthode donne de bons résultats, l'utilisation du GMRES en tant que lisseur semble coûteuse et même si le GMRES garantit la convergence, la mise en œuvre semble bien plus complexe qu'une méthode de type Gauss-Seidel ou Jacobi. Par contre, l'idée de l'utilisation d'une méthode multigrille pour préconditionner des méthodes de Krylov est intéressante (voir 3.2.2).

La méthode de Kaczmarz nous a souvent été proposée au cours d'un séjour aux États-Unis. Cette méthode de réduction de bruit, principalement utilisée en traitement du signal, n'est malheureusement pas réalisable en *matrix-free* [52]. De plus, Pesqué a étudié cette méthode [46] qui ne s'est pas montrée particulièrement performante.

À noter aussi des travaux de Gander. Dans une présentation au CERFACS, il présente un multigrille sur Helmholtz basé sur une modification de la matrice du niveau grossier et une adaptation du nombre d'étapes du lisseur en fonction du nombre d'onde. À partir d'informations non suffisantes, Pesqué [46] a testé cette méthode sans succès. Nous n'avons pas eu le temps de tester les propositions apparues dan un texte récemment disponible sur la page web de Gander [19].

Finalement, après avoir considéré toutes ces méthodes de lissage et en ayant gardé en mémoire le fait que le lisseur devra être *matrix-free*, la méthode employée sera le *damped-Jacobi* (cf. 1.4.1). Bien que ses performances, en tant que lisseur, soient inférieures à celles d'un Gauss-Seidel, son potentiel parallèle et sa simplicité en font un candidat idéal. De plus, la méthode de Jacobi est facilement implémentable en *matrix-free* car seule la diagonale nécessite d'être assemblée (voir l'équation 1.9, page 21).

3.2.2 Méthode multigrille en tant que préconditionneur

Les matrices des problèmes à forts nombres d'ondes ont tendance à devenir non définies positives. Dans ces conditions, les méthodes multigrilles ne convergent que lentement, ou divergent [46]. Une solution consiste à utiliser les méthodes multigrilles non comme un solveur, mais plutôt comme un préconditionneur. De nombreux articles traitent de cette utilisation spécifique :

- Gopalakrishnan et al. [25] étudient la méthode multigrille appliquée aux équations de Maxwell harmoniques. Elle est utilisé pour préconditionner un GMRES ;
- Chen et al. [13] utilisent la méthode multigrille pour préconditionner un MINRES dans le cadre des équations de Maxwell. Les résultats obtenus sont similaires à ceux de Gopalakrishnan [25] ;
- dans les articles [18, 55], Erlangga et al. utilisent la méthode multigrille pour préconditionner un Bi-CGSTAB.

La tendance actuelle pour les problèmes les plus difficiles est d'utiliser la méthode multigrille en tant que préconditionneur. Dans notre cas, nous avons d'abord considéré le préconditionnement d'un Gradient Conjugué (GC). Pesqué montre dans [46] que c'est le meilleur préconditionneur du GC pour les équations de Helmholtz. Malheureusement, dans notre cas la restriction n'est pas le transposé de la prolongation (itération non symétrique), ce qui dégrade fortement les performances [46]. C'est pourquoi nous avons préféré préconditionner un GMRES qui accepte un préconditionneur non symétrique contrairement au GC.

3.3 Synthèse

À la lecture de tous ces articles, il est évident que la résolution de équations de Maxwell par une méthode multigrille est complexe. Cette complexité est principalement issue de l'opérateur et des éléments de Nédélec utilisés. La solution choisie consiste à réaliser un full-multigrid, permettant le raffinement automatique des maillages.

Sur le niveau grossier le solveur direct est employé pilotant le reste de la résolution. La distribution calculée sur ce niveau est conservée sur les niveaux fins, garantissant un équilibrage de charges. Cette distribution, calculée pour limiter les communications lors de la résolution directe, permettra aussi de limiter les communications lors des résolutions sur les niveaux

fins.

Les étapes de lissage consistent en quelques itérations de damped-Jacobi en mode *matrix-free*. Pour chaque niveau fin, la matrice utilisée (non assemblée) correspond à la rediscrétisation du problème sur ce niveau. Cette rediscrétisation nous permet d'avoir recours à des opérateurs de restriction et de prolongation simples et garantit la cohérence entre la matrice et la physique simulée. Rappelons de plus que le problème dont nous attendons la solution est celui sur la grille fine : sa matrice ne saurait être déduite de celle sur le niveau grossier par une méthode de Galerkin.

Les interpolations d'un niveau grossier vers un niveau fin se baseront sur les fonctions de base des éléments de Nédélec. L'opération inverse transférant un vecteur du niveau fin vers le niveau grossier consistera en une injection des valeurs fines vers les valeurs grossières correspondantes.

Finalement, pour les problèmes que l'on ne peut résoudre à l'aide d'une FMG seule, une méthode GMRES préconditionnée est employée. Elle se base sur un code GMRES en *reverse communication* développé par Frayssé, Giraud, Gratton et Langou [23]. La méthode de *reverse communication* employée par ce code implique que toutes les opérations algébriques nécessaires à la résolution sont effectuées par l'utilisateur du GMRES et non par le solveur. En utilisant un produit matrice-vecteur de type *matrix-free*, la méthode GMRES est alors elle aussi *matrix-free*.

Deuxième partie

Contribution au domaine

Chapitre 4

Conception du solveur

Dans cette partie, nous nous intéresserons à la conception du solveur. Les différents modules constituant le solveur multigrille seront progressivement introduits. Nous commencerons par étudier l'existant en décrivant succinctement les données et fonctionnalités fournies par le code de calcul 3D. Nous verrons ensuite comment est conçu le solveur multigrille. Une fois les deux codes mis en jeu présentés, nous décrirons finalement la méthode de couplage utilisée.

4.1 Étude de l'existant

Dans cette partie, nous étudierons l'environnement des codes dans lequel va devoir être intégré le solveur multigrille. Il s'agit principalement du Code de Calcul 3D (CC3D) du CEA avec lequel le solveur multigrille devra interagir et le solveur direct parallèle PaStiX, qui va gouverner le parallélisme de la méthode. Bien que nous ayons décidé de concevoir un solveur multigrille géométrique **généraliste** applicable à une large gamme de problèmes, nous nous focalisons ici sur les problèmes électromagnétiques et plus particulièrement la résolution des équations de Maxwell. Ce problème spécifique nous permettra de déterminer les structures de données et les outils logiciels nécessaires à la résolution de problèmes de type éléments finis. Cette étude nous a permis de mieux concevoir le solveur multigrille, tout en garantissant un couplage performant avec le CC3D.

4.1.1 Le code électromagnétique

Présentation générale

Le solveur développé devra s'intégrer dans le code électromagnétique du CEA. Il permet de calculer la SER d'un objet éclairé par une onde électromagnétique. Les calculs peuvent être réalisés pour toute une gamme de fréquences, angles d'incidence et angles d'observation. Les données en entrée sont :

- un maillage par sous domaine ;
- un fichier "physique" par domaine. On y trouve les types de conditions limites, les matériaux et toutes les informations physiques relatives au domaine ;
- un fichier "pilote" qui décrit la résolution dans sa globalité : solveurs employés, type de

calcul, fréquences, angles d'incidence et d'observation, etc.

Notre solveur est destiné à effectuer les résolutions sur chacun des sous-domaines. Les principales informations qui nous intéressent sont le maillage et la physique du sous-domaine courant. Par la suite, nous allons voir comment est gérée la géométrie et comment sont calculés les matrices élémentaires et les seconds membres. Nous verrons aussi l'application des conditions limites.

Géométrie centralisée

Le maillage du sous-domaine courant est lu par le code de calcul. Bien que la résolution soit parallèle, tous les processus prenant part au calcul ont une connaissance globale de la géométrie. Quatre principaux composants définissent la géométrie : les nœuds, les arêtes, les faces, et les tétraèdres. Chacun de ces composants est identifié de manière unique par un entier. La figure 4.1 représente la structure de données employée par le code de calcul.

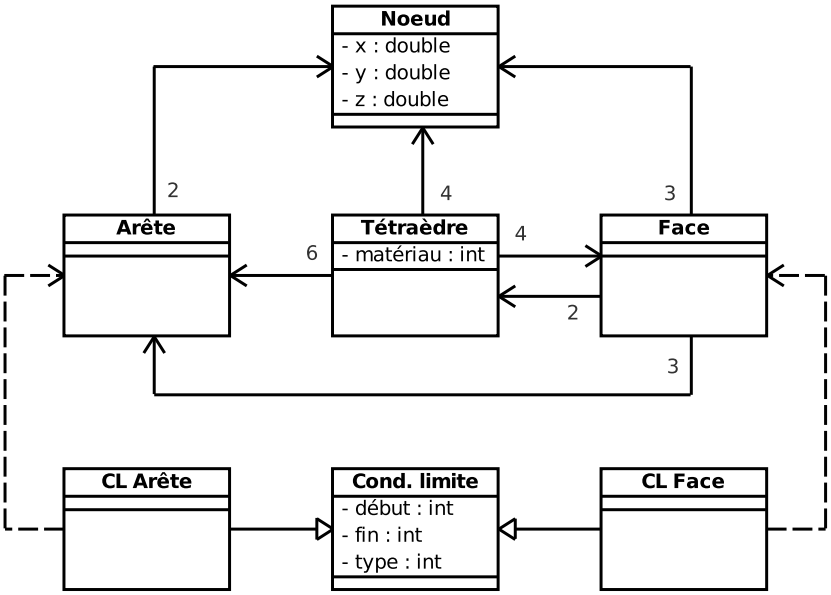


FIGURE 4.1: Stockage de la géométrie et des conditions limites.

Il ressort de ce graphique que les informations stockées concernent principalement les relations entre les différents types de composants. Toutes les relations de compositions sont réalisées par le stockage d'indices. On peut remarquer deux types de redondances : les directes et les transitives. La redondance directe face-tétraèdre est une astuce de conception permettant de limiter la quantité de calculs : certains traitements sur les faces ont besoin de connaître les tétraèdres adjacents. Plutôt que de chercher parmi les tétraèdres ceux contenant la face en question, il a été décidé de stocker cette information. Les redondances transitives, telles que les relations tétraèdre-face-nœud et tétraèdre-nœud n'ont pas d'utilité autre que garantir une cohérence de la géométrie. Bien que ces redondances ne soient pas indispensables, il faudra tout de même en tenir compte. Lors du raffinement, ces relations devront **impérativement** être préservées.

Les conditions limites, sont elles aussi stockées avec la géométrie. Elles ne concernent que les

arêtes et les faces. Pour faciliter les calculs, on distingue parmi les arêtes et les faces plusieurs types :

– les arêtes ou faces internes, ne faisant partie d'aucune frontière ;
– les arêtes ou faces de la frontière interne du sous-domaine ;
– les arêtes ou faces de la frontière externe du sous-domaine ;
– les arêtes ou faces d'autres frontières (entre des matériaux).

Le type 1 n'est concerné par aucune condition limite. Les types 2 et 3 sont les composants sur lesquels sont effectués le couplage de la DDM. La convention utilisée pour la numérotation est de donner des identifiants contigus aux arêtes (faces) d'un même type. Lors du raffinement il est alors impératif d'identifier le type de chaque nouvelle arête et face pour les numéroter correctement.

Calcul des matrices élémentaires

Le calcul des matrices élémentaires se fait de façon relativement simple. Une fonction, prenant en paramètre un numéro de tétraèdre renvoie la matrice élémentaire. Le calcul effectué repose sur la géométrie précédemment introduite. Lors du développement du solveur multigrille, il faudra alors faire attention à ce que le code de calcul dispose de la géométrie en accord avec le niveau de raffinement considéré. Le passage de la numérotation des éléments finis à celle de la géométrie doit être direct : par exemple, le lien entre une inconnue et une arête doit se calculer en temps constant. Les correspondances élément-tétraèdre et inconnue-arête doivent alors être garanties. Cette dualité éléments finis - géométrie est indispensable.

Calcul des conditions limites

Les conditions limites sont représentées sous forme matricielle. On parlera alors de matrices élémentaires de conditions limites. Pour un algorithme *matrix-free*, cette façon d'appliquer les conditions limites est idéale : l'application des conditions limites de type Dirichlet font souvent appel à une méthode de mise à l'identité de la partie de la matrice concernée. En *matrix-free*, ce type de traitement est difficilement réalisable. D'un autre côté, la méthode employée par le code de calcul, mettant une grande valeur sur la diagonale et laissant les lignes et colonnes intactes nous simplifie les choses. La modification de la matrice se fait localement à la matrice élémentaire de conditions limites, sans altérer les matrices élémentaires "voisines".

Calcul des seconds membres

Le calcul des seconds membres est réalisé de manière globale : tous les processus ont le même second membre, chacun sélectionnant la partie de vecteur relative aux inconnues locales. Pour calculer le second membre, chaque processus doit connaître la géométrie globale (ou tout du moins la géométrie des surfaces externes et internes du sous-domaine). Pour un solveur multigrille, cette nécessité de la géométrie globale est un problème : après plusieurs raffinements la géométrie globale peut être très volumineuse et ne peut alors pas être stockée localement. Pour pouvoir résoudre ce problème de façon pérenne, il faudrait légèrement modifier le calcul des seconds membres pour récupérer les seconds membres élémentaires, leur assemblage étant réalisé localement à chaque processeur.

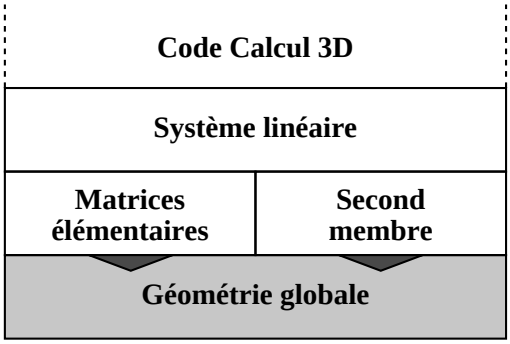


FIGURE 4.2: Schéma de masse du code de calcul. Les fonctionnalités fournies reposent toutes sur la géométrie globale.

Conception du code de calcul : synthèse

Tout ce qui a été dit précédemment permet d'établir le schéma global du code de calcul présenté dans la figure 4.2. La géométrie globale est à la base de tous les traitements réalisés. Toutes les fonctions de calcul se basent sur cette géométrie. De plus, le code est développé en Fortran, avec beaucoup de données globales (la géométrie en faisant partie). Lors du couplage, il faut faire attention à ne pas laisser en mémoire de données relatives à la géométrie globale, potentiellement source de bug (données à la fois locales et globales). La méthode que l'on va employer consiste à traiter les géométries raffinées de la même manière que la géométrie globale initiale : un fichier de maillage fin est généré, pour ensuite être chargé par le code de calcul lors du traitement de ce niveau. Ainsi, on se repose sur le code déjà existant pour garantir la cohérence des données. Cette solution permettra un couplage plus sûr, mais ne pourra pas rester tel quel pour un code de production : le passage d'informations entre le multigrille et le code de calcul par des fichiers de maillage n'est pas une solution performante qui passe à l'échelle et devra être remplacée.

4.1.2 Le solveur direct parallèle PaStiX

Comme indiqué dans le chapitre 3, nous avons décidé d'exploiter un solveur parallèle direct performant sur le niveau grossier. Le solveur direct parallèle PaStiX [28], utilisé actuellement dans le code de calcul, a été sélectionné.

Les factorisations et résolutions directes parallèles se décomposent en cinq étapes :

1. la renumérotation, pour limiter le remplissage dans les facteurs ;
2. la factorisation symbolique, pour calculer la structure de la matrice factorisée ;
3. la redistribution, pour équilibrer la charge de travail entre les processus ;
4. la factorisation numérique ;
5. la résolution par descente-remontée sur les facteurs calculés à l'étape 4.

Chacune de ces opérations a un champ d'action bien délimité. Certaines d'entre elles travaillent sur la structure de la matrice (les indices non-zéros), alors que d'autres travaillent

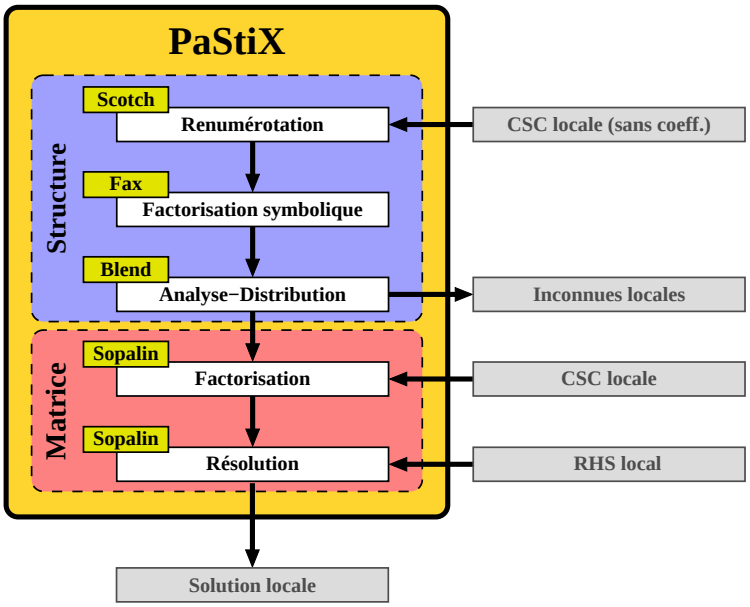


FIGURE 4.3: Schéma de masse du solveur direct parallèle PaStiX.

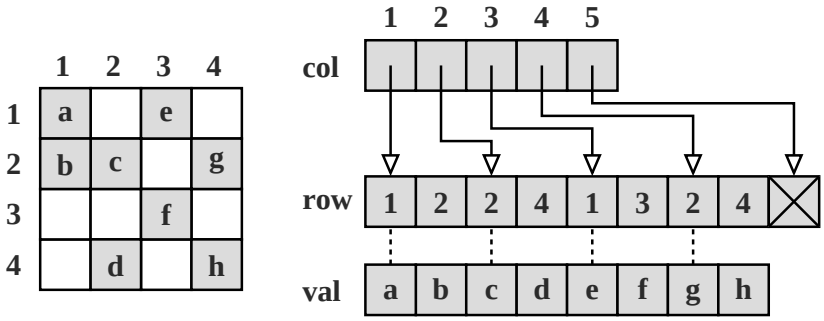


FIGURE 4.4: Représentation CSC d'une matrice creuse.

sur les valeurs de la matrice. Le schéma de masse de la figure 4.3 présente l'enchaînement des opérations, les données d'entrée-sortie de chaque étape et les noms des codes qui définissent ces opérations élémentaires.

Renumérotation

La renumérotation consiste à calculer de nouveaux indices pour les inconnues. Cette opération vise à limiter le remplissage de la matrice lors de la factorisation. Elle est réalisée en travaillant sur le graphe des non-zéros de la matrice, aussi appelé *graphe d'adjacence*. Le partitionneur de graphe parallèle employé pour cette étape est le logiciel Scotch, ou plus particulièrement PT-Scotch pour sa version parallèle [44].

Le graphe d'adjacence en entrée de Scotch considère une matrice au format CSC (Compressed Sparse Column) sans les coefficients (figure 4.4). La structure de la CSC dépend directement des éléments finis utilisés : le coefficient a_{ij} de la matrice est non-nul si et seulement si les inconnues i et j appartiennent à un même élément. Le calcul du graphe d'adjacence global

nécessite un parcours de tous les éléments. Par la suite, le calcul de la structure sera nommé *pré-assemblage* de la matrice.

PT-Scotch travaille avec un graphe distribué. Tout comme le graphe d'adjacence global, il se présente sous la forme d'une sous-matrice au format CSC sans coefficients par processus. En plus du graphe distribué, l'utilisateur fournit un tableau d'indices permettant de passer de la numérotation locale des inconnues vers leur numérotation globale (appelé par la suite `l2g`, de l'anglais "*Local to Global*"). À partir du graphe distribué, PT-Scotch calcule en parallèle une partition des inconnues. Le résultat de cette opération est l'arbre d'élimination associé à la matrice. À partir de cet arbre, il est possible de calculer la nouvelle numérotation des inconnues minimisant le remplissage.

Contrairement au graphe global, le calcul du graphe local ne nécessite que le parcours des éléments contenant les inconnues locales. Le choix de la distribution initiale des inconnues est laissé à l'utilisateur. Pour p processus, nous avons divisé les n inconnues en p groupes de taille n/p et d'indices contigus. Chaque groupe est ensuite attribué à un processus de PT-Scotch.

Factorisation symbolique

Une fois la renumérotation et l'arbre d'élimination calculés, une "simulation" de factorisation, la *factorisation symbolique*, est effectuée. Cette étape permet de calculer la structure par blocs de la matrice factorisée à partir de l'arbre d'élimination renuméroté. La structure de la matrice factorisée permet de prévoir la quantité de non-zéros du système factorisé et donc la mémoire nécessaire au stockage de la matrice. La factorisation symbolique fait aussi apparaître les dépendances existant entre les blocs, permettant de déterminer le schéma d'exécution de la factorisation.

Analyse et distribution

Les bonnes performances de la factorisation de PaStiX sont liées à l'adéquation entre le partitionnement et la distribution des données sur les nœuds de calcul. La distribution se base sur la structure par blocs obtenue avec la factorisation symbolique par blocs. Cette étape, issue des travaux de thèse de Pascal Hénon [27], calcule de manière statique une régulation équilibrée pour le solveur. Le calcul de la distribution, prenant en compte le plan d'exécution et un modèle de coût [48], fait en sorte que la charge entre les nœuds soit équilibrée. La distribution prend aussi en compte les différents niveaux de parallélisme mis en évidence par l'arbre d'élimination pour effectuer la répartition des inconnues :

- parallélisme à gros grain lié à l'indépendance des calculs entre les sous-arbres de l'arbre d'élimination (blocs denses non couplés, MPI) ;
- parallélisme à grain moyen lié à la parallélisation de la factorisation des blocs denses ;
- parallélisme à grain fin qui concerne les optimisations parallèles locales (pipelines des processeurs superscalaires). Ce parallélisme est fortement lié à la taille des blocs mis à jour par des primitives BLAS-3 [16, 34].

La distribution calculée lors de cette étape est une distribution statique (*static mapping*). Une fois que le plan d'exécution et la distribution ont été établis, aucune modification n'est apportée à la distribution des données. Cela permet d'avoir un comportement reproductible de la méthode directe, et permet de garantir un haut niveau de performances.

Après cette étape, chaque processus prenant part à la factorisation dispose de la liste des inconnues locales. Cette liste est analogue au tableau `l2g` fourni en entrée de la renumérotation parallèle. À partir de ce tableau, soit l'utilisateur fournit la CSC distribuée comme calculée par cette étape, soit la CSC dans la distribution initiale. Dans ce dernier cas, la redistribution de la matrice est effectuée par le solveur PaStiX de manière transparente.

Il faut noter que les structures manipulées jusque là (graphe et factorisation symbolique) ne comportent pas de coefficients. Le coût mémoire de ces étapes est alors très inférieur à celui de la factorisation. Ces étapes de partitionnement, renumérotation, analyse et distribution forment un prétraitement à la méthode directe. Dans le cas où plusieurs matrices de même structure doivent être factorisées, le résultat de ces étapes peut être réutilisé.

Factorisation

La factorisation parallèle repose sur le plan d'exécution calculé lors des étapes d'analyse et de distribution. L'ordre des opérations à effectuer est directement lié à l'arbre d'élimination calculé lors du partitionnement, et ne peut être modifié. Ce déterminisme dans la méthode de calcul fait que les étapes précédentes sont considérées comme des phases de prétraitement. Le parallélisme, mis en évidence dans l'arbre d'élimination, est exploité par l'utilisation de MPI et des threads. Le MPI (*Messsage Passing Interface*) permet aux nœuds de communiquer entre eux alors que les threads permettent un parallélisme au sein de chaque nœud. Pour le parallélisme à grain fin, PaStiX fait appel aux routines BLAS-3 et Lapack [2]. L'utilisation conjointe de ces techniques permet à PaStiX d'exploiter au mieux les performances de la machine utilisée.

La matrice à factoriser est donnée à PaStiX sous forme d'une CSC avec coefficients utilisant soit la distribution initiale, soit la distribution calculée. Afin de ne pas modifier la matrice utilisateur, et comme la structure de la factorisation diffère, PaStiX stocke en interne la matrice symétrisée. Cette copie implique que les matrices utilisateur et factorisée sont stockées simultanément. Le coût de stockage des facteurs dépend de la qualité de la renumérotation, mais dépasse largement la taille de la matrice initiale. Toutefois, une fois la factorisation effectuée, la matrice utilisateur peut être libérée si aucune étape de raffinement itératif n'est nécessaire (ce qui est notre cas).

Résolution

Une fois la factorisation réalisée, la résolution ne nécessite qu'un vecteur. En parallèle, le vecteur doit être fourni avec la même distribution que la matrice lors de l'étape précédente. Tout comme la factorisation, la résolution est elle aussi déterministe et se base principalement sur l'arbre d'élimination calculé plus tôt.

Si plusieurs résolutions doivent être réalisées, deux solutions sont possibles : soit fournir tous les seconds membres et utiliser la résolution multi seconds membres, soit faire appel à la descente-remontée pour chaque vecteur. La résolution multi seconds membres a l'avantage d'utiliser des primitives BLAS-3 et Lapack, accélérant les calculs. Le résultat est calculé *in-place*, le vecteur d'entrée contient alors la solution.

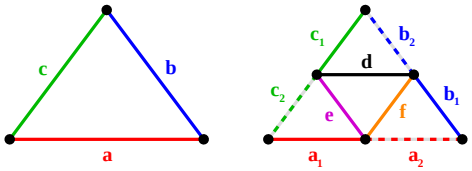


FIGURE 4.5: Création-disparition de DDL lors du raffinement d'un élément-arête 2D.

4.2 Conception du solveur

Bien que la méthode développée soit un multigrille géométrique, la connaissance du maillage et tous les traitements s'y rapportant sont liés au problème physique et à sa discrétisation. Nous désirons fournir une solution applicable à tout type d'élément fini et tout type de géométrie. Dans le cas actuel, il est évident que la géométrie et les éléments finis de Nédélec sont assez particuliers. Classiquement, avec des DDL aux nœuds, un raffinement fait apparaître des inconnues alors qu'un déraffinement en fait disparaître. Avec les éléments de Nédélec, certains raffinements peuvent faire disparaître des inconnues (figure 4.5) : une inconnue de la grille grossière disparaît au profit de deux DDL sur la grille fine. Ce comportement prouve qu'il est impossible de fournir un solveur totalement générique en se basant sur la géométrie. Les éléments finis sont le seul dénominateur commun et se basent uniquement sur une version "allégée" de la géométrie. Que ce soient des DDL-arêtes, DDL-nœuds ou tout autre type de DDL, un élément est toujours défini de la même manière et ne fait intervenir que les identifiants des DDL. Le solveur développé, à défaut de connaître exactement la géométrie, s'appuiera sur la définition des éléments finis.

De ce constat, la méthode de multigrille géométrique ne manipulera jamais de géométrie à proprement parler, mais plutôt des éléments finis. Alors que la géométrie stocke les nœuds, arêtes, et faces pour chaque tétraèdre, la définition d'un élément fini de Nédélec se limite à une liste de 6 indices : les 6 arêtes constituant le tétraèdre. Lorsque l'on aura besoin de calculer la matrice élémentaire d'un élément quelconque, il suffira de fournir l'indice de cet élément qui l'identifie de façon unique dans la géométrie. Le CC3D permet de calculer les coefficients de la matrice élémentaire à partir de la géométrie de l'élément (tétraèdre).

Le couplage entre le code de calcul et le solveur se fera donc par l'intermédiaire de la relation éléments finis-géométrie. La géométrie étant déléguée au CC3D, il devra raffiner et déraffiner un maillage. Il en va de même pour la prolongation et la restriction. En résumé, le code multigrille est un "chef d'orchestre", comptant sur le CC3D pour réaliser toutes les opérations nécessaires à son bon fonctionnement. Ce type de mécanisme ressemble fortement aux approches de *reverse-communication*, où le solveur ne fait qu'ordonner les tâches. Dans notre cas, nous nous reposons plutôt sur une mécanique de *callbacks*. A l'initialisation du solveur, nous lui fournissons les différentes fonctions dont il aura besoin : calcul de matrices élémentaires, restriction d'un vecteur, prolongation d'un vecteur, (dé-)raffinement. Dès qu'une de ces opérations est nécessaire, le solveur multigrille fera appel à ces routines. Toutes les opérations algébriques seront quant à elles réalisées en interne par le solveur. Cette approche permet d'exprimer une certaine généricité et indépendance du solveur MG vis à vis du type d'élément et de la géométrie.

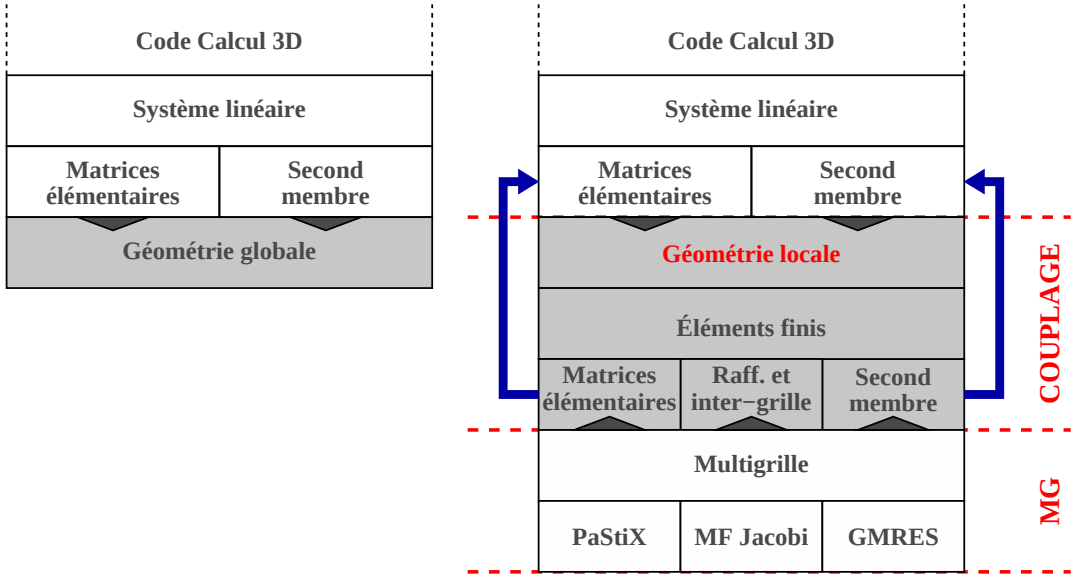


FIGURE 4.6: Couplage du solveur multigrille et du code de calcul 3D. Le schéma de gauche représente le code de calcul dans son état initial. Le schéma de droite montre le couplage du CC3D et du solveur multigrille. Les parties "purement" couplage et multigrille sont mises en évidence. Les flèches bleues représentent les appels implicites de fonctions.

4.3 Couplage multigrille-code de calcul

Dans les parties précédentes, nous avons mis en évidence l'importance du mécanisme de couplage. Le couplage, développé spécifiquement pour le code de calcul du CEA, permet de faire l'interface entre le multigrille et le code de calcul. D'un côté, il fournit au solveur multigrille les interfaces vers les fonctions de calcul, et d'un autre côté il permet de "piloter" le code de calcul. Dans notre cas, le pilotage est essentiel : le code dans lequel nous nous intégrons n'a pas été conçu pour être utilisé en tant que fournisseur de services, comme nous voulons l'employer ici. Sa conception nous a obligé à l'adapter pour qu'il puisse calculer ce que nous voulons. Pour les matrices élémentaires, le CC3D se base sur sa géométrie globale interne. Dans notre cas, les calculs doivent se baser sur les maillages fins générés. Il est alors nécessaire de substituer notre maillage raffiné au maillage interne du CC3D. Une fois la substitution effectuée, le code de calcul peut calculer les matrices élémentaires voulues. Contrairement à son maillage initial, les maillages nécessaires au calcul des matrices élémentaires sont des maillages locaux (sous ensemble du maillage global contenant les arêtes locales). Au lieu d'utiliser un maillage global raffiné, le code de calcul n'a besoin que du maillage local raffiné. Pour des niveaux de maillages très raffinés, le gain en mémoire est important.

Toutefois, cette substitution a un coût : le code de calcul a besoin d'une géométrie globale pour certains calculs. Par exemple, le calcul du second membre nécessite la géométrie globale. Le problème du second membre ne peut être contourné simplement car il nécessite une modification du CC3D. Heureusement, il nous sera possible de calculer les seconds membres fins à partir du second membre grossier, sans modification du CC3D.

Les interactions et dépendances entre les codes de calcul, de couplage et multigrille sont complexes. Dans la figure 4.7, le cheminement de la méthode ainsi que les dépendances sont

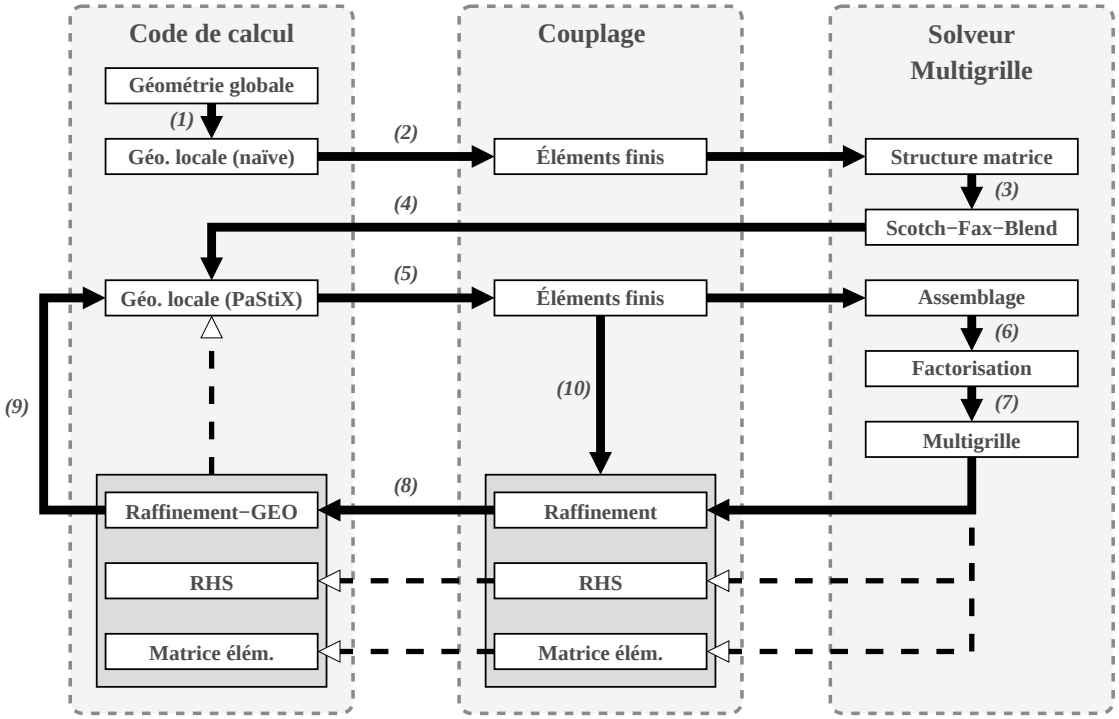


FIGURE 4.7: Ordre d'exécution et dépendances des composants logiciels. Les flèches solides représentent l'ordre d'exécution, les flèches pointillées représentent les dépendances.

mises en évidence. À partir de la géométrie globale, la géométrie locale à chaque processus est calculée (1). Et de cette géométrie locale, les éléments finis sont initialisés (2).

Une fois ces étapes effectuées, le solveur multigrille est exécuté. La première étape du solveur consiste à calculer la structure de la matrice, en se basant sur les éléments finis locaux. Cette structure (pattern) est ensuite passée au prétraitement de la méthode directe nommée ici Scotch-Fax-Blend (3). L'étape de prétraitement calcule une distribution de la matrice. Chaque inconnue du système linéaire étant liée à une arête, il est possible de reconstruire une géométrie locale à partir des inconnues locales calculées lors du prétraitement (4). Cette relation entre la distribution des inconnues et la distribution de la géométrie est un point central de la méthode. C'est par cette relation que le solveur direct pilote la distribution de la méthode itérative. De façon analogue à l'étape (2), on initialise à partir de la géométrie locale les éléments finis (5), cette fois-ci en adéquation avec la distribution du solveur direct.

Le solveur multigrille pilote alors la factorisation du système linéaire. À cette fin, la matrice est assemblée de manière parallèle, en se basant sur les éléments finis distribués, puis factorisée (6). Le système factorisé permet alors de calculer l'estimation initiale de la solution sur le niveau grossier, point de départ de la méthode full-multigrid (7).

Lors de son exécution, la méthode multigrille repose sur les opérations fournies par le couplage : calcul des matrices élémentaires, calcul du second membre, et les opérations intergrilles. Lorsque le solveur multigrille demande au couplage de (dé)raffiner, le (dé)raffinement de la géométrie est déclenché (8). Le raffinement géométrique met à jour la géométrie locale du code de calcul (9), qui à son tour met à jour les éléments finis (5). La modification de la géométrie locale implique une modification implicite des éléments finis (10) et, par extension,

du système linéaire.

Le cycle logique composé des étapes (8), (9), (5) et (10) est la base de toute la méthode multigrille. Toutefois, dans la pratique il s'avère que l'initialisation des éléments finis à partir de la géométrie (5) est une opération relativement coûteuse. Pour éviter de la répéter à chaque changement de niveau, on distingue deux opérations distinctes : le raffinement des éléments finis et le raffinement de la géométrie. Il est ensuite fait en sorte que ces deux raffinements conservent la dualité éléments finis-géométrie.

Chapitre 5

Réalisation

Le “squelette” du solveur étant désormais défini (voir chapitre 4), nous allons détailler les principaux composants de la méthode multigrille. Dans un premier temps, l'organisation algorithmique du solveur multigrille sera présentée, avec ses différents types de cycles. Nous présenterons ensuite plus en détails chacune des opérations associées à un cycle. Les points abordés ici concernent à la fois le prétraitement du côté du code de calcul, le couplage et le solveur multigrille. Globalement, le solveur fait appel aux différents outils présentés dans ce chapitre, tout en se basant sur les algorithmes multigrilles eux aussi présentés ici.

5.1 Algorithmique du multigrille

Le solveur multigrille a été développé comme un enchaînement de tâches élémentaires. Parmi celles-ci se trouvent le lisseur, les opérations inter-grilles, la résolution directe. Le solveur en lui même ne fait qu'enchaîner ces étapes. Cette partie, plutôt que de détailler les outils et les astuces de codage, présente le solveur multigrille comme un groupe de composants logiciels qu'il pilote.

5.1.1 Le cycle en V

Le cycle en V est à la base de tout algorithme multigrille. Sa décomposition algorithmique est représentée dans la figure 5.1. La présentation faite ici ne repose pas sur la version récursive de la définition du cycle en V. Ce choix est principalement dû au coût des algorithmes récursifs, mais il permet également de mieux gérer les allocations mémoire : réutilisation de buffers, pré-allocations, etc.

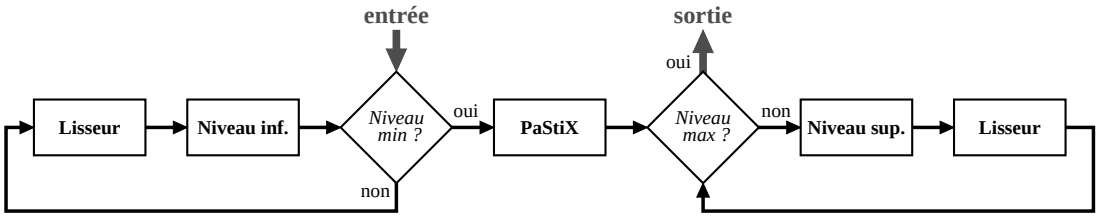


FIGURE 5.1: Schéma algorithmique du cycle en V.

Dans cette figure, on distingue deux niveaux importants :
– le niveau *minimum*, dénotant le niveau le plus grossier ;
– le niveau *maximum*, représentant le niveau d'entrée du cycle en V.
On définit la *profondeur* du V-cycle par la soustraction de ces deux grandeurs. Dans cette figure, le test "niveau min" sert à détecter le niveau grossier, sur lequel est effectuée la résolution directe. Pour sa part, le test de "niveau max" détecte le retour sur le niveau d'entrée, qui est aussi le point de sortie. Par exemple, pour un cycle de profondeur 0, où le niveau d'entrée est le niveau grossier, le traitement se limite à la résolution directe. Dans tous les autres cas, les opérations de lissage et de transferts inter-grilles sont utilisées. L'avantage de cette méthode est de fournir une solution, quelle que soit la profondeur du cycle demandée. Les opérations inter-grilles "niveau sup." et "niveau inf." de ce schéma regroupent respectivement le raffinement-prolongation et le déraffinement-restriction et seront présentées plus en détails par la suite.

Pour un même maillage d'entrée, il est possible d'enchaîner les cycles en V, en utilisant la sortie de l'un comme entrée de l'autre. Comme le niveau grossier est constant d'un V-cycle à l'autre (du fait d'un raffinement déterministe), les systèmes linéaires associés à ces deux itérations sont identiques. Pour tirer avantage de ce comportement, la factorisation du système grossier n'est réalisée qu'une seule et unique fois. Toutes les résolutions directes sur la grille grossière ne seront alors que des descentes remontées, de complexité $n^{4/3}$.

5.1.2 Le full-multigrid

Contrairement au cycle en V où le niveau d'entrée désigne le niveau de sortie, le full-multigrid nécessite de connaître le niveau de sortie (maximum). De façon analogue à la profondeur d'un cycle en V, le niveau maximum désigne la *hauteur* du FMG. C'est sur ce niveau qu'est calculé le résultat final. La figure 5.2 représente l'enchaînement algorithmique des étapes. Le niveau d'entrée est ici le niveau 0. La première étape de V-cycle est alors de profondeur 0, se limitant à une résolution directe (voir figure 5.1). La solution grossière est ensuite prolongée pour servir d'estimation initiale pour le système sur le niveau supérieur. Dans le cas où le niveau maximum est le niveau grossier (aucun raffinement demandé), l'étape préliminaire de résolution directe permet de retourner tout de même la solution. L'étape dénotée "V-cycle" dans ce schéma est en réalité composée d'un enchaînement de V-cycles. Si la convergence d'un seul V-cycle n'est pas suffisante, on enchaîne alors sur un autre, jusqu'à ce que la qualité de la solution obtenue soit suffisante.

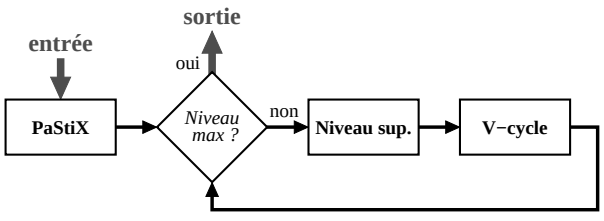


FIGURE 5.2: Schéma algorithmique du cycle full-multigrid.

Pour des raisons pratiques, nous avons décidé de "court-circuiter" les V-cycles intermédiaires. Comme précédemment, le V-cycle d'entrée dénote en réalité une résolution directe sur grille grossière (V-cycle de profondeur 0). L'estimation grossière est ensuite directement prolongée

jusqu'au niveau maximum (fin) pour servir en entrée d'une résolution par V-cycle. Cette modification algorithmique est présentée dans la figure 5.3. Pour un problème donné, la méthode classique et celle court-circuitée convergent de la même manière. On en déduit que, pour des hauteurs de FMG raisonnables, la prolongation de l'estimation grossière initiale est suffisante pour être utilisée comme estimation initiale fine (même sans étapes de lissage intermédiaires).

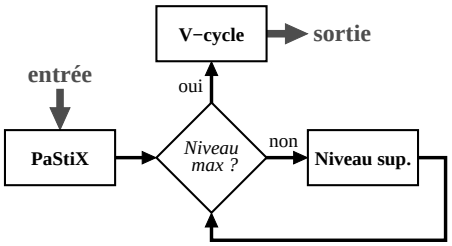


FIGURE 5.3: Schéma algorithmique du cycle full-multigrid modifié.

5.2 Distribution parallèle de la géométrie

Puisque les éléments finis que nous manipulons sont liés aux arêtes, la distribution de la géométrie est assujettie à la distribution des arêtes. A l'instar de la géométrie présentée dans la section 4.1.1, la géométrie locale (contenant uniquement les composantes géométriques utiles au calcul local) est composée de nœuds, arêtes, faces et tétraèdres. À ces informations est ajoutée la notion de localité des composants. Dans le cas présent, c'est la localité des arêtes (inconnues) qui induit la localité des tétraèdres et faces. Les nœuds seront toujours considérés comme des données locales. Un tétraèdre ou une face sont stockés localement si au moins une de leurs arêtes est locale. On fait une distinction entre les tétraèdres purement locaux, dont toutes les arêtes sont locales, et les tétraèdres partiellement locaux. Bien que tous les tétraèdres (faces) soient stockés de la même manière, les arêtes non locales sont marquées comme étant distantes. Ce marquage permettra de distinguer les tétraèdres (faces) purement et partiellement locaux lors des différents calculs effectués. Ces arêtes non-locales intervenant dans des composants locaux seront appelées *arêtes fantômes*. Bien que non locales, elles doivent être stockées afin d'avoir une description géométrique complète.

5.2.1 Sélection des arêtes locales

Supposons que l'on dispose de na arêtes locales quelconques. La première étape de chargement de la géométrie distribuée consiste à charger ces données en mémoire. Depuis le maillage global, seules les na arêtes locales sont lues et se voient attribuer un identifiant local. La méthode implémentée implique que deux arêtes a et b d'indices globaux $ga < gb$ ont des indices locaux respectifs la et lb de telle sorte que $la < lb$. En effet, alors que le passage d'un indice local à un indice global se fait par l'intermédiaire du tableau de renumérotation `aretes_l2g`, l'inverse est plus complexe. En ayant trié ces indices de la sorte, le calcul d'un indice local à partir de l'indice global se résume à une recherche par dichotomie dans le tableau `aretes_l2g`. Non seulement cette dichotomie nous donne l'indice local mais permet aussi de détecter les arêtes non locales lorsque la recherche échoue. Nous verrons par la suite que le

passage de l'indice global à l'indice local est une opération importante et doit être optimisé en temps de calcul.

5.2.2 Sélection des composants géométriques locaux

À partir des arêtes locales, il est nécessaire de reconstruire la géométrie adjacente à ces arêtes : tout composant géométrique (tétraèdre, face) contenant au moins une arête locale sera sélectionné et ajouté à la géométrie locale. Le traitement est alors relativement simple : pour tous les tétraèdres et faces, on ne sélectionne que ceux dont une arête est locale. Puisque l'on se base sur la géométrie globale, les indices des arêtes composant les tétraèdres et faces sont dans la numérotation globale. La détection des arêtes locales se fait alors en utilisant la recherche par dichotomie de l'indice global dans le tableau `aretes_l2g`. Une fois détecté comme local (purement ou partiellement), le composant est ajouté à la géométrie locale. Les indices des arêtes sont convertis en indices locaux, et le composant se voit attribuer lui aussi un indice local.

5.2.3 Sélection des arêtes fantômes

Tous les tétraèdres partiellement locaux étant désormais définis, certaines arêtes de ces tétraèdres ne font néanmoins pas partie des arêtes locales. Ces arêtes non locales mais intervenant dans la géométrie locale seront appelées *fantômes*. Pour que la géométrie locale soit complète, ces arêtes fantômes doivent être détectées puis lues dans le maillage global. Tout comme les arêtes locales, elles se voient attribuer un indice local, distinct des indices d'arêtes locales. Cette distinction nous permet, lors des raffinements, de faire évoluer les deux listes de manière indépendante, sans avoir à modifier les numérotations. La convention que l'on choisit est de référencer les arêtes fantômes par des indices négatifs. L'indice de référence i_r se calcule alors à partir de l'indice i de l'arête :

$$i_r = \begin{cases} i & \text{s'il s'agit d'une arête locale,} \\ -(i+1) & \text{s'il s'agit d'une arête fantôme.} \end{cases} \tag{5.1}$$

Cette convention permet de faire rapidement la distinction entre les arêtes fantômes et locales, et ainsi de détecter facilement les tétraèdres et faces partiellement locaux.

5.2.4 Écritures sur disque

Comme il a été vu lors de la conception du solveur, le couplage entre le solveur et le multigrille nécessite d'écrire sur disque les géométries locales. Comme nous nous sommes basés sur la structure de la géométrie du code de calcul pour notre géométrie locale, la génération du maillage est quasiment directe. Toutefois, la notion d'arêtes fantômes n'existe pas dans la convention du code de calcul. Pour intégrer les fantômes au maillage, une transformation de leurs indices est nécessaire : les arêtes fantômes seront stockées en tant qu'arêtes classiques. Une arête fantôme g du maillage local sera stockée en tant qu'arête $g_{\text{NetCDF}} = nl + g$, avec nl le nombre d'arêtes locales et g_{NetCDF} l'indice de l'arête fantôme dans le fichier de maillage au format NetCDF [50]. Toutes les références aux arêtes fantômes (indices négatifs) doivent

elles aussi être transformées en utilisant cette renumérotation. Une fois ces transformations réalisées, la géométrie respecte la convention du code de calcul.

5.3 Les éléments finis

5.3.1 Structure de données

Comme présenté lors de la conception, les données réellement manipulées par le solveur multigrille sont les éléments finis. Nous avons vu que les éléments-tétraèdres et éléments-faces (CL) sont utilisés par le code de calcul. Comme ces deux types d'éléments ont des dimensions différentes, ils devront être traités différemment. Pour faire la distinction entre les différents types d'éléments, une structure de données est définie. Pour chaque type d'élément, il faut stocker le nombre d'éléments, ainsi que leurs indices. Ces indices serviront à faire le lien entre les éléments et la géométrie. Pour plus de facilité, nous avons choisi d'utiliser la même numérotation pour les éléments finis que pour la géométrie sous-jacente. Le pseudo-code de cette structure de données est présenté dans la figure 5.4.

```
struct ef_type_t {
  int   dim_elt;            /* dimension des elements      */
  int   num_elt;            /* nombre d'elements           */
  int   elt_loc[num_elt];   /* id elements                 */
  int * elt_l2g;            /* id element > id geo globale */

  int   connectivite[*][dim_elt];
};
```

FIGURE 5.4: Structure de données représentant un type d'élément fini quelconque.

Dans le code présenté, on peut remarquer que `elt_loc` et `elt_l2g` n'ont pas forcément la même taille. C'est une astuce pour ne pas dupliquer des données déjà présentes dans la géométrie. Par exemple, pour la gestion des conditions limites, toutes les faces n'y sont pas soumises. On spécifie dans `elt_loc` uniquement les faces concernées. Plutôt que de recalculer un tableau `l2g` correspondant à un sous-ensemble de faces, on utilise directement le `l2g` des faces de la géométrie. On peut alors passer de la numérotation des éléments locaux à la numérotation de la géométrie globale par la relation :

élément-face		face locale (géo)		face globale (géo)
`i`	$\longrightarrow$	`j = elt_loc[i]`	$\longrightarrow$	`k = elt_l2g[j] = face_l2g[j]`

Par cette relation, tout élément-face est lié à une face locale. La modification de la numérotation de la géométrie globale n'impacte que le tableau `face_l2g` liant les géométries. Cette méthode évite aussi la duplication des tableaux `l2g` entre éléments finis et géométrie locale, potentiellement source de bugs.

Comme plusieurs types d'éléments peuvent cohabiter (tétraèdres de Nédélec, faces de conditions limites, etc.), un cadre général doit être défini. La structure de données contenant ces informations, liées au système linéaire lui même, sera nommée `EF`. Le pseudo-code de la figure 5.5 présente cette structure.

```
struct ef {
  int                n_inconnues_loc; /* nombre inconnues locales  */
  int                n_inconnues_gst; /* nombre inconnues fantomes */
  int                n_inconnues_glo; /* nombre inconnues globales */

  int                loc_l2g[n_inconnues_loc];
  int                gst_l2g[n_inconnues_gst];

  int                n_types_ef;
  struct  ef_type_t  elements[n_types_ef];
 };
```

FIGURE 5.5: Pseudo-code pour la représentation des éléments finis.

Le choix de stocker une liste d'éléments permet de représenter le lien entre la géométrie et les éléments finis. Dans notre cas, nous avons besoin d'appliquer les conditions limites sur certaines faces uniquement. Cette solution permet de spécifier quelles faces sont concernées dans la numérotation locale des faces. Le calcul de la matrice élémentaire ne nécessite alors aucun changement de numérotation. À l'inverse, si une nouvelle numérotation avait été créée, il aurait fallu stocker un tableau de translation entre la numérotation des éléments-faces vers les faces locales. Dans ce cas, il n'y a pas de surcoût mémoire par rapport à la méthode réutilisant la numérotation existante ; par contre, le calcul des matrices élémentaires nécessite alors un changement de numérotation supplémentaire. De plus, lors de l'instantiation des éléments finis, au lieu de calculer le tableau `l2g` des éléments, il suffit de lui fournir le `l2g` des faces de la géométrie.

5.3.2 Fonctionnalités

En plus de réaliser le stockage des données, les éléments finis doivent fournir certaines fonctionnalités. Parmi celles-ci, on peut trouver le calcul de matrices élémentaires, le calcul du second membre, et le raffinement (figure 4.7). Plutôt que de réaliser une description algorithmique détaillée de ces tâches, nous en décrirons la logique.

En ce qui concerne les calculs de matrices élémentaires et de seconds membres, nous avons vu lors de la conception (chapitre 4) que ces opérations reposent sur les routines du code de calcul. La méthode de *callbacks* que l'on emploie nous oblige à stocker la routine de calcul pour chaque type d'élément. Lors de l'initialisation des EF par le code de calcul, chaque type d'élément est défini et se voit attribuer une routine de calcul. Cette routine sera stockée dans la structure `ef_type_t`, pour être réutilisée par la suite. Pour plus de robustesse, l'appel à ces routines par des composants externes n'est pas direct : avant tout appel, le module éléments finis convertit l'indice de l'élément pour lequel on désire calculer la matrice élémentaire. Cette conversion permet d'appeler le code de calcul tout en respectant ses conventions de numérotation. Pour réaliser cela, le module éléments finis s'appuie sur les différents tableaux `l2g`. Une fois calculées, les matrices élémentaires sont retournées à la routine appelante de façon transparente, sans que la conversion d'indices soit vue. De cette façon, le code de calcul et le code multigrille conservent leur convention, les éléments finis faisant le lien entre les deux.

Le module éléments finis sert aussi d'intermédiaire lors des opérations inter-grilles. Lors d'un raffinement, la géométrie mais également les éléments finis doivent être raffinés. On a donc scindé le traitement en deux : le raffinement des éléments finis et le raffinement géométrique. Bien que très semblables, ces deux opérations ne sont pas réalisées simultanément, ni par le même acteur. Idéalement, le code de calcul est en charge du raffinement géométrique, alors que le module éléments finis s'occupe du raffinement des éléments. La cohérence entre les deux raffinements est impérative. Dans la pratique ces deux opérations sont très liées : le raffinement des éléments finis implique le raffinement géométrique. Pour des raisons de simplicité, nous avons géré ces deux opérations de manière indépendante, tout en garantissant la cohérence entre les éléments finis et la géométrie. C'est à ce niveau que les opérations de substitution de maillages sont réalisées (voir la section 4.3, relative à la conception du couplage). Le détail du raffinement sera présenté plus tard dans ce chapitre.

5.4 Distribution-inconnues et distribution-éléments

Le point de départ de la méthode multigrille est la géométrie globale du code de calcul. La première étape consiste à distribuer la géométrie entre les processus (étape (1) de la figure 4.7). Sans connaissances *a priori* de la géométrie, les arêtes (inconnues) sont réparties de manière équitable entre les processus. En se basant sur la géométrie locale obtenue, les éléments finis distribués sont initialisés : création des éléments et inconnues et attribution des fonctions de *callback* aux différents types d'éléments.

La structure de la matrice, l'assemblage et les produits matrice-vecteur sont calculés à partir des éléments finis. Toutes ces opérations nécessitent une distribution des éléments et non une distribution des inconnues. Le pilotage de la méthode multigrille par le solveur direct implique alors la construction d'une distribution-éléments basée sur la distribution-inconnues du direct.

La méthode directe calcule sa distribution en fonction des inconnues. Comme le lien entre les inconnues et les arêtes est bijectif dans le cas des éléments de Nédélec, la distribution du solveur direct se résume à une distribution des arêtes. Or, la méthode itérative que l'on veut employer par la suite et le calcul de structure nécessitent une distribution des éléments. Le passage de la distribution-arêtes à la distribution-éléments n'est pas direct et nécessite une certaine attention.

En considérant deux arêtes distribuées sur deux processus différents, faisant partie d'un même élément (tétraèdre), on met en évidence le problème : l'élément est "à cheval" entre deux processus. On distingue alors trois types d'éléments :

– les éléments *purement locaux*, dont toutes les inconnues sont locales ;
– les éléments *partiellement locaux* ou *partagés*, dont au moins une inconnue est locale ;
– les éléments *distants*.

Lors du calcul de la structure présenté précédemment, les éléments utilisés sont les éléments purement et partiellement locaux. Toutefois, pour des calculs tels que le produit matrice-vecteur *matrix-free*, la gestion des éléments partiellement locaux est plus fine : la matrice élémentaire d'un élément partagé ne doit être calculée qu'une seule et unique fois. Cette nécessité impose d'attribuer chaque élément partagé à un unique processus. Pour chaque processus, le calcul du produit *matrix-free* nécessite alors le calcul des éléments purement

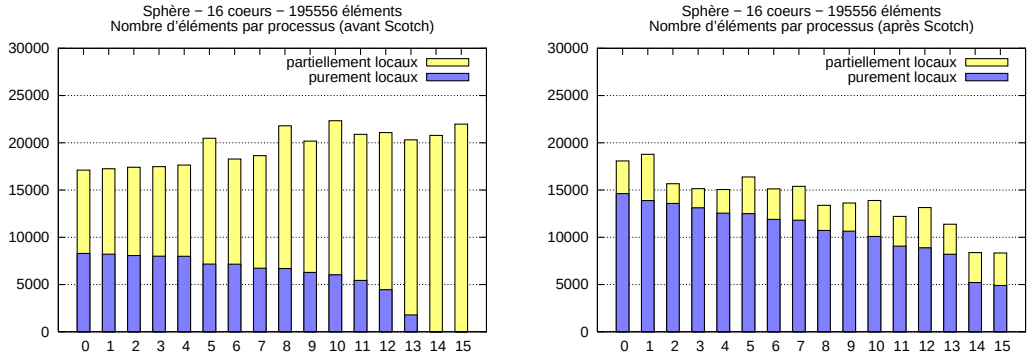


FIGURE 5.6: Effet de la renumérotation sur le nombre d'éléments purement locaux et partagés pour le cas test de la sphère. Les données ont été triées selon le nombre d'éléments purement locaux.

locaux et des éléments partiellement locaux qui lui ont été attribués.

Pour un produit matrice-vecteur parallèle performant, le nombre de calculs de matrices élémentaires effectués par chaque processus doit être sensiblement le même. La distribution du solveur direct concernant les arêtes (associées aux inconnues du système) et non les éléments, l'équilibrage de charge en termes d'éléments n'est pas garanti. Dans cette section, nous allons voir l'impact de la distribution du solveur direct sur l'équilibrage en termes d'éléments purement locaux, puis nous proposerons une méthode d'attribution des éléments partagés permettant de rattraper le déséquilibre de charge.

5.4.1 Effet de la distribution du solveur direct

Initialement, la distribution des arêtes employée est une distribution naïve des inconnues. De cette distribution initiale des arêtes, on a calculé la distribution des éléments et mis en évidence le nombre d'éléments purement et partiellement locaux. Le graphique de gauche de la figure 5.6 présente le résultat obtenu. À la vue de ce graphique, il est évident que l'équilibrage de charge en termes d'inconnues n'implique pas un équilibre en termes d'éléments. On s'intéresse alors à l'effet de la distribution des inconnues du solveur direct sur ce déséquilibre des éléments. Le graphique de droite de la figure 5.6 présente, pour le même problème, la distribution-éléments reconstruite à partir de la distribution directe des arêtes. En comparant les deux graphiques, il apparaît que la distribution directe réduit considérablement le nombre d'éléments partagés. Par contre, on peut constater un nombre d'éléments purement locaux allant du simple au double. Pour l'assemblage et le calcul de la structure de la matrice, basés sur les éléments partiellement et purement locaux, cette distribution présente un avantage : avec la distribution initiale des arêtes, chaque processus se basait sur en moyenne 30000 éléments, alors qu'avec la nouvelle distribution ces calculs ne nécessitent que 20000 éléments en moyenne. De plus, si on considère que les éléments partagés impliquent des communications lors du produit matrice-vecteur, la distribution du solveur direct améliore beaucoup les choses.

Cet exemple met en évidence le second problème : l'équilibrage de charge en termes d'éléments, basé sur la répartition des éléments partagés, ne peut être parfait. Par exemple, le processus 0 calculera au maximum 15000 éléments (tous ses éléments partagés lui sont attribués), alors

que le processus 9 en calculera au minimum 25000 (aucun élément partagé attribué). Malgré ce déséquilibre, nous allons tout de même tenter d'attribuer du mieux possible les éléments partagés.

5.4.2 Attribution des éléments partiellement locaux

Contrairement à l'assemblage, les calculs *matrix-free* nécessitent que chaque élément ne soit calculé qu'une seule fois. Les éléments partiellement locaux doivent alors être associés à un seul processus (les éléments purement locaux étant automatiquement attribués). Pour répartir les éléments partagés, nous avons plusieurs choix :

– calculer une répartition équilibrant la charge ;
– ou répartir de manière "aléatoire" ces éléments, sans prendre en compte l'équilibrage.

Équilibrage de charge : les tournois

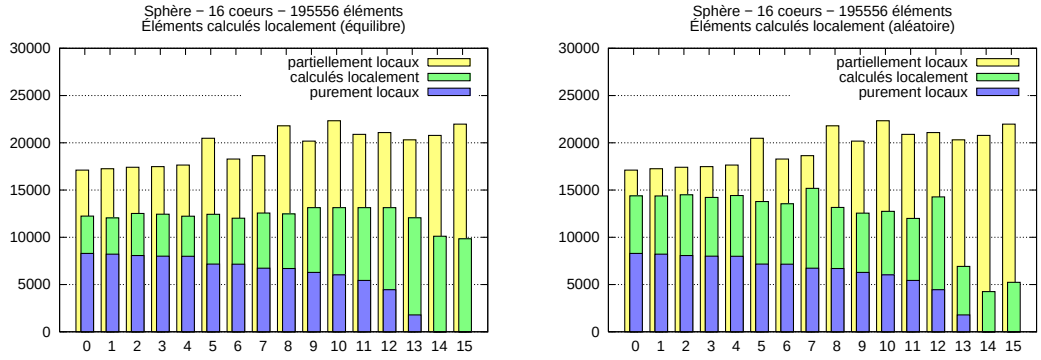
Pour la première méthode équilibrant la charge, un algorithme a été développé. Il repose sur des heuristiques simples afin de choisir quel processus calculera l'élément. Pour chaque élément partagé, un "tournoi" est organisé. Afin de limiter les communications, on fait en sorte que chaque concurrent (processus) participant à un tournoi possède au moins une arête de l'élément en question. Le choix du vainqueur se base sur les données suivantes :

– charge du processus (éléments locaux) par rapport aux autres concurrents ;
– charge globale moyenne en termes d'éléments calculés (éléments purement locaux et éléments partiellement locaux déjà attribués) ;
– nombre de tournois disputés par le processus ;
– nombre de tournois disputés par les autres concurrents ;
– majorité des arêtes dans un élément.

Le dernier critère veut qu'un processus ayant la majorité des arêtes au sein d'un élément en soit propriétaire. Bien que réduisant fortement les communications lors des produits matrice-vecteur, ce choix peut potentiellement réduire la marge de manœuvre quant à l'équilibrage de charge.

Les résultats obtenus par cette méthode sont présentés dans les figures 5.7 pour une distribution *avant* renumérotation. La barre jaune "purement + partiellement locaux" représente le nombre maximum d'éléments que le processus est susceptible de calculer. La barre "purement locaux" représente quant à elle le nombre minimum d'éléments calculés, attribués automatiquement au processus car seul compétiteur. La colonne "calculés localement" montre alors la quantité d'éléments effectivement calculés : les purement locaux plus ceux gagnés lors de tournois. On peut remarquer que dans le cas présenté dans ces figures, le ratio initial purement-partiellement local est plutôt mauvais. Pour ce test, nous avons utilisé un problème non renuméroté pour montrer de manière évidente l'amélioration de l'équilibrage apporté par les tournois. Ce déséquilibre est alors principalement dû à la numérotation initiale du maillage et au nombre de cœurs employés. Lors d'une résolution MG, cet algorithme travaillera sur un maillage renuméroté (pour le solveur direct) dont l'équilibrage sera meilleur : le prétraitement du solveur direct numérote de façon continue des parties connexes du graphe.

Malgré des résultats corrects et un bon équilibrage de charge (en termes d'éléments), l'algorithme utilisé est séquentiel, un processus ayant la charge de résoudre tous les tournois.



(a) Répartition équilibrée.

(b) Répartition aléatoire.

FIGURE 5.7: Attribution des éléments partagés pour calculs *matrix-free* (distribution avant renumérotation).

Pour des cas avec plus d'éléments, cette centralisation des tournois nécessite une quantité de communications trop importante. De plus, le déséquilibre observé dans les deux figures précédentes est symptomatique d'une mauvaise utilisation de PaStiX (trop de processeurs pour trop peu d'inconnues). Dans des conditions classiques, ce déséquilibre est moindre (semblable à celui de la figure 5.6) et une méthode plus basique est capable de gérer ce problème.

Attribution "aléatoire"

Malgré des résultats prometteurs de la méthode précédente, il a été choisi de répartir "aléatoirement" les éléments partagés. La méthode employée est simple et se repose sur la définition des éléments finis : un élément sera considéré local si sa première inconnue est locale au processus. Tous les processus se basant sur la même définition des éléments finis, ce calcul est réalisé en parallèle, sans communications et de manière déterministe. De plus, la complexité de cette méthode est bien inférieure à celle de la méthode précédente et ne nécessite que la classification des éléments entre locaux et distants : la notion de purement local, servant à attribuer automatiquement les éléments, n'est plus nécessaire ici.

Par contre, l'équilibrage de charge obtenu à l'issue de cette méthode dépend des données en entrée (EF). Une simple modification de l'ordre dans lequel sont listées les inconnues d'un élément peut changer radicalement son attribution. Cette méthode **ne garantit pas l'équilibrage de charge** et doit tenir compte de la structuration informatique des données. Malgré tout, en considérant à nouveau le graphique de droite de la figure 5.6, on peut supposer que la distribution-arêtes calculée par PaStiX mène à une distribution-éléments pas trop déséquilibrée pour des problèmes suffisamment gros.

5.5 Calcul parallèle de la structure de la matrice

Une fois les éléments finis distribués, il est nécessaire de connaître la structure de la matrice (pattern) afin de calculer le prétraitement de la méthode directe (renumérotation et distribution). La structure se présente sous la forme d'une CSC sans coefficients. Le traitement

consiste à détecter les non-zéros de la matrice : pour chaque couple d'inconnues (i, j), A_{ij} est un non-zéro si, et seulement si, i et j appartiennent à un même élément. Plus simplement, l'algorithme consiste à parcourir chaque élément et noter les non-zéros créés. Dans le cas parallèle, le traitement est identique, sauf que seules les inconnues locales nous intéressent. La procédure 5.1 présente le traitement réalisé en parallèle.

Procédure 5.1 : structureCSC

```
1 pour tous les éléments e faire
2     pour i = 0, dimension faire
3         si l'inconnue e[i] est locale alors
4             pour j = 0, dimension faire
5                 ajouter le non-zéro e[j] dans la colonne (locale) e[i]
```

Malheureusement, on ne connaît pas *a priori* la taille de la matrice locale. Le calcul de la structure nécessite alors soit l'utilisation de structures de données dynamiques, soit une matrice dense de dimension `nloc*n`. Pour limiter la consommation mémoire de cette étape, le choix des structures dynamiques a été retenu.

Afin d'obtenir une méthode parallèle efficace, on se base sur la distribution initiale des éléments présentée précédemment (avant attribution des éléments partiellement locaux). En effet, dans la procédure 5.1, on peut remarquer que seuls les éléments contenant une inconnue locale génèrent un non-zéro dans la matrice locale. Ainsi la boucle ne concerne que ces éléments, qui sont les éléments purement et partiellement locaux. Pour cette étape, l'attribution des éléments partiellement locaux n'est pas nécessaire : comme on ne fait qu'utiliser la liste des inconnues de l'élément, aucun calcul n'est effectué par cette opération. Le fait de "calculer" un même élément plusieurs fois est alors peu coûteux. De plus, le calcul multiple d'un élément permet d'éviter les communications de non-zéros entre processus.

5.6 Produits matrice-vecteur parallèle

Les produits matrice-vecteur constituent la base des méthodes employées par le solveur multigrille. Toutes les opérations *matrix-free* reposent sur le produit matrice-vecteur *matrix-free*. Nous verrons d'abord le produit matrice-vecteur parallèle dans le cadre général, avant d'aborder la notion de *matrix-free*.

5.6.1 Approche parallèle classique

Supposons la matrice A de dimension $n \times n$ distribuée entre différents processus p, chacun possédant n_p colonnes. Pour chaque processus, nous noterons A_p sa matrice locale. Le découpage de la matrice A en bloc-colonnes implique une distribution analogue du vecteur x : si une colonne i est locale, alors le terme x_i est lui aussi local. La figure 5.8 représente la matrice globale et sa distribution.

Le calcul $Ax = y$ utilisant ces notations se résume à :

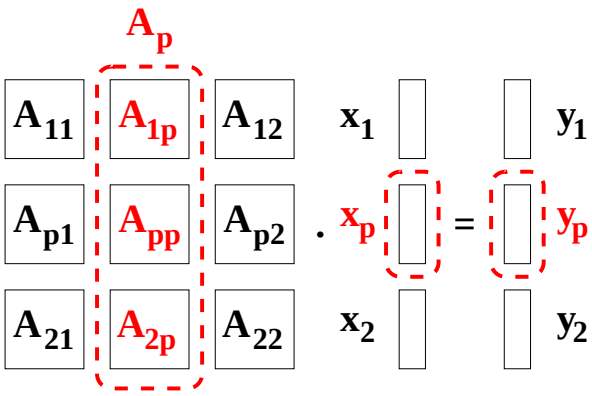


FIGURE 5.8: Matrice distribuée.

$$y_1 = A_{11}x_1 + A_{12}x_2 + A_{1p}x_p, \tag{5.2}$$
$$y_2 = A_{21}x_1 + A_{22}x_2 + A_{2p}x_p, \tag{5.3}$$
$$y_p = A_{p1}x_1 + A_{p2}x_2 + A_{pp}x_p. \tag{5.4}$$

De ces notations, on en déduit que la valeur de y_p reçoit des contribution non locales ($A_{p1}x_1 + A_{p2}x_2$), et que le bloc x_p contribue pour sa part aux calculs des blocs distants y_1 et y_2. Dans le cas dense, ces dépendances imposent que tous les processus disposent du vecteur x. Par contre, dans le cas creux, le calcul de y_p nécessite uniquement les valeurs de x qui sont en regard des non-zéros de A_{1p} et A_{2p}. Avec la renumérotation du solveur direct, la quantité de données distantes nécessaires correspond au nombre de connexions entre les inconnues locales et les inconnues distantes (appartenant à un autre sous-graphe). Comme le partitionneur génère des sous-graphes connexes et fait en sorte de limiter la coupe, ces interfaces entre sous-graphes sont de petite taille et ne concernent qu'un nombre limité de processus. Dans ce cas, on a un produit matrice-vecteur parallèle efficace, ne nécessitant que peu de communications.

Synthétiquement, l'algorithme se décompose en :

1. calcul de la contribution locale de A_p au vecteur y ($A_{1p}x_p$, $A_{2p}x_p$ et $A_{pp}x_p$) ;
2. envoi des contributions y_1 et y_2 aux processus gérant ces termes ;
3. réception des contributions externes $A_{p1}x_1$ et $A_{p1}x_2$ à y_p et mise à jour.

Hormis les envois et réceptions (voir section 5.6.3 pour plus de détails), ce calcul est simple, et met en jeu seulement des produits matrice-vecteur et une addition vecteur-vecteur.

5.6.2 Produit *matrix-free*

La présentation du produit matrice-vecteur creux précédent suppose que la matrice A est stockée. Dans le cadre des éléments finis, la matrice A est assemblée à partir de matrices élémentaires A^e. Chaque matrice élémentaire que l'on considère ici est une matrice de dimension $n \times n$ composée quasi-exclusivement de zéros. Pour un élément quelconque, seuls les termes A^e_{ij} avec i et j un couple d'inconnues de l'élément sont non-nuls (figure 5.9).

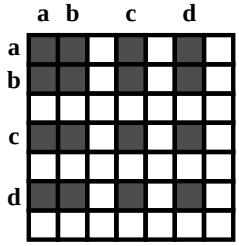


FIGURE 5.9: Matrice élémentaire A^e de l'élément composé par les inconnues (a, b, c, d).

L'assemblage de A se fait en sommant les matrices élémentaires de tous les éléments du maillage. Dans notre cas, le calcul de A_p se fait en assemblant uniquement les matrices élémentaires ayant une contribution à la matrice locale. On utilise alors la formule suivante :

$$Ax = (\sum A^e)x = \sum (A^e x). \tag{5.5}$$

En se basant sur cette expression du produit matrice-vecteur, le calcul peut être effectué sans jamais former A en totalité. Bien que dans cette expression A^e soit de dimension $n \times n$, elle est composée presque uniquement de zéros. Seuls les non-zéros seront stockés sous forme de structure de matrice dense.

Un des inconvénients de cette méthode concerne les éléments à cheval entre deux processus (faisant intervenir des inconnues locales et distantes). En effet, ils doivent soit être calculés qu'une seule fois, soit ne mettre à jour que les inconnues locales. Dans la figure 5.10 les termes de la matrice élémentaire qui posent problème sont mis en évidence (termes rouges).

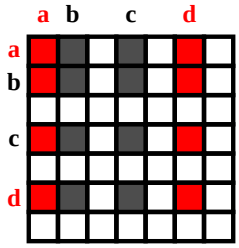


FIGURE 5.10: Matrice élémentaire A^e de l'élément composé par les inconnues (a, b, c, d), avec seulement b et c locales.

Soit chaque processus calcule sa matrice A_p locale, en utilisant tous les éléments ayant une contribution locale, soit l'élément n'est calculé que par un seul processus, les termes non locaux étant tout de même utilisés localement. Cette seconde méthode est valide car ces termes non locaux ont une contribution dans les termes en regard des non-zéros de la matrice A_p, contributions qui sont ensuite communiquées au processus possédant ces inconnues.

Pour limiter le nombre de calculs de matrices élémentaires, c'est cette seconde solution que nous avons choisi d'implémenter. Pour cela, il faut, pour toute matrice élémentaire, définir quel processus calcule la matrice élémentaire. Ce choix, présenté dans la partie 5.4.2, se base sur la définition de l'élément fini : si la première inconnue de l'élément est locale, alors le calcul de l'élément est réalisé localement, sinon le calcul est réalisé par un autre processus.

C'est ce que l'on a appelé la "distribution aléatoire". Cette technique permet de ne calculer qu'une seule et unique fois chaque matrice élémentaire, et ce à moindre coût (la détection se fait en temps constant).

5.6.3 Communications

Les contributions distantes calculées localement doivent désormais être transmises à leurs propriétaires. En se basant sur la définition des éléments finis, chaque processus connaît ses inconnues locales. De plus, chacun est capable de calculer la structure de son bloc-colonne A_p. Chaque ligne de A_1p et A_2p contenant un terme non nul représente une communication. On connaît alors quelles sont les inconnues nécessaires, mais pas leur propriétaire.

Afin de calculer ces appartenances, tous les processus construisent une "requête" composée des indices des inconnues distantes nécessaires au calcul (soit tous les termes de x en regard des non-zéro de A_1p et A_2p). Une fois cette requête calculée, elle est transmise au processus voisin dans une topologie en anneau. Chaque processus reçoit de son prédécesseur la liste des inconnues manquantes. À partir de cette liste, chacun la partitionne entre les inconnues locales, et les inconnues distantes. Les inconnues locales sont alors des inconnues que l'on devra envoyer au processus ayant émis la requête. La requête est modifiée comme dans la figure 5.11. Chaque processus réduit alors la taille de la liste à traiter, marquant le début de ses inconnues locales par un drapeau.

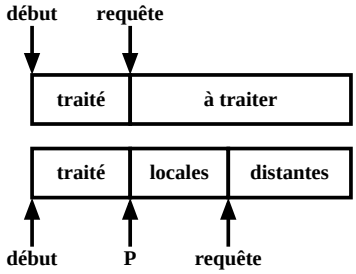


FIGURE 5.11: Modification de la requête par le processus p. Le début des arêtes locales à p est marqué.

Au fur et à mesure de la progression de la requête dans l'anneau, la taille de la partie "à traiter" diminue. Une fois revenue à son expéditeur, la requête est vide car toutes les inconnues ont été attribuées à un processus. À partir des drapeaux placés par chaque processus sur la requête, chaque processus connaît le propriétaire des données nécessaires au calcul.

En marge de ce traitement, les données à envoyer sont elles aussi stockées. Une fois la partition locale-distante effectuée, le processus est alors certain que les inconnues locales doivent être envoyées au processus émetteur de la requête. À la fin de cet algorithme, tous les processus savent exactement à qui envoyer des données et de qui en recevoir. Ces informations sont capitales, car une fois la distribution effectuée, c'est toujours le même schéma de communication qui sera utilisé. Le seul problème qui pourrait intervenir, est celui du raffinement. En effet, si les inconnues concernées par les communications sont modifiées par le raffinement, il faut alors modifier ce plan de communication. Nous verrons plus loin que dans notre cas cette modification n'est pas nécessaire, car les inconnues concernées sont constantes d'un niveau à l'autre.

5.7 Lisseur

À partir du produit matrice-vecteur présenté précédemment, nous sommes en mesure de développer le lisseur. Le choix du lisseur s'est porté sur la méthode de Jacobi amortie (Damped Jacobi). Comme présenté dans la partie 1.1.2, nous utilisons l'équation d'itération :

$$x^{[k+1]} = x^{[k]} + \omega D^{-1}(b - Ax^{[k]}). \tag{5.6}$$

Cette opération nécessite deux composantes : le produit matrice-vecteur et la diagonale de la matrice. Puisque nous souhaitons effectuer cette méthode sans assembler la matrice mais que la diagonale est nécessaire, on a décidé de n'assembler que cette dernière. Dans l'algorithme, la diagonale n'est utilisée qu'après le produit matrice-vecteur. Comme ce dernier parcourt tous les éléments finis, il est possible d'assembler la diagonale en même temps que le produit est calculé. Comme la diagonale est constante au cours des itérations, cet assemblage n'est à réaliser qu'une seule fois, lors du premier appel au produit matrice-vecteur. À l'instar des autres vecteurs, la diagonale est elle aussi distribuée. Son coût de stockage est alors minime.

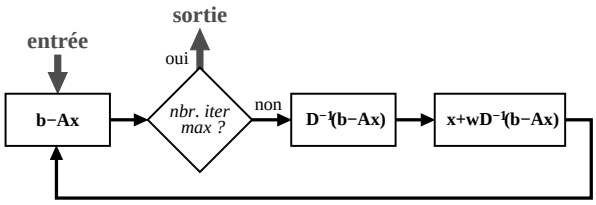


FIGURE 5.12: Algorithme de Jacobi.

On peut remarquer que le calcul du résidu est effectué à chaque itération. On utilise cette information pour suivre l'évolution de la méthode. Afin de mieux tracer le comportement, un calcul de résidu supplémentaire est effectué en sortie de l'algorithme : cela permet, une fois la dernière itération effectuée, de connaître le résidu final calculé par Jacobi (voir figure 5.12). La modification du placement du test d'arrêt présentée dans la figure 5.12 fait en sorte que la première et la dernière opération effectuées soient un calcul de résidu.

5.8 Raffinement et déraffinement

Maintenant que toutes les opérations algébriques ont été définies, il est nécessaire de déterminer la méthode de raffinement. Comme on se place dans le cadre des éléments de Nédélec 3D, on s'intéressera tout particulièrement aux raffinements de tétraèdres. Plusieurs méthodes de raffinement existent, mais nous étudierons plus particulièrement les deux suivantes :

– le raffinement au centre de gravité, connectant chaque sommet au centre de gravité du tétraèdre ;
– le raffinement au centre des arêtes, ajoutant un nœud au centre de chaque arête ainsi qu'au centre de gravité de chaque tétraèdre.

Nous présenterons ici ces deux types de raffinements d'un point de vue géométrique. Il est important de rappeler que les maillages raffinés devront respecter les conventions du code de calcul.

5.8.1 Raffinement au centre de gravité

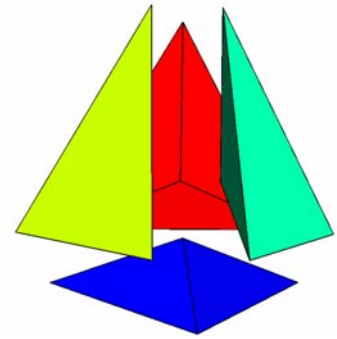


FIGURE 5.13: Raffinement au centre de gravité en 3D.

Pour chaque tétraèdre, on ajoute un nœud au centre de gravité que l'on relie à chaque sommet (figure 5.13). En notant a_f, f_f et t_f respectivement les nombres d'arêtes, de faces et de tétraèdres de la grille fine (idem pour la grille grossière avec $\cdot_g$), on exprime :

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 6 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} a_g \\ f_g \\ t_g \end{pmatrix} = \begin{pmatrix} a_f \\ f_f \\ t_f \end{pmatrix}. \tag{5.7}$$

Le principal avantage de cette méthode est la conservation de la frontière (faces des tétraèdres). Dans la DDM, cette propriété est importante, car le couplage se fait par l'intermédiaire des faces externes des sous-domaines. Ainsi, le couplage domaine-domaine de la DDM n'est pas modifié par ce raffinement (maillages conformes). Par contre, après plusieurs raffinements, les éléments créés tendent à s'aplatir (exemple en 2D dans la figure 5.14). De plus, cette méthode n'est pas optimale en termes de longueurs d'arêtes : les arêtes grossières sont conservées et certains centres de gravité convergent vers les milieux des arêtes grossières. Ce comportement implique que la taille moyenne des arêtes diminue lentement. Pour les problèmes d'électromagnétisme où la longueur moyenne des arêtes et la qualité des éléments (rapport d'aspect) ont une grande importance, ce type de raffinement n'est pas forcément bien adapté.

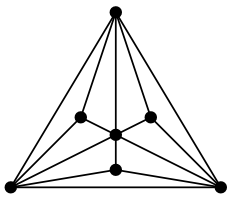


FIGURE 5.14: Aplatissement des éléments 2D après 2 raffinements.

5.8.2 Raffinement au centre des arêtes

Le raffinement au centre des arêtes n'a pas les problèmes du raffinement au centre de gravité : chaque arête du maillage initial est scindée en deux et les nouveaux tétraèdres ne sont pas

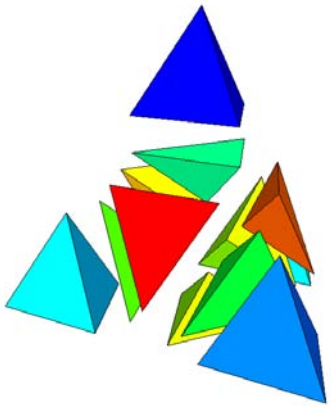


FIGURE 5.15: Raffinement au centre des arêtes en 3D.

aplatis. Par contre, la modification du maillage est très importante : le nombre de tétraèdres fins est douze fois plus important que le nombre de tétraèdres grossiers. Avec les mêmes notations que précédemment on exprime la taille du maillage fin :

$$\begin{pmatrix} 2 & 3 & 6 \\ 0 & 4 & 16 \\ 0 & 0 & 12 \end{pmatrix} \begin{pmatrix} a_g \\ f_g \\ t_g \end{pmatrix} = \begin{pmatrix} a_f \\ f_f \\ t_f \end{pmatrix}. \tag{5.8}$$

Cette augmentation rapide de la taille du maillage présente un désavantage : les maillages auront tendance à devenir très volumineux pour des niveaux de raffinements peu élevés (2-3 raffinements).

5.8.3 Choix du raffinement : motivations

Comme on a pu le voir dans les présentations des raffinements à notre disposition, chacun a ses spécificités. Toutefois, le fait de coupler notre méthode multigrille avec le code de calcul du CEA nous oblige à respecter plusieurs contraintes :

– la conservation des surfaces externes des domaines ;
– les conventions de numérotation du code de calcul.

La conservation des surfaces est nécessaire au bon fonctionnement de la DDM. Si ce n'est pas le cas, la solution fine retournée par le solveur multigrille portera sur des arêtes inconnues du code de calcul. Le seul moyen de retourner la solution serait alors de restreindre le résultat calculé jusqu'au niveau grossier. On aurait alors calculé une solution fine pour ensuite en retourner une version "dégradée". La perte d'informations lors de la restriction risque alors de rendre notre méthode multigrille inadaptée.

De plus, pour pouvoir calculer les matrices élémentaires, le code de calcul se base sur sa géométrie locale. Il faut alors que les géométries calculées respectent les conventions du code de calcul. Dans le cas du raffinement au centre des arêtes, ces conventions et surtout les redondances (voir section 4.1.1) sont complexes et coûteuses à calculer.

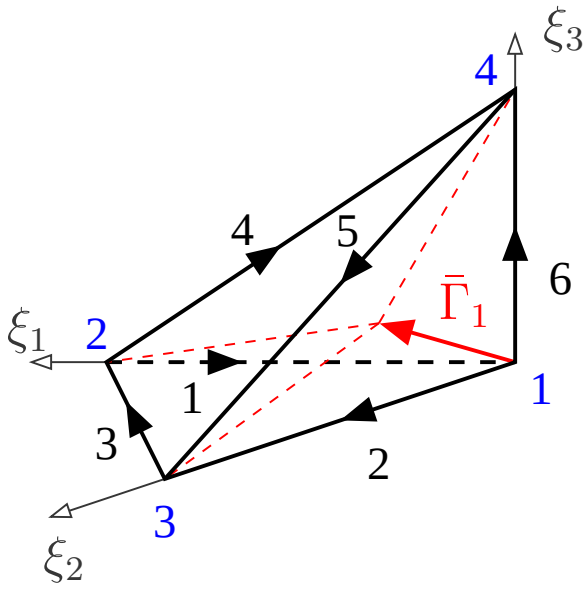


FIGURE 5.16: Élément de référence et prolongation.

Finalement, avec les expressions reliant les nombres d'arêtes, faces et tétraèdres des grilles grossières et fines, il est évident que le raffinement au centre des arêtes raffine trop rapidement la géométrie : après deux raffinements, le nombre de tétraèdres est multiplié par 144, et par 1728 après trois raffinements. Si initialement le nombre de tétraèdres est mal réparti, le déséquilibre est amplifié très rapidement.

Bien que non exempte de défauts, la méthode de raffinement au centre de gravité a été choisie. La principale raison de ce choix est liée à la conservation de la surface externe : pour le couplage à la DDM, la solution fine peut être restreinte à la surface externe sans perte d'information (injection). De plus, le calcul de la nouvelle géométrie est très simple et les conventions du code de calcul peuvent être respectées sans trop de calculs supplémentaires. Toutefois, le problème de l'aplatissement des éléments persiste : pour un nombre de raffinements raisonnable (3-4), on considérera que l'aplatissement des éléments est acceptable.

Le raffinement parallèle se fera de la sorte : chaque processus raffine uniquement les éléments qui lui ont été attribués (voir la section 5.4 discutant de la distribution des éléments). Les arêtes créées par raffinement, internes à l'élément local, sont automatiquement attribuées au processus propriétaire de l'élément.

5.9 Prolongation et restriction

5.9.1 Prolongation : fonctions de base

Dans cette partie, nous nous intéresserons à la prolongation utilisée lors du raffinement au centre de gravité. En posant γ_s la valeur de l'inconnue portée par l'arête $\bar{\Gamma}_s$ qui relie le sommet s au centre de gravité et en utilisant la définition des éléments de Nédélec (2.28) :

$$\gamma_s = \int_{\bar{\Gamma}_s} E(x) \cdot \vec{\bar{\tau}}_s \, \mathrm{d}\bar{\Gamma}_s. \tag{5.9}$$

En utilisant l'expression du champ électrique dans l'élément (tétraèdre) de référence (2.29) :

$$\gamma_s = \int_{\bar{\Gamma}_s} \left[\sum_{i=1}^{6} (\alpha_i \vec{p}_i(x)) \right] \cdot \vec{\bar{\tau}}_s \, \mathrm{d}\bar{\Gamma}_s, \tag{5.10}$$

avec $\vec{p}_i$ les fonctions de bases liées aux arêtes Γ_i de l'élément de référence. En considérant l'arête $\bar{\Gamma}_1$ orientée comme dans la figure, la relation (5.10) se simplifie :

$$\gamma_1 = \frac{1}{4}(-\alpha_1 + \alpha_2 + \alpha_6). \tag{5.11}$$

Pour toute autre arête $\bar{\Gamma}_s$ reliant le sommet s au centre de gravité (orientée vers le centre de gravité) la même expression apparaît : l'inconnue γ_s est une combinaison des valeurs α_i portées par les arêtes incidentes à s. Le signe dépend uniquement de l'orientation de ces arêtes : si l'arête Γ_i est orientée vers le sommet s (entrante) le signe est négatif, sinon il est positif (sortante). Pour l'élément de référence, on a alors :

$$\gamma_1 = \frac{1}{4}(-\alpha_1 + \alpha_2 + \alpha_6), \tag{5.12a}$$
$$\gamma_2 = \frac{1}{4}(+\alpha_1 - \alpha_3 + \alpha_4), \tag{5.12b}$$
$$\gamma_3 = \frac{1}{4}(-\alpha_2 + \alpha_3 - \alpha_5), \tag{5.12c}$$
$$\gamma_4 = \frac{1}{4}(-\alpha_4 + \alpha_5 - \alpha_6). \tag{5.12d}$$

Pour toutes les autres inconnues portées par les arêtes grossières, les valeurs sont calculées par injection. Sous forme matricielle, la prolongation dans l'élément de référence se note :

$$\frac{1}{4}\begin{pmatrix} +4 & . & . & . & . & . \\ . & +4 & . & . & . & . \\ . & . & +4 & . & . & . \\ . & . & . & +4 & . & . \\ . & . & . & . & +4 & . \\ . & . & . & . & . & +4 \\ -1 & +1 & . & . & . & +1 \\ +1 & . & -1 & +1 & . & . \\ . & -1 & +1 & . & -1 & . \\ . & . & . & -1 & +1 & -1 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix}, \tag{5.13}$$

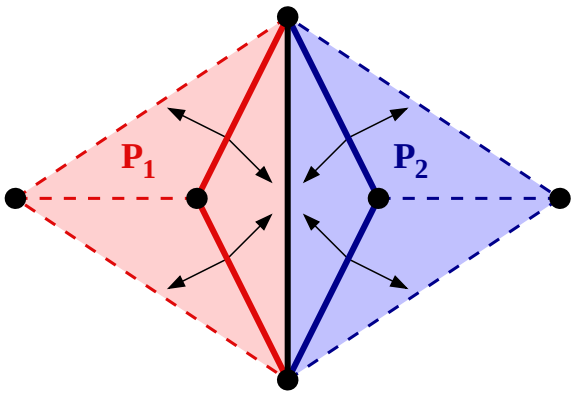


FIGURE 5.17: Contributions lors de la restriction pour les éléments de Nédélec 2D en parallèle. Les éléments sont sur les processus P_1 et P_2, l'arête centrale sur P_3.

avec α et γ représentant respectivement les anciennes (grossières) et nouvelles (fines) inconnues.

Les expressions (5.12) ne contenant pas de données relatives à la base (ξ_1, ξ_2, ξ_3), le passage de l'élément de base à l'élément réel est immédiat. Dans le code de calcul et lors du raffinement, l'orientation des arêtes n'est pas contrainte et peut ne pas correspondre à l'orientation de l'élément de référence. Toutefois, par définition (2.28) on sait que pour deux arêtes superposées Γ_i et Γ_j de vecteurs directeurs respectifs $\vec{\tau}_i = -\vec{\tau}_j$, on a $\alpha_i = -\alpha_j$. Lors du calcul de la prolongation, le traitement à réaliser consiste à détecter les arêtes incidentes à un sommet, à déterminer leurs orientations relatives (entrantes ou sortantes) pour ensuite combiner leurs valeurs. Ce calcul relativement simple permet d'obtenir un opérateur de prolongation respectant la définition des éléments finis sur chaque niveau.

5.9.2 Restriction : injection

Comme présenté dans la section 1.2.2, l'opération de restriction peut dépendre de la restriction. Si l'on prend la restriction comme la transposée de l'opérateur (5.12), un problème se pose : une arête peut appartenir à plusieurs éléments. Lors de la restriction avec l'opérateur transposé, les arêtes grossières reçoivent des contributions des arêtes fines. Ces contributions arrivent alors de tous les éléments adjacents. Si les éléments appartiennent à des processus différents, la restriction nécessite des communications.

La figure 5.17 montre ce comportement : si l'arête centrale (noire) est gérée par le processus P_3, la restriction nécessite les valeurs des arêtes rouges et bleues, situées respectivement sur P_1 et P_2 (les processus à qui ont été attribués les éléments). Le schéma de communication nécessaire à cette opération est complexe à mettre en place : P_3 doit avoir connaissance des arêtes internes (fines) de P_1 et P_2. Cela revient alors à calculer le raffinement des éléments fantômes.

Afin de résoudre ces problèmes, le choix de la restriction s'est porté sur l'injection des données de la grille fine vers la grille grossière. Sous forme matricielle, l'opérateur de restriction se note :

$$\begin{pmatrix} 1 & . & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . & . \\ . & . & . & 1 & . & . & . & . & . & . \\ . & . & . & . & 1 & . & . & . & . & . \\ . & . & . & . & . & 1 & . & . & . & . \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{pmatrix}. \tag{5.14}$$

Le choix de l'injection en tant que restriction rend l'opérateur multigrille (1.29) non symétrique. Cette non symétrie fait que des méthodes telles que le gradient conjugué ne peuvent utiliser ce solveur en tant que préconditionneur.

5.10 Calcul de l'opérateur

La méthode multigrille nécessite la définition de l'opérateur pour chaque niveau de grille. L'utilisation de la méthode de Galerkin, calculant l'opérateur fin à partir des opérateurs de restriction-prolongation et de l'opérateur grossier est difficilement applicable. Pour une méthode multigrille *matrix-free* à deux niveaux, avec e le nombre d'éléments sur le niveau grossier, la méthode de Galerkin s'exprime :

$$A_f = PA_gR = \sum_e (PA_g^eR). \tag{5.15}$$

En généralisant au cas du multigrille à n niveaux, cette expression devient :

$$A_n^e = P^n \dots P^1 A_0^e R^1 \dots R^n, \tag{5.16}$$

avec A_n^e la matrice "élémentaire" sur le niveau fin et A_0^e la matrice élémentaire sur le niveau grossier. P^k et R^k représentent les opérateurs de prolongation et de restriction entre les niveaux $k-1$ et k. La première remarque porte sur la matrice A_n^e : cette matrice élémentaire, couvre tous les éléments fins issus des raffinements de l'élément grossier e. D'un niveau à l'autre, le nombre de matrices élémentaires reste constant, alors que leur dimension augmente. De plus, A_0^e étant dense, A_n^e l'est aussi. Du fait de l'approximation de l'opérateur et des matrices denses de grandes dimensions, cette méthode n'est pas applicable ici.

On a alors choisi de rediscrétiser le problème sur chaque niveau. Cette rediscrétisation est effectuée par le code de calcul. Dans notre cas, fournir le nouveau maillage au code de calcul suffit : les calculs de matrices élémentaires se baseront alors non plus sur l'ancienne géométrie, mais sur la géométrie du niveau actuel. Les matrices élémentaires calculées correspondent alors à la discrétisation du problème sur ce nouveau maillage.

Troisième partie

Validation numérique et tests

Chapitre 6

Validation numérique et passage à l'échelle

6.1 Méthodologie

Pour évaluer les performances numériques de la méthode multigrille, nous nous sommes intéressés à deux points : la qualité de la solution calculée et la convergence de la méthode. La qualité de la solution est relative au fait que le code de calcul converge vers une SER correcte tout au long des itérations globales. Nous nous intéresserons aussi au solveur multigrille (MG) en lui même et nous estimerons ses performances en nous focalisant sur la convergence de la méthode (hors itération externe du code de calcul). Finalement, la scalabilité de la méthode sera testée, tout d'abord en démontrant une indépendance de la convergence par rapport au nombre de cœurs prenant part au calcul, puis en comparant les temps de calcul sur des cycles équivalents (même nombre de cycles en V, même nombre de pré et post-lissages).

L'objectif de cette thèse est de remplacer le solveur direct utilisé pour résoudre les systèmes linéaires à l'intérieur de chaque sous-domaine de notre code de calcul. Il nous faudra donc montrer que la convergence du MG est suffisante pour que le processus global (itérations sur les sous-domaines) ne soit pas perturbé et que la SER obtenue soit la même que celle calculée actuellement. Le MG doit donc conduire à des niveaux d'erreur suffisamment bas et avec une bonne vitesse de convergence : ces deux points seront examinés dans les courbes de convergence présentées ci-dessous. Dans un code complexe comme celui utilisé ici (le MG est ici imbriqué dans un processus itératif externe sur les sous-domaines que nous avons appelé "itération globale"), il est impossible de décider *a priori* quel niveau de résidu relatif sur le MG est suffisant pour que le processus global fonctionne. La qualité de la solution globale sera donc évaluée par la comparaison des SER obtenues sur le maillage fin par un calcul classique utilisant le solveur direct et par un calcul où celle-ci est remplacée par le MG. Bien sûr, cette comparaison sera limitée aux cas atteignables par un solveur direct, c'est à dire quelques dizaines de millions d'inconnues.

Les tests que nous présentons permettent de mettre en évidence le fait que la convergence de la méthode MG se dégrade quand le nombre d'onde k augmente (rappel : $k = 2\pi f\sqrt{\epsilon_0\mu_0}n$ où n est l'indice du matériau et f la fréquence de l'onde incidente). On peut établir que plus k augmente, plus la matrice du système devient non définie positive (voir équation 2.5). La

fréquence et l'indice jouent un rôle similaire vis à vis de k. En conséquence, en section 6.2.3 nous montrerons la dégradation de la convergences due à l'augmentation de l'indice et en section 6.3.1 la même dégradation due à l'augmentation de la fréquence.

À nombre d'onde donné et à fréquence donnée, la présence de pertes diélectriques améliore la convergence : ce point est démontré en section 6.2.3 où nous comparons des convergences pour des matériaux de même indice mais avec et sans pertes.

À nombre d'onde donné, la convergence s'améliore aussi quand on augmente le nombre de mailles par longueur d'onde, même si c'est au prix d'une augmentation de la taille du système linéaire : ce point est bien montré pour Helmholtz [46]. Nous le retrouverons ici pour Maxwell en section 6.3.1 pour le cas à 5GHz. Néanmoins, ceci est contrebalancé par le fait que le raffinement de maillage choisi ici conduit à un *aplatissement progressif des éléments finis*, ce qui a un effet négatif sur la convergence (suite de la section 6.3.1). La question de l'influence de cette dégradation du rapport d'aspect du maillage sur la valeur de la SER calculée est abordée en fin de section 6.2.5. Néanmoins, cela fait partie des points qui restent à approfondir car nous avons d'autres cas (non présentés ici) où la SER calculée sur des maillages "aplatis" n'est pas parfaite.

Quand la dégradation de convergence est telle que les niveaux de résidus relatifs sont insuffisants ou même qu'il y a divergence, nous avons repris les idées d'Elman et Erlangga [17, 18] sur Helmholtz : nous utiliserons le MG comme préconditionneur d'une méthode de Krylov. Comme notre opérateur de restriction n'est pas le transposé de celui de prolongation, un tel préconditionneur est non-symétrique, c'est pourquoi nous avons choisi de préconditionner un GMRES : en section 6.2.4, nous comparons le comportement d'un GMRES seul à celui d'un GMRES préconditionné par le MG dans le cas où le MG seul convergeait (grâce à la présence de pertes) et dans le cas où il ne convergeait pas.

Toutes ces comparaisons se feront essentiellement sur deux cas tests : la sphère et l'haltère. Un troisième cas purement tridimensionnel a été testé, mais n'est pas présenté. Avant tout cela, nous aurons indiqué en section 6.2.1 comment nous avons choisi le coefficient de sous-relaxation du lisseur Jacobi pour le cas test de la sphère et nous comparerons les convergences d'un V-cycle et d'un MG en section 6.2.2.

Ce chapitre se terminera par des présentations de performances en termes de scalabilité des différentes parties de l'algorithme : à partir du maillage de l'haltère plus ou moins raffiné, nous montrerons que la partie multigrille passe très bien à l'échelle. Néanmoins, nous mettrons en évidence une plus faible scalabilité du solveur direct due à la faible quantité de calculs par cœur au niveau grossier. En contournant ce problème par l'utilisation de niveaux grossiers plus volumineux, nous arriverons à traiter un problème sur le niveau fin comportant plus de 1.3 milliards d'inconnues avec de bonnes performances d'ensemble.

Tous les tests présentés ici ont été réalisés sur la machine TERA100 du CEA/DAM. Cette machine regroupe 4324 serveurs de calcul à 32 cœurs (Intel Xeon 7500) pour un total de 138 368 cœurs de calculs. L'ensemble du système est interconnecté par des commutateurs *Infiniband* QDR (Quad Data Range). En juin 2011 cette machine se classe 9^{e} machine mondiale et 1ère européenne dans le TOP500 (www.top500.org) et a atteint 1.05 pétaflops sur le test du Linpack en Novembre 2010 pour une performance crête de 1.25 pétaflops.

6.1.1 Comparaison de SER

Comme indiqué ci-dessus, nous comparerons des SER calculées en utilisant la méthode multigrille sur le maillage fin à des SER calculées par la méthode habituelle (solveur direct) sur les mêmes maillages fins. En section 6.2.5 nous la comparerons également à la SER calculée sur le maillage grossier qui, dans ce cas, est suffisamment maillé. Nous montrerons par ce test que l'aplatissement des éléments finis n'a pas trop d'impact sur le calcul et la qualité de la SER.

Pour ces comparaisons, nous nous plaçons dans le cadre des études bistatiques : la SER est calculée pour plusieurs angles d'observation et ce pour un angle d'incidence donné. L'angle d'incidence étant unique, une seule résolution est nécessaire. Une fois calculé, le champ électrique sur la surface externe permet de reconstituer la SER de l'objet pour tout angle d'observation. Par la suite, toutes les SER présentées seront exprimées en décibel (gain) :

$$\mathrm{SER}_{dB} = 10 \log_{10}(|\,\mathrm{SER}\,|). \tag{6.1}$$

Du fait de cette expression logarithmique de la SER, les faibles niveaux seront plus sensibles aux variations de la solution. Pour la simulation électromagnétique, et plus particulièrement la simulation d'objets furtifs, ce sont ces faibles niveaux de SER qui sont importants.

6.1.2 Convergence de la méthode

La qualité de la solution est évaluée au niveau du code de calcul. Pour le calcul de SER, la résolution multigrille s'inscrit dans l'itération globale et participe alors à la bonne (ou mauvaise) convergence de la méthode. En comparant les SER calculées à des SER de référence, on peut voir si la convergence est bonne et donc conclure si les solutions volumiques sont de bonne qualité. D'un autre côté, le comportement intrinsèque de la méthode multigrille nous intéresse aussi. Pour évaluer ce point, nous nous intéresserons à l'évolution de la norme du résidu relatif

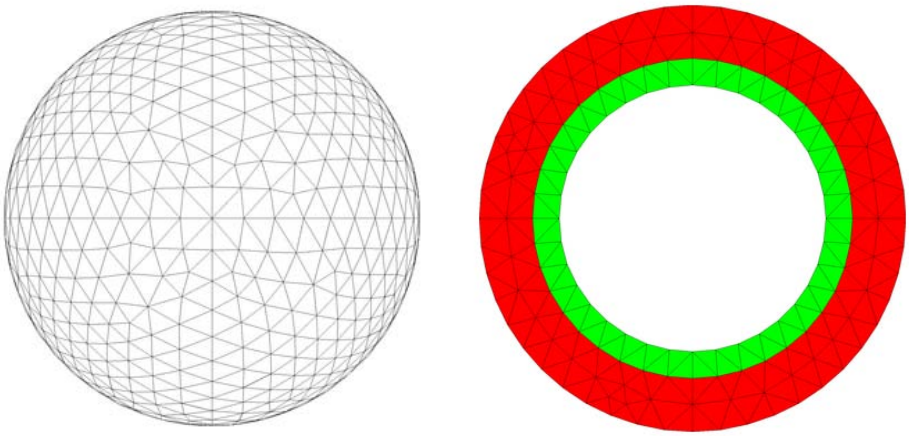
$$\frac{\|r\|_2}{\|b\|_2} = \frac{\|b - Ax\|_2}{\|b\|_2} \tag{6.2}$$

avec la norme vectorielle complexe habituelle

$$\forall v \in \mathbb{C}^n, \|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i \bar{v}_i}. \tag{6.3}$$

Néanmoins, l'application des conditions limites étant réalisée par la méthode du "grand nombre" [1] les erreurs commises au niveau de la condition limite sont amplifiées par la diagonale D. Dans les cas extrêmes, ces erreurs modifient en profondeur le résidu relatif et donnent une vision erronée de la qualité réelle de la solution. Pour remédier à ce problème, nous n'utiliserons pas la norme 2, mais plutôt la norme D^{-1}, définie comme suit :

1. Soit c la valeur à imposer à x_i : en posant $b_i + hc = hx_i + \sum A_{i,k}x_k$ avec $h \gg A_{i,k}$ on a $c \simeq x_i$.



(a) vue d'ensemble. (b) coupe transversale.

FIGURE 6.1: Cas test de la sphère.

$$\forall v \in \mathbb{C}^n, \|v\|_{D^{-1}} = \sqrt{\sum_{i=1}^{n} \frac{v_i \bar{v}_i}{|d_i|}}. \tag{6.4}$$

Cette normalisation de termes permet alors de mieux se rendre compte de la qualité réelle de la solution.

6.2 Cas test de la sphère

Le cas test de la sphère représente le maillage nécessaire au calcul du champ électromagnétique diffracté par une sphère. Elle est constituée d'un noyau conducteur parfait, d'une couche de matériau et d'une couche d'air. Le maillage initial est constitué d'un seul sous-domaine et ne représente que la couche de matériau et la couche d'air, le noyau conducteur parfait ne nécessitant pas de maillage. La géométrie maillée est présentée dans les figures 6.1(a) et 6.1(b) avec en vert le matériau et en rouge la couche d'air.

Ce maillage permet de contourner un des problèmes du couplage entre la méthode multigrille et le code de calcul : lors de l'itération globale, le calcul du champ lointain par les EID repose sur la surface externe du problème. En présence d'un unique matériau s'étendant jusqu'à la surface externe, l'augmentation de l'indice perturbe à la fois la résolution interne, mais aussi le calcul des EID. Dans le cas de la sphère, la couche d'air permet d'éviter ce cas de figure. En augmentant l'indice du matériau interne, tout en laissant l'indice de la couche externe (air) inchangé, seule la complexité de la résolution volumique est modifiée. Ce cas test permet alors de tester la capacité du solveur multigrille à augmenter automatiquement son nombre de mailles par longueur d'onde, tout en garantissant une résolution des EID de qualité. Quel que soit l'indice du matériau, la qualité de la SER calculée ne dépend alors que de la qualité de la solution volumique (calculée par la méthode multigrille).

TABLE 6.1: NMLO pour le cas test de la sphère avec une onde incidente de fréquence fixe. "niv. k" désigne le maillage obtenu après k raffinements.

fréquence	niv. 0	niv. 1	niv. 2	ϵ	μ	n	k
basse	12	17	23	2	1	1.41	4.44
basse	6	9	11	4	2	2.82	8.89
basse	3	4	6	8	4	5.64	17.77
basse	12	17	23	1+1i	1+1i	1.41	4.44
basse	6	9	11	2+2i	2+2i	2.82	8.89
basse	3	4	6	4+4i	4+4i	5.64	17.77
inconnues	14 625	57 909	231 045				
arête moy.	11mm	8mm	6mm				

Tous les tests réalisés seront effectués sur le même maillage original grossier avec une onde incidente basse fréquence. Les tests effectués consistent à faire varier l'indice du matériau interne tandis que le nombre de mailles par longueur d'onde est présentée dans le tableau 6.1. Comme les tests ont concerné les matériaux sans et avec pertes (perméabilité et permittivité à parties imaginaires non nulles), ces deux configurations sont présentées. Il a été fait en sorte que les indices avec et sans pertes coïncident.

Dans les cas où le nombre d'onde k vaut 4.44 (n=1.41), tous les maillages sont suffisants. Pour k=8.89, le niveau initial est alors légèrement sous-maillé, le nombre de mailles par longueur d'onde tombant en dessous de la barre de 7 (on rappelle que l'on considère un problème comme suffisamment maillé lorsque le nombre de mailles par longueur d'onde est supérieur à 7 et idéalement supérieur à 10 pour les hautes fréquences). Dans les mêmes conditions, les deux niveaux de raffinement seront considérés comme suffisamment maillés. Pour k=17.77, tous les maillages sont plus ou moins sous-maillés. Ce dernier test permet de voir l'amélioration apportée par le raffinement automatique du maillage et ce malgré des mailles de plus en plus "écrasées".

6.2.1 Évaluation du facteur de sous-relaxation pour le lisseur Jacobi

Il est montré que pour les équations de Maxwell, le lisseur Jacobi nécessite un facteur de sous-relaxation pour converger. Avant de mener les tests, une évaluation expérimentale du facteur de sous relaxation a été effectuée sur le cas de la sphère. Le multigrille à un niveau a été exécuté avec différents facteurs de sous-relaxation pour les itérations de Jacobi. La sphère a été soumise à un champ incident basse fréquence avec un matériau d'indice 1.41 (première ligne du tableau 6.1). Les figures 6.2 montrent l'évolution du résidu relatif pour un FMG à un niveau avec divers facteurs de sous relaxation.

Dans ces figures, on se focalise sur l'évolution du résidu sur le niveau fin, l'axe des abscisses ne représentant que des points de passage dans le code. Ces "étapes" correspondent à l'état initial en entrée du cycle en V (marqué par des croix), à chaque fin d'itération de Jacobi (carrés vides) et à l'état après correction grossière (carrés pleins). Entre le dernier pré-lissage et l'étape de correction grossière se trouvent les étapes de restriction, résolution directe, prolongation et correction de la solution. Dans la figure 6.2, on a présenté l'évolution d'un FMG classique (reconnaissable au résidu relatif initial différent de 1). Pour chacun de ces tests, 10 V-cycles

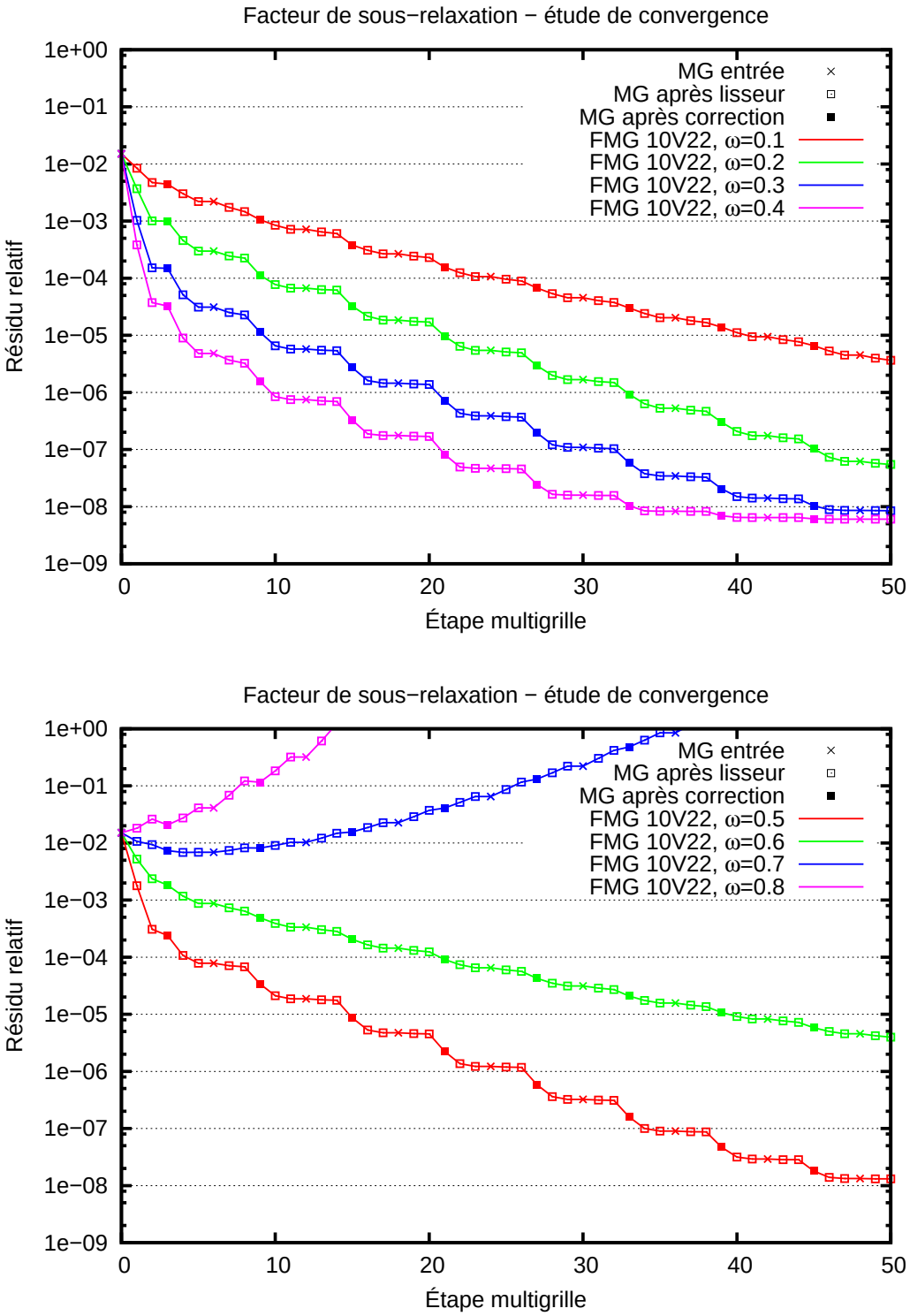


FIGURE 6.2: Tests avec différents facteurs de sous-relaxation pour la méthode de Jacobi.

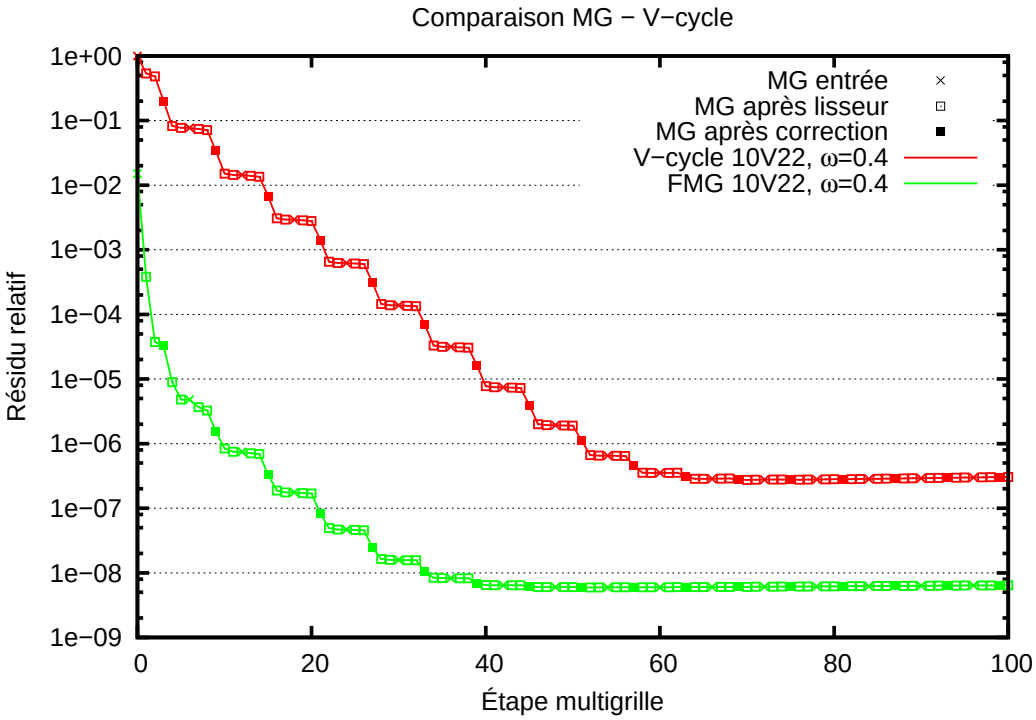


FIGURE 6.3: Comparaison de la convergence des V-cycles et FMG (1 niveau). Facteur de sous-relaxation fixé à 0.4.

avec 2 pré-lissages et 2 post-lissages ont été effectués (configuration nommée 10V2-2), seul le facteur de sous-relaxation étant modifié. Comme présenté dans ces figures, chaque facteur de relaxation a un effet distinct sur la convergence de la méthode. On peut remarquer que les facteurs de 0.1 à 0.6 font que la méthode multigrille converge, alors que tout indice supérieur diverge. Dans tous les cas où il y convergence, la méthode stagne à hauteur de 1e-8 en résidu relatif, le facteur 0.4 ayant la convergence la plus rapide. Du fait de la sensibilité du Jacobi aux valeurs propres du système, le facteur de sous-relaxation 0.3 sera souvent préféré afin d'augmenter la marge de sécurité. Bien que légèrement moins performant que 0.4 en termes de rapidité de convergence, ce facteur de sous-relaxation donne des résultats comparables.

6.2.2 Comparaison FMG et V-cycle

Comme présenté dans les chapitres précédents, le multigrille se décline sous diverses formes. Dans ces tests, nous comparons les cycles en V et FMG. Pour plus de simplicité, seuls les FMG modifiés seront utilisés (interpolation directe de la solution grossière vers le niveau fin). La seule différence entre le FMG et le cycle en V est alors l'estimation initiale de la solution : alors que le FMG se base sur une interpolation de la solution grossière, le cycle en V part d'une solution à zéro. Le résidu relatif en entrée de V-cycle est alors de 1.

La comparaison entre le FMG et le V-cycle présentée dans la figure 6.3 (même configuration que précédemment) permet de mettre en évidence la meilleure convergence de la méthode FMG. Alors que le V-cycle a un plateau de convergence au niveau de 1e-6, celui du FMG

se situe en 1e-8. La vitesse de convergence de ces deux courbes est différente : le FMG atteint son plateau en 40 étapes alors que le V-cycle l'atteint en 60 étapes. Les traitements effectués par ces deux méthodes étant strictement identiques, la meilleure convergence du FMG est principalement due à l'estimation initiale et est probablement liée au *near-null space* de l'opérateur. On suppose que la solution directe grossière interpolée est plus éloignée du *near-null space* que la solution initiale utilisée par le V-cycle (vecteur nul). Les composantes contenues dans le *near null space* entraînent un résidu quasi-nul (alors qu'on est "loin" de la solution) qui, une fois restreint sur la grille grossière, correspond à une erreur grossière nulle. La correction sur le niveau fin n'est alors pas en capacité d'annuler ces composantes, d'où une convergence qui "stagne". Cette hypothèse n'a pu être vérifiée expérimentalement, mais ce type de comportement est souvent observé dans les méthodes multigrilles appliquées aux équations de Maxwell. Malgré tout, ce phénomène peut être mis en évidence en comparant la convergence des méthodes multigrilles à la méthode de Jacobi.

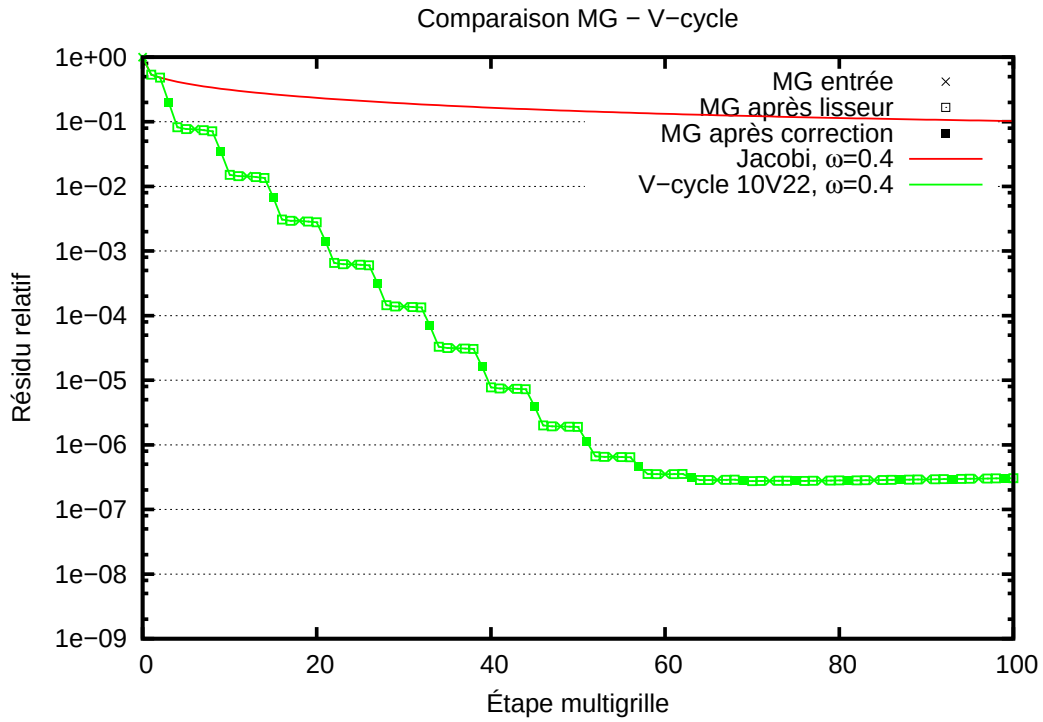
Dans les figures 6.4(a) et 6.4(b) le V-cycle et le FMG sont respectivement comparés au Jacobi. Pour que la comparaison soit équitable, les tests utilisant Jacobi reçoivent les mêmes estimations initiales de la solution que les cycles multigrilles : soit l'interpolation de la solution grossière ou soit le vecteur nul. Dans la première comparaison entre le V-cycle et Jacobi (figure 6.4(a)), on peut clairement retrouver le problème de convergence du multigrille au niveau du Jacobi. En partant d'une estimation nulle, le Jacobi converge très mal arrivant à un résidu relatif de 1e-1 au bout de 100 itérations. De plus, la vitesse de convergence décroît très rapidement. La convergence du cycle en V est alors principalement due à la correction grossière : les étapes de pré et post-lissages n'améliorent que très peu le résidu relatif (convergence en escalier).

En prenant comme estimation initiale l'interpolation de la solution grossière, les comportements du multigrille (FMG) et du Jacobi sont très différents (figure 6.4(b)). Les traitements réalisés étant les mêmes que réalisés précédemment, seule l'estimation initiale de la solution diffère. Cette modification a un impact important sur le comportement du Jacobi : le Jacobi converge désormais vers un plateau à 1e-7 en 40 itérations. Ce meilleur comportement du Jacobi est visible dans la convergence du FMG dont les premières itérations ne sont plus en escalier.

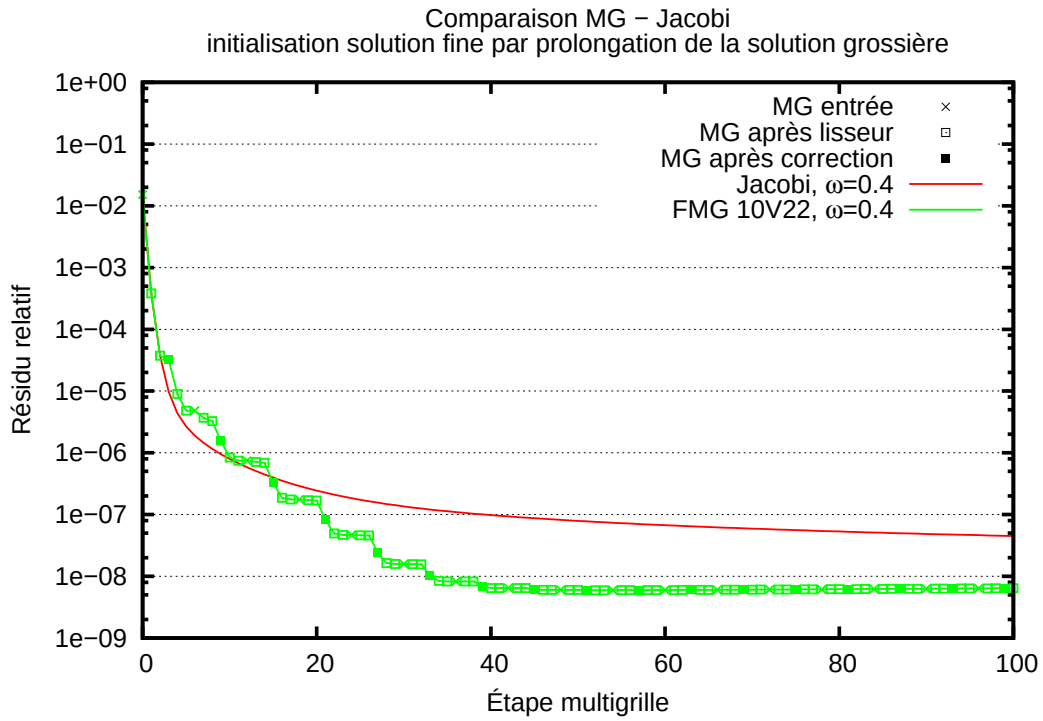
6.2.3 Effet de l'indice du matériau sur la convergence de la méthode multigrille

La convergence de la méthode multigrille est étudiée sur le cas test de la sphère. Pour ce test, on se focalise sur les basses fréquences et l'indice du matériau interne varie de 1.41 à 5.64 (cf. tableau 6.1). Pour tous ces matériaux, le FMG à 1 niveau a été utilisé. Nous ne nous intéressons ici qu'à la convergence de la méthode en termes de résidu relatif. Les résultats de convergence obtenus sont présentés dans la figure 6.5.

La figure 6.5(a) présente les résultats de convergence de la méthode multigrille pour les matériaux avec pertes (perméabilité et permittivité à partie imaginaire non nulle). On peut voir une dégradation de la convergence de la méthode avec l'augmentation de l'indice du matériau. En regardant le résidu relatif initial de chacune de ces courbes, on peut supposer que l'augmentation de l'indice réduit la qualité du calcul direct grossier. Cette hypothèse est soutenue par l'évolution du résidu au niveau des corrections grossières : pour n=5.64, la

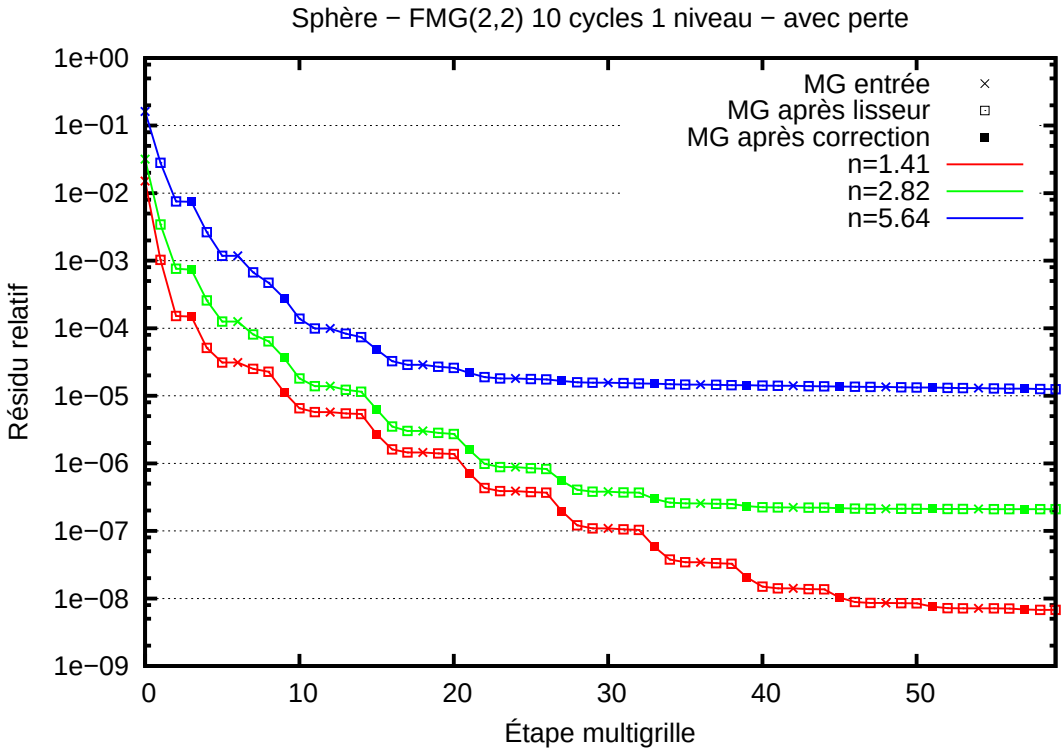


(a) Comparaison entre Jacobi et V-cycle.

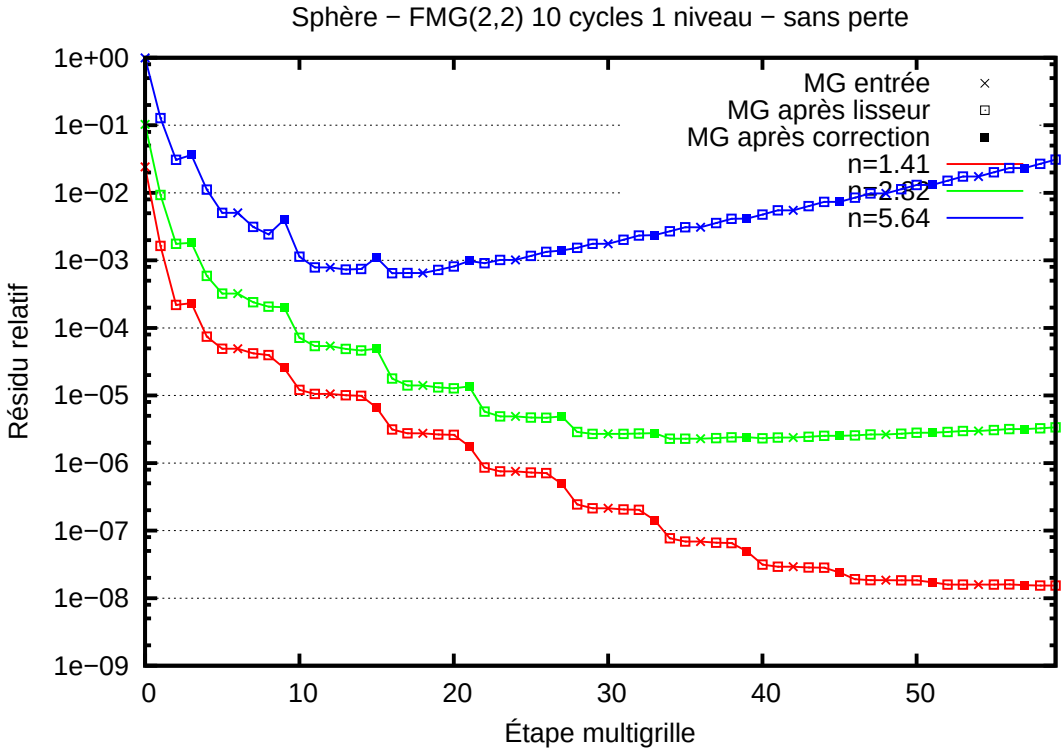


(b) Comparaison entre Jacobi et FMG.

FIGURE 6.4: Comparaison de Jacobi avec les méthodes multigrilles.



(a) Matériau avec pertes.



(b) Matériau sans perte.

FIGURE 6.5: Comparaison de convergence du FMG à un niveau selon le matériau.

correction grossière n'agit plus au bout de 2 itérations, contre 8 itérations pour n=1.41. Une fois que la correction grossière n'apporte plus d'amélioration, seul le Jacobi influence le résidu, d'où le plateau de convergence.

Pour des matériaux sans perte (6.5(b)), le comportement est encore plus mauvais : non seulement on retrouve la différence de convergence selon l'indice du matériau, mais désormais la méthode diverge au bout d'un certain temps. En se focalisant sur les indices 2.82 et 5.64, on peut clairement voir la divergence apparaître plus ou moins tard (la courbe pour n=1.41 diverge elle aussi mais n'apparaît pas dans ce graphique). Ce comportement est similaire à celui observé par Pesqué [46] pour le problème de Helmholtz.

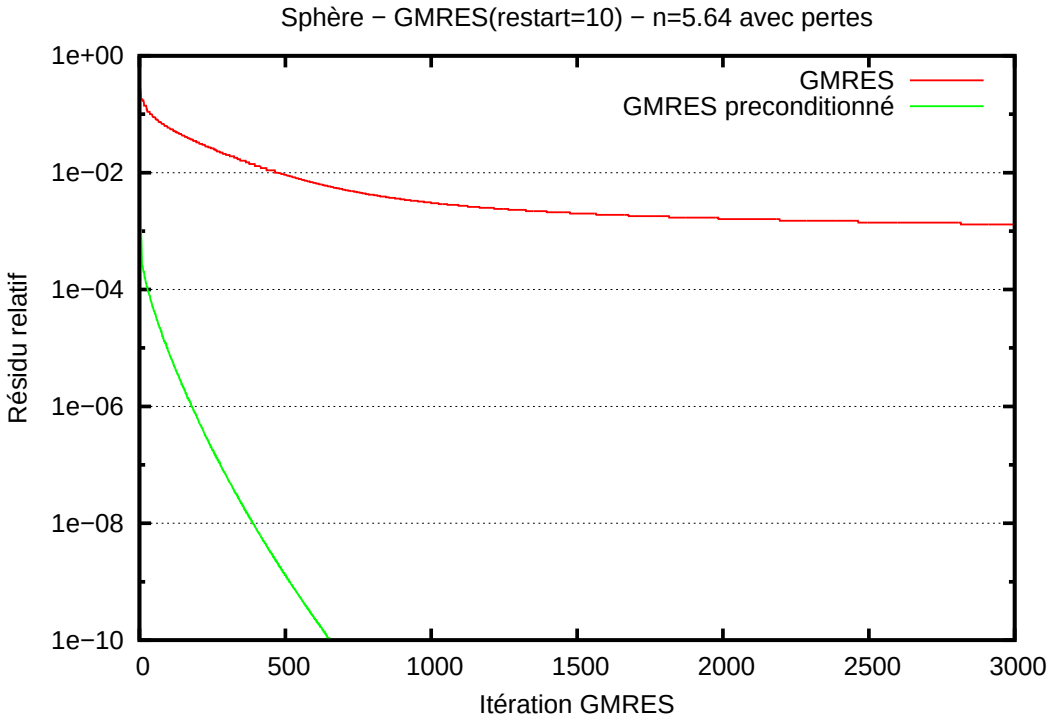
Ces tests ont donc permis de classifier les problèmes selon leurs difficultés :
– plus l'indice du matériau est élevé, plus le problème est dur numériquement ;
– à indices équivalents, un problème sans perte est plus difficile à résoudre qu'un problème avec pertes.

6.2.4 Comportement du GMRES préconditionné

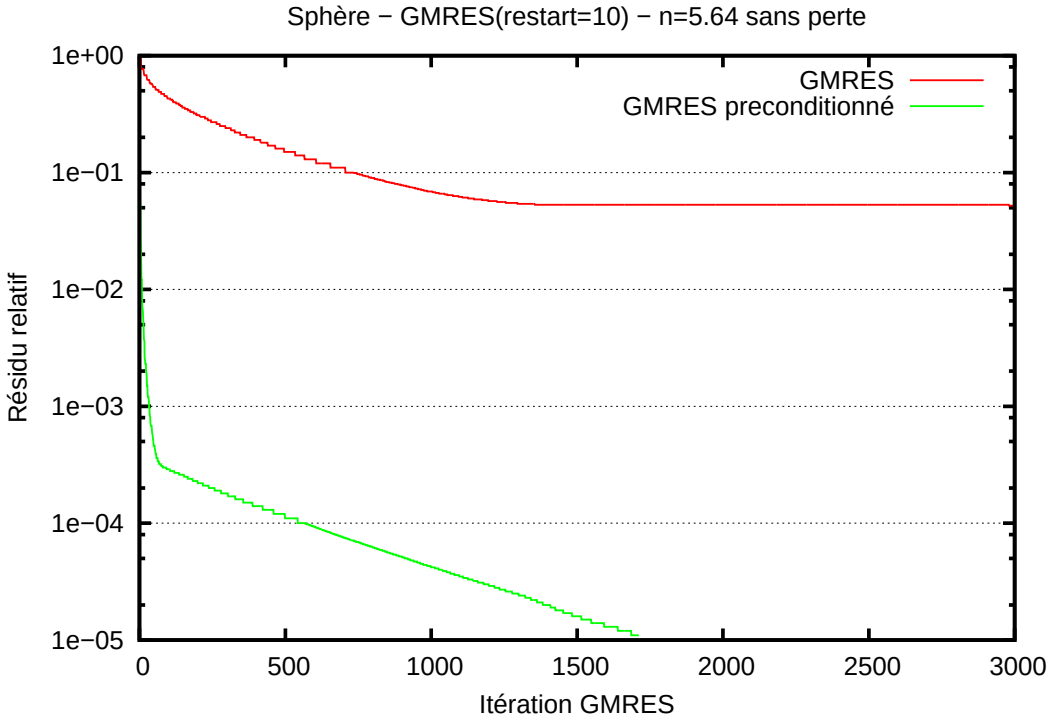
Comme présenté précédemment, dans les cas où l'indice est trop important ou que le matériau est sans perte, la méthode multigrille converge mal. Dans ces conditions, la méthode multigrille ne peut être utilisée seule dans l'itération globale. On la remplace alors par un GMRES préconditionné par la méthode multigrille. Dans cette partie, nous allons comparer le comportement du GMRES classique avec celui du GMRES préconditionné dans les cas de matériaux d'indices n=5.64 (avec et sans pertes). La figure 6.6 présente les résultats obtenus.

Dans les deux cas présentés, la différence entre le GMRES classique et préconditionné est importante. Avec ou sans pertes, le GMRES classique converge mal. Au bout de 3000 itérations la norme du résidu relatif se situe entre 1e-1 et 1e-2. Comme pour la méthode multigrille, on peut remarquer que le comportement du GMRES classique est meilleur sur les matériaux avec pertes que sans perte. En effectuant les mêmes tests, mais en préconditionnant le GMRES par un multigrille (10 V-cycles à 1 niveau avec 2 pré-lissages et 2 post-lissages, que l'on notera 10V(2,2)), la convergence est meilleure. En termes d'itérations GMRES, la convergence en 1e-10 est atteinte en 600 itérations pour le cas avec pertes et atteint 1e-5 en 1700 itérations pour le cas sans perte. Par contre, l'itération de GMRES préconditionnée est bien plus chère que l'itération du GMRES classique : alors que le GMRES classique n'exécute qu'un seul produit matrice-vecteur par itération, le GMRES préconditionnée nécessite ici 40 produits matrice-vecteur par itération (2 pré-lissages et 2 post-lissages répétés 10 fois, la résolution grossière étant considérée comme négligeable). En comparant ces résultats en termes de produits matrice-vecteurs, la méthode GMRES préconditionnée est toujours supérieure à la méthode classique en termes de rapidité de convergence et de niveau du plateau. De plus, le MG en temps que préconditionneur comporte habituellement seulement 1 à 2 cycles en V.

Pour les cas complexes, où le multigrille en tant que solveur ne donne pas de résultats satisfaisants, on pourra donc utiliser la méthode GMRES préconditionnée qui s'est avérée ici performante. Dans la littérature sur Helmholtz [18, 55] (en particulier dans le cas sans perte), un traitement similaire est appliqué sur un opérateur *translaté*. Cette méthode consiste à ajouter artificiellement des pertes en translatant l'opérateur de préconditionnement par un Laplacien complexe. La translation complexe permet d'améliorer le convergence du GMRES préconditionné par MG. Bien que donnant des résultats intéressants, la méthode du précon-



(a) Convergence du GMRES pour la sphère. Matériau avec pertes d'indice n=5.64.



(b) Convergence du GMRES pour la sphère. Matériau sans perte d'indice n=5.64.

FIGURE 6.6: Comparaison entre GMRES non préconditionné et préconditionné par multigrille (restart=10).

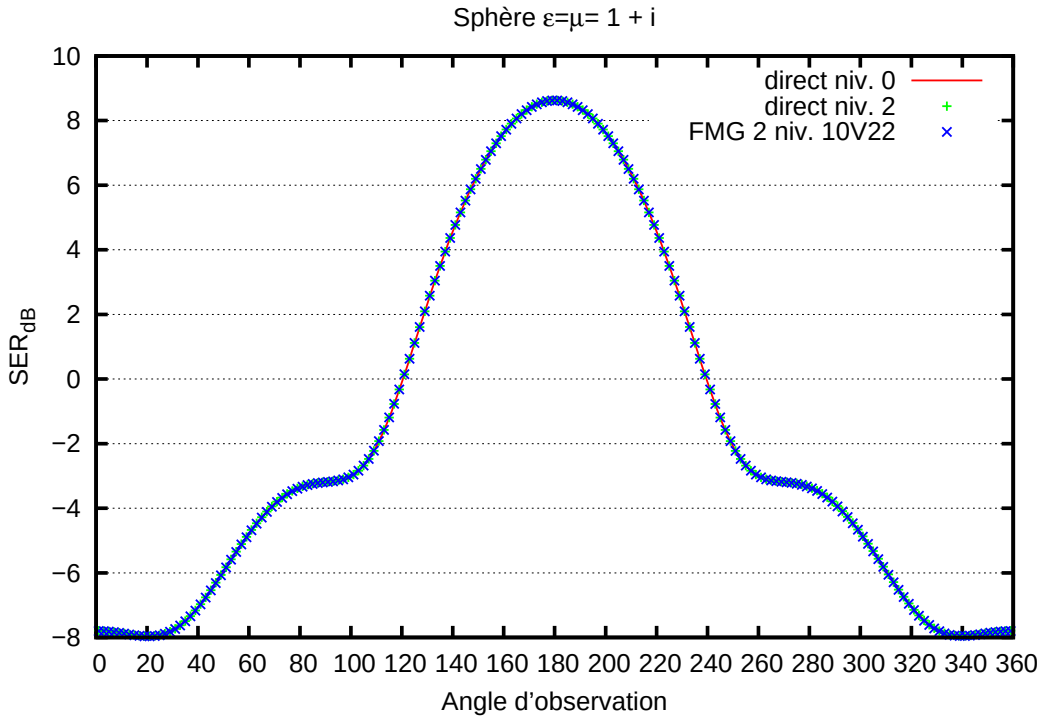


FIGURE 6.7: SER de la sphère calculée sur les niveaux 0 et 2 à basse fréquence (onde incidente à 0 degré).

ditionneur translaté n'a pas été implémentée.

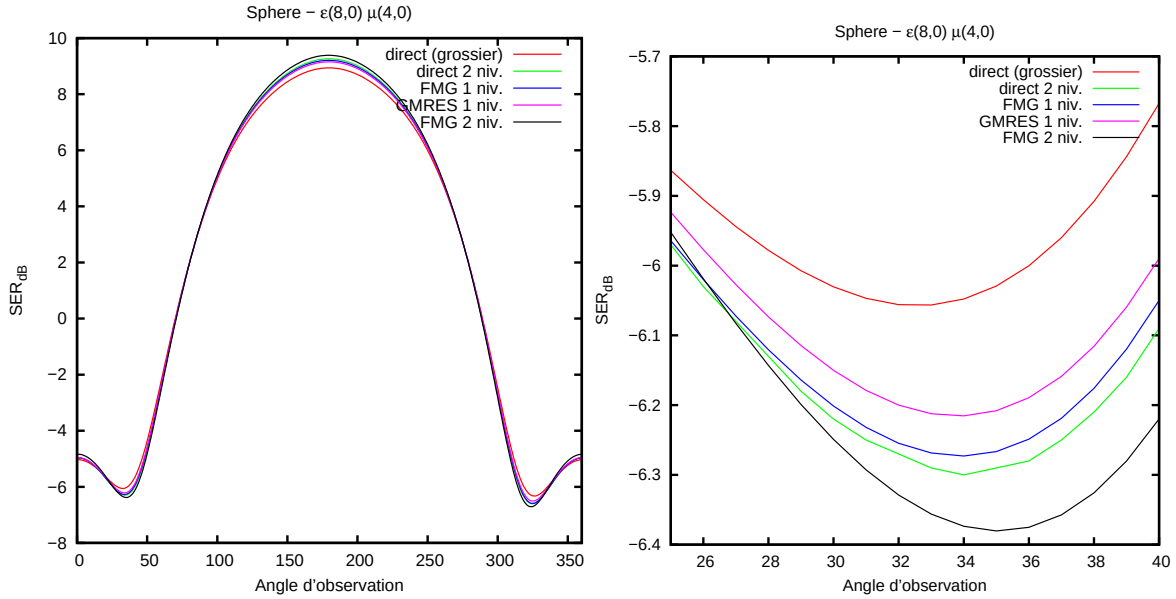
6.2.5 Validation du calcul de SER

Nous allons ici vérifier que la méthode multigrille permet de calculer des solutions équivalentes à celles calculées avec le solveur direct. Pour cela, nous comparons les calculs effectués sur la sphère dans les cas suivants :

– solveur direct sur le maillage initial ;
– solveur direct sur le maillage raffiné deux fois ;
– multigrille à deux niveaux.

On utilise ici le cas test de la sphère éclairée par une onde basse fréquence avec un matériau d'indice 1.41 ($\mu = \varepsilon = 1 + i$). Le NMLO est de 12 sur le niveau grossier et de 23 sur le niveau fin. On compare les SER calculées après 9 itérations globales. Les résultats sont présentés dans la figure 6.7. Le maillage utilisé pour la résolution directe au niveau 2 est le maillage calculé automatiquement par le multigrille. Ainsi, les maillages de niveau 2 utilisés pour le direct et lors du multigrille sont identiques. Pour ce faire, une exécution préalable (séquentielle) du code a permis d'extraire le maillage niveau 2. Cette solution expérimentale n'est donc pas utilisable en production.

Dans cette figure, les trois courbes sont confondues : le maillage grossier étant suffisamment maillé, la SER y était déjà bonne. Toutefois ce test permet de montrer que la distorsion des éléments a peu d'incidence sur la SER calculée : après deux raffinements, le ratio arête max.



(a) n = 5.64 sans perte.

(b) n = 5.64 sans perte (zoom).

FIGURE 6.8: Sphère - SER basse fréquence - indice =5.64 sans perte.

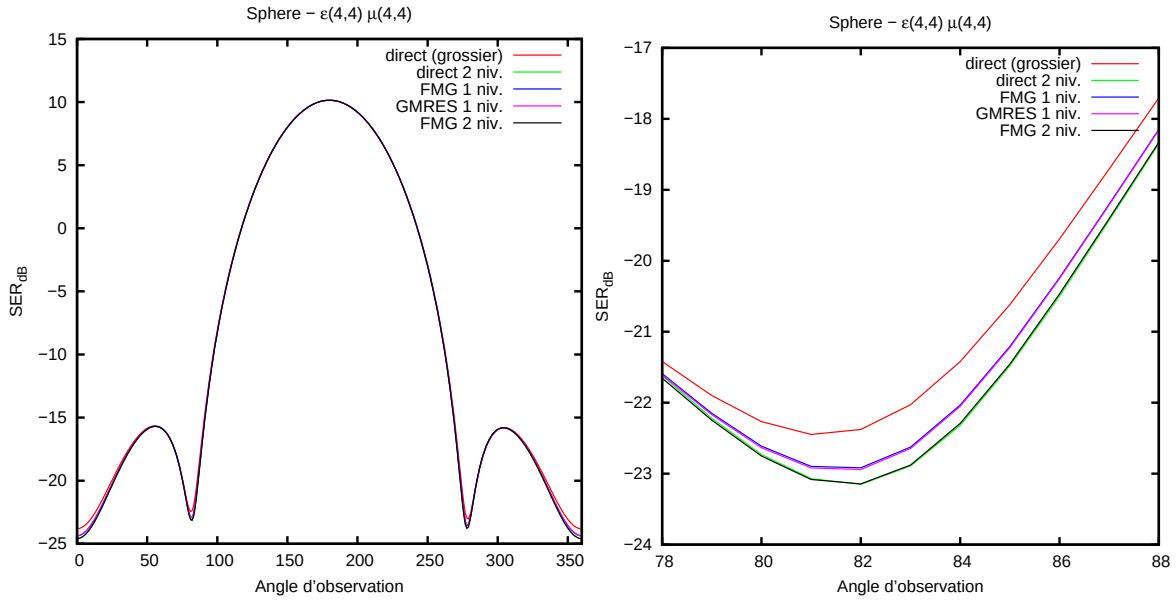
sur arête min. est de 30 alors qu'il est de 3 sur le niveau grossier. Malgré cette distorsion des éléments, la solution calculée correspond au résultat attendu.

6.2.6 SER avec augmentation de l'indice du matériau

On utilise le cas test de la sphère afin d'augmenter progressivement la difficulté du problème. En se basant sur les données présentées dans le tableau 6.1, on fait varier l'indice du matériau interne du domaine (représenté en vert dans la figure 6.1) tout en fixant la fréquence de l'onde incidente. En conservant un maillage de la surface externe suffisamment maillé pour calculer une SER de qualité, le problème volumique est de plus en plus complexe.

On a traité deux cas de figure : avec pertes et sans perte. Le fait d'avoir une partie imaginaire non nulle pour la perméabilité et la permittivité permet d'obtenir un système mieux conditionné. De ce fait, la convergence de la méthode est de meilleure qualité et la SER s'en trouve améliorée. Les figures 6.8 et 6.9 représentent les calculs de SER réalisés sur la sphère en exécutant soit une résolution directe (maillage grossier raffiné deux fois), soit un FMG (1 et 2 niveaux) ou soit un GMRES (à 1 niveau préconditionné par multigrille). Ce dernier calcul consiste à réaliser un GMRES sur le niveau fin, tout en préconditionnant le système par l'emploi de cycles en V.

Les résultats de SER pour les indices n=1.41 et n=2.82 étant totalement confondus, ils ne seront pas présentés ici. Seuls les tests à n=5.64 présentent une différence de SER pour les différents niveaux. La figure 6.8(a) montre une différence entre la SER calculée sur les différents niveaux de maillage pour un matériau sans perte d'indice n=5.64. Le NMLO de ce problème sur le niveau grossier étant de 3 (tableau 6.1), la méthode directe donne une SER non satisfaisante. En la comparant avec la SER calculée sur le maillage raffiné deux fois (NMLO=6), on peut clairement voir la différence entre les deux courbes. Que ce soit sur



(a) n = 5.64 avec pertes.

(b) n = 5.64 avec pertes (zoom).

FIGURE 6.9: Sphère - SER basse fréquence - indice =5.64 avec pertes.

les hauts ou les bas niveaux de SER, la différence est de l'ordre de 1dB. Bien que le niveau grossier ne soit pas suffisamment maillé pour calculer une SER satisfaisante, nous l'utiliserons comme base de la méthode multigrille. Les FMG à un et deux niveaux, ainsi que le GMRES à un niveau ont été exécutés sur ce problème. Les résultats obtenus sont présentés eux aussi dans la figure 6.8. Afin de mieux voir la différence entre ces courbes, la figure 6.8(b) est un agrandissement de la figure 6.8(a). Sur cette dernière figure, on peut alors voir que les SER calculées par FMG et GMRES s'éloignent de la SER calculée par le solveur direct sur le niveau grossier. Mieux encore, elles se rapprochent de ce qui a été calculé par méthode directe sur le deuxième niveau de raffinement. On peut alors en conclure que les méthodes FMG et GMRES multi-niveaux apporte une amélioration pour le calcul de la SER : bien qu'ils se basent sur un maillage grossier à faible NMLO ("limite" en termes de NMLO), les raffinements utilisés par le FMG et le GMRES calculent une SER de meilleure qualité.

Les mêmes tests ont été réalisés pour un matériau avec pertes. Les résultats sont présentés dans la figure 6.9. Le problème étant moins complexe que dans le cas du matériau sans perte, on peut observer une amélioration pour les fortes valeurs de SER. Dans la figure 6.9(a), on peut voir que seules les faibles valeurs de SER diffèrent. Dans la figure 6.9(b), on a grandi la zone présentant les faibles valeurs de SER. Comme dans le cas sans perte, on peut voir l'amélioration portée par les méthodes multigrilles : cette fois-ci, les méthodes à 1 niveau se situent proche de la SER calculée par le solveur direct sur le second niveau de raffinement. Sur la même figure, cette dernière SER et celle calculée par le FMG à deux niveaux sont confondues. On montre ainsi que dans le cas de matériaux avec pertes, le comportement du multigrille est très satisfaisant et permet de calculer une SER correcte malgré un NMLO de 3 sur le maillage de niveau grossier.

Tous ces tests ont permis de mettre en évidence le bon fonctionnement de la méthode multigrille dans l'itération globale du code de calcul. On a aussi mis en évidence l'amélioration

TABLE 6.2: NMLO pour le cas test de l'haltère. Les données concernent le maillage initial ainsi que 4 niveaux de raffinement pour un indice de matériau fixé à 1.41.

fréquence	niv. 0	niv. 1	niv. 2	niv. 3	niv. 4	niv. 5	n	k
1 GHz	20	29	38	49	62	78	1.41	29.62
2 GHz	10	14	19	24	31	39	1.41	59.24
3 GHz	7	10	13	16	21	26	1.41	88.86
5 GHz	4	6	8	10	12	16	1.41	148.1
inconnues	1.2 E6	5.2 E6	21.1 E6	84.6 E6	338.5 E6	1.3 E9		
arête moy.	10mm	7mm	5mm	4mm	3mm	2mm		

apportée à la SER par l'utilisation des niveaux de raffinement pour des maillages initiaux à faible NMLO.

6.3 L'haltère

Le cas test de l'haltère disposant d'un nombre d'inconnues plus important, il nous a permis de mener à bien des tests à plus grande échelle. Contrairement à la sphère présentée dans la précédente section, l'haltère n'est pas recouverte d'une couche d'air. De ce fait, la SER est directement calculée sur la surface du matériau. Cette spécificité fait que l'augmentation de la difficulté volumique est répercutée sur le calcul des EID : quel que soit le niveau de raffinement interne, la surface externe est constante. Cela est dû au raffinement des éléments finis retenu. La diminution du NMLO dans le volume est accompagnée d'une diminution du NMLO sur les EID. La comparaison de SER est alors difficile à analyser car la source d'erreur peut être due soit au calcul volumique (FMG, GMRES, direct), soit au calcul surfacique sous-maillé. Ce test va tout de même être utilisé pour déterminer le comportement du multigrille sur des cas complexes rencontrés lors de l'augmentation de la fréquence de l'onde incidente.

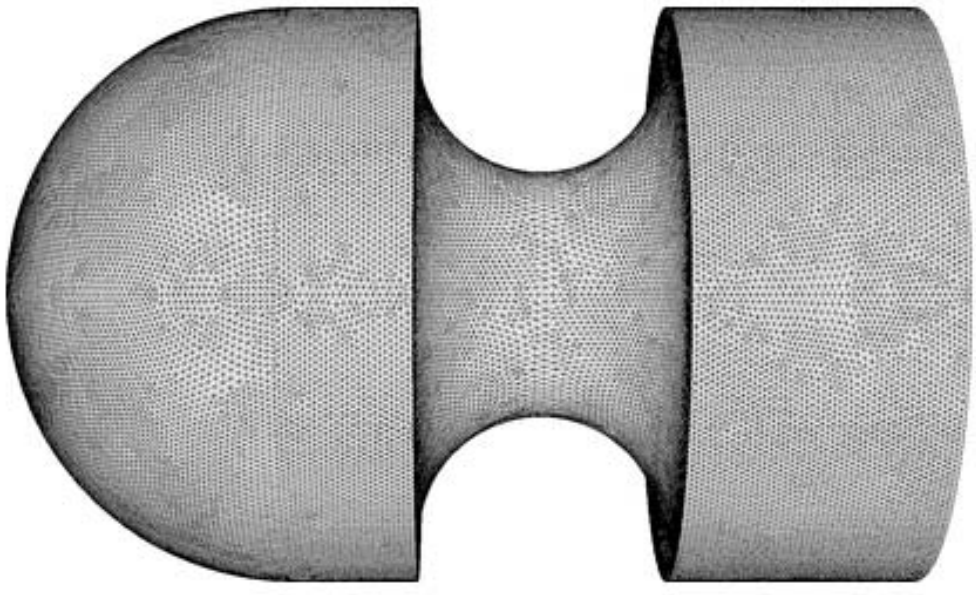
La géométrie de l'haltère est présentée dans la figure 6.10. Contrairement au cas test de la sphère présenté précédemment, on considère désormais un indice de matériau fixe (zone rouge dans la figure 6.10(b)) d'indice 1.41 avec pertes ($\varepsilon = \mu = 1 + i$) et une onde incidente de fréquence variant de 1GHz à 5GHz. Les nombres de mailles par longueur d'onde et les nombres d'inconnues sont présentées dans le tableau 6.2.

6.3.1 Balayage fréquentiel

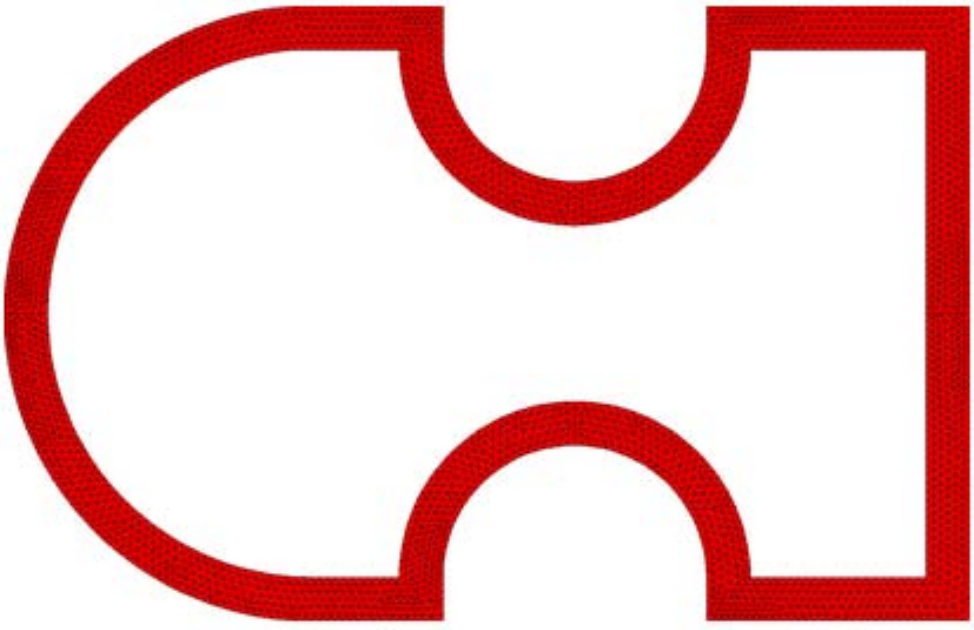
Convergence du multigrille

Le premier test effectué sur cet objet se fait avec une onde incidente à 1GHz avec un indice de matériau à n=1.41 ($\varepsilon = \mu = 1 + i$). À cette fréquence, le maillage initial est très suffisamment maillé et permet de faire une première vérification de la convergence de la méthode multigrille. Les tests sont ensuite effectués en augmentant progressivement la fréquence de l'onde incidente jusqu'à atteindre les 5GHz, réduisant le NMLO à 4 sur le niveau grossier. Nous allons voir la dégradation progressive de la convergence de la méthode multigrille avec l'augmentation de la fréquence et donc du nombre d'onde k.

Les figures 6.11 et 6.12 représentent les convergences des méthodes FMG et V-cycles pour les

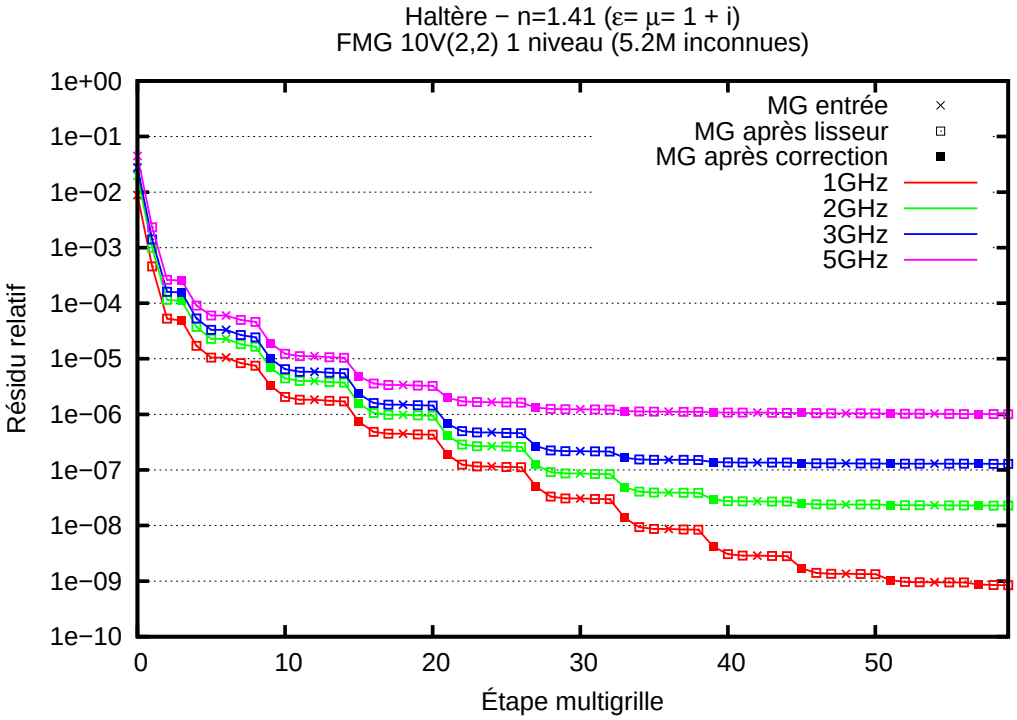


(a) vue d'ensemble.

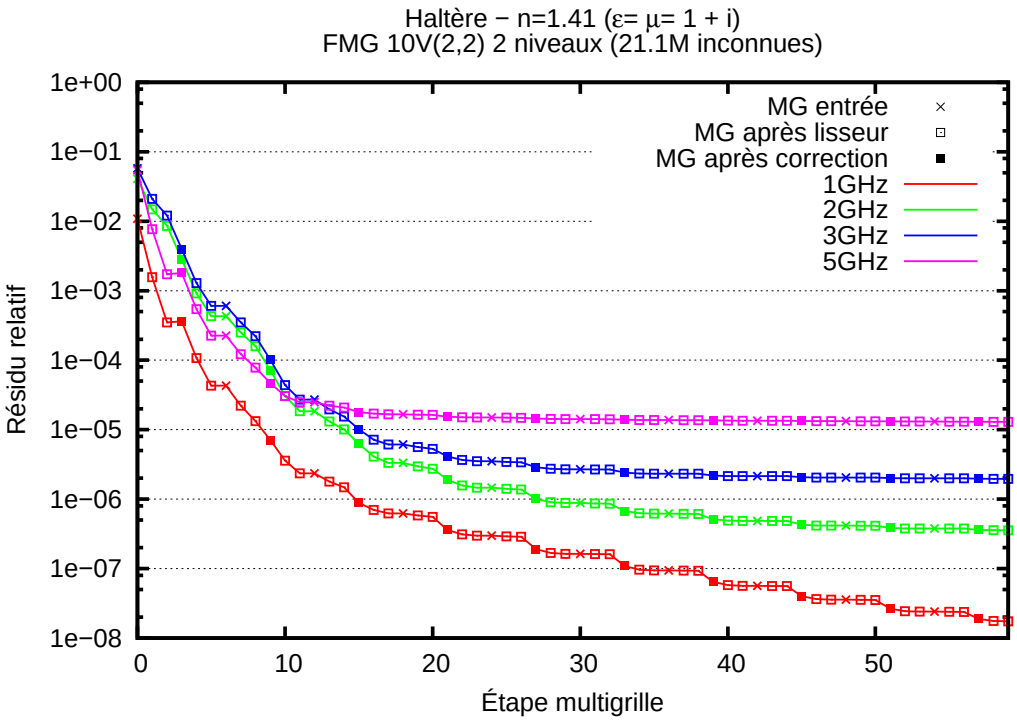


(b) coupe transversale.

FIGURE 6.10: Cas test de l'haltère.



(a) Haltère, FMG 1 niveau.



(b) Haltère, FMG 2 niveaux.

FIGURE 6.11: Convergence du FMG sur l'haltère pour un matériau d'indice n=1.41.

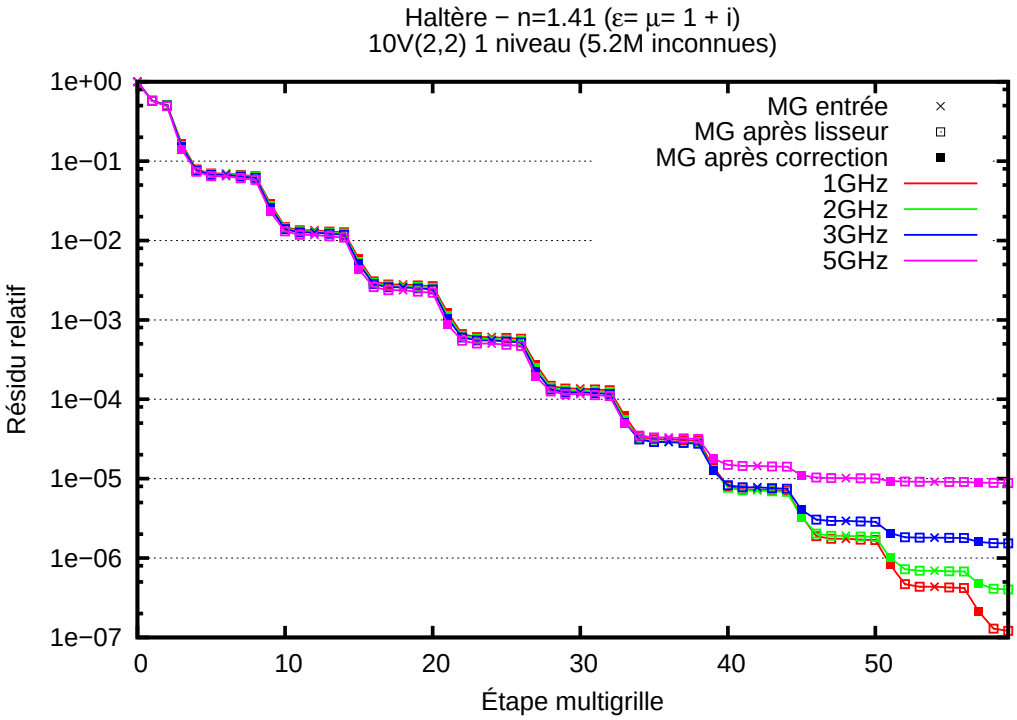


FIGURE 6.12: Convergence du V-cycle sur l'haltère pour un matériau d'indice n=1.41.

différentes fréquences testées. Le graphique 6.11(a) présente les résultats obtenus par FMG à 1 niveau. D'après le tableau 6.2, tous les maillages de ce niveau sont suffisamment maillés pour les fréquences utilisées. Seul le cas à 5GHz est légèrement sous maillé. On peut observer que la complexité du problème détériore la convergence de la méthode : alors qu'à 1GHz le FMG à 1 niveau converge vers un résidu relatif de 1e-9, il ne converge qu'à 1e-6 à 5GHz. Tous les tests atteignent leur plateau à peu près en même temps (40 à 50 étapes multigrille).

Le second test se déroule dans les mêmes conditions avec deux niveaux de raffinement. Les résultats obtenus sont présentés dans la figure 6.11(b). Les NMLO sont désormais de 38, 19, 13 et 8, ce qui implique que le maillage est bien maillé pour 5GHz et sur-maillé pour toutes les autres fréquences. D'après les résultats, on peut alors voir une convergence rapide du cas à 5GHz (10 à 20 étapes multigrille). C'est un effet que nous attendions : l'augmentation du NMLO à nombre d'onde constant améliore la convergence [41]. Néanmoins ce point ne se vérifie pas pour toutes les fréquences. Plus particulièrement, le cas à 1GHz converge vers un résidu de 1e-8 mais n'atteint pas son plateau en 10 itérations (60 étapes multigrille). On peut alors en conclure que l'aplatissement des éléments finis entraîne une dégradation de la vitesse de convergence de la méthode, alors qu'un maillage "correct" (5GHz à 2 niveaux) améliore le comportement de la méthode FMG. Il faut aussi remarquer que les résidus relatifs sont supérieurs à ceux obtenus par l'utilisation d'un unique niveau de raffinement. Ce comportement peut être dû au sur-maillage, à la forme des éléments (très aplatis) ou à l'accumulation d'erreurs dans la solution (solutions calculées sur respectivement 5.2 et 21.1 millions d'inconnues).

Pour finir cette étude, les mêmes tests ont été menés avec des V-cycles (estimation initiale

de la solution nulle) sur un seul niveau. Les résultats sont présentés dans la figure 6.12. De manière similaire à ce que l'on a présenté sur la sphère, la convergence de la méthode est en escalier (le Jacobi ne réduit que très peu la norme du résidu). On peut à nouveau noter que le résidu relatif obtenu se dégrade avec la complexité du problème. À 5GHz, on obtient désormais un résidu relatif de 1e-5 par V-cycle, contre 1e-6 par FMG.

Calcul de SER

Le calcul de la SER de l'haltère a été réalisé en utilisant la méthode multigrille. La première étape consiste à vérifier la qualité de la SER calculée. Pour ce faire, nous nous sommes intéressés au problème à 3GHz. La vérification consiste alors à calculer la SER par FMG et V-cycles à deux niveaux et par solveur direct sur le niveau 2. La figure 6.13 présente les résultats obtenus.

Dans cette figure on présente les trois résultats obtenus, les croix représentant la SER de référence calculée directement sur le niveau 2. Les courbes direct et FMG sont confondues, prouvant l'efficacité de la méthode FMG. Toutefois, la SER calculée par V-cycle diffère des autres SER calculées, en particulier pour les faibles valeurs de SER (figure 6.13(b)). On peut alors en conclure que la convergence du V-cycle, présentée dans la figure 6.12, n'est pas suffisante pour le calcul de ces faibles valeurs de SER. Ce résultat prouve la supériorité de la méthode FMG par rapport aux V-cycles. Dans le cadre des résolutions électromagnétiques à hautes fréquences, ce résultat nous conforte sur la pertinence du choix de la méthode FMG.

Dans l'exemple précédent, le V-cycle calcule une SER non satisfaisante. Toutefois, pour des fréquences plus faibles, la convergence du V-cycle suffit à calculer la SER correctement. L'utilisation de V-cycles en tant que solveur sera alors réservée pour les cas simples à basses fréquences ou faibles indices de matériaux.

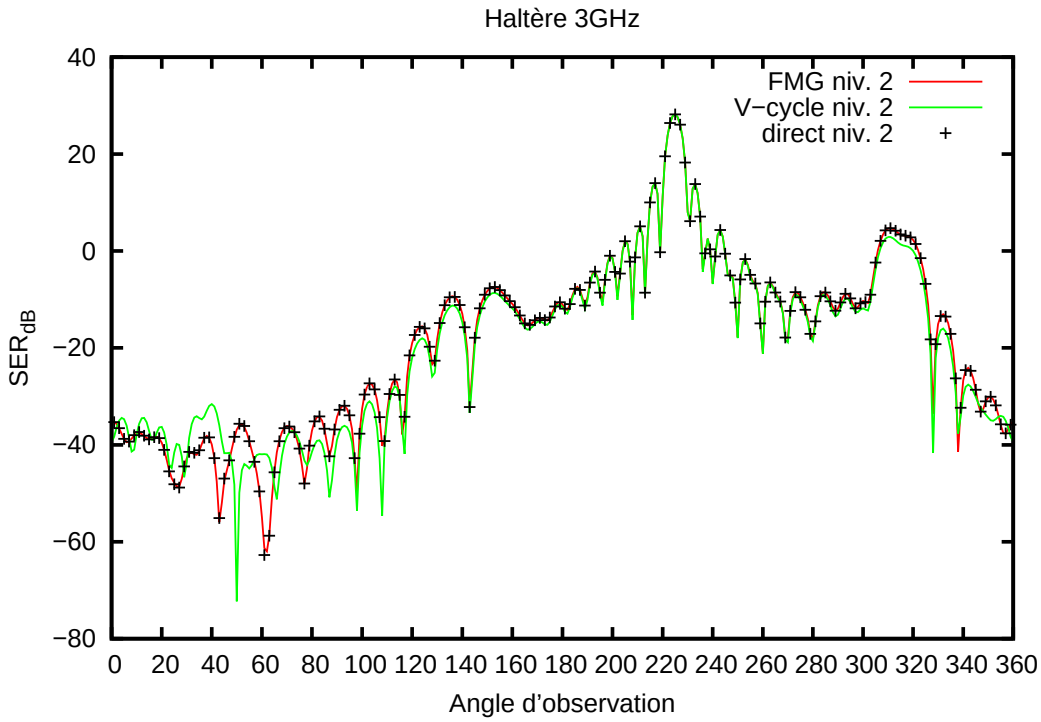
6.3.2 Scalabilité de la méthode

Le principal avantage de la méthode multigrille est sa scalabilité. Dans cette partie, nous étudierons les performances de la méthode FMG sur l'haltère. Nous focaliserons notre étude sur un FMG constitué de 10 itérations de V-cycles avec deux pré et post-lissages, avec une exception pour le cas de plus grande taille où nous ferons 20 cycles.

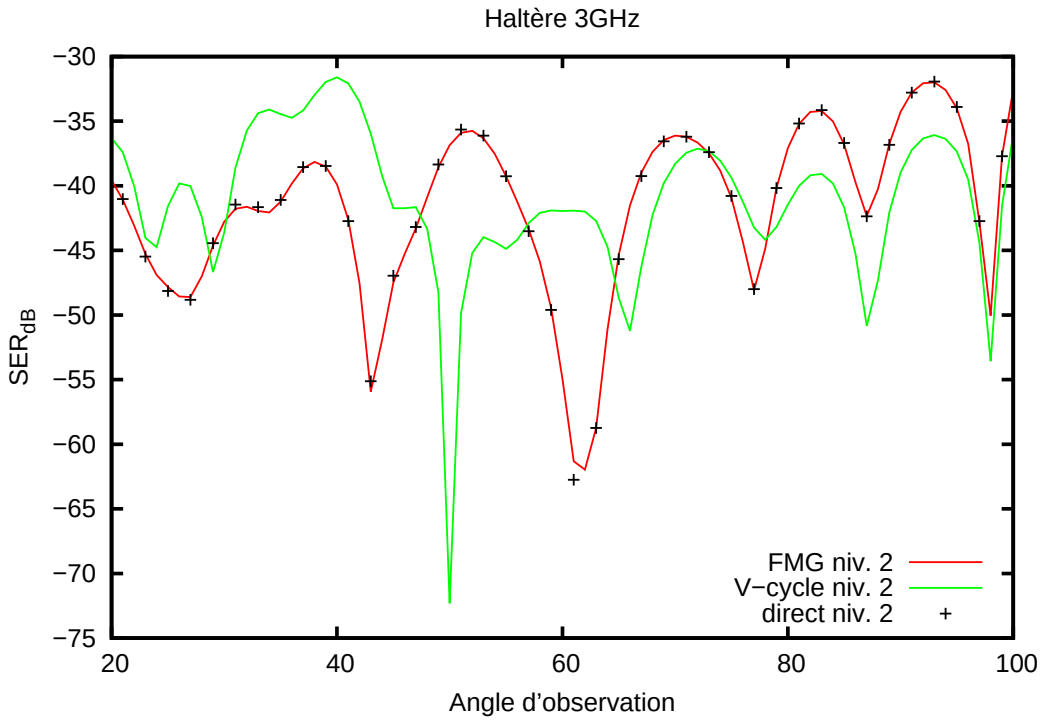
Comparaison FMG et solveur direct

Tout d'abord, nous comparons les temps des méthodes directes et multigrilles à nombre d'inconnues équivalent. Pour ce faire, un FMG à deux niveaux est exécuté sur le maillage de l'haltère à 1.2 millions d'inconnues, et le direct sur le même maillage raffiné deux fois. Au final, les deux méthodes calculent une solution à 21.1 millions d'inconnues. La comparaison sera faite pour un calcul à 128 cœurs de calcul. Le tableau 6.3 présente les temps (en secondes) obtenus pour les deux méthodes.

Dans ce tableau, les deux premières colonnes représentent le nombres d'inconnues pour chacun des solveurs. Les colonnes suivantes représentent respectivement le temps de prétraitement de la méthode directe (renumérotation et distribution), le temps de factorisation et le temps de résolution directe. Ces informations sont relatives au niveau sur lequel le solveur direct est



(a) Observations de 0 à 360 degrés.



(b) Zoom sur les faibles valeurs de SER.

FIGURE 6.13: SER de l'haltère à 3GHz.

TABLE 6.3: Haltère - 128 cœurs de calcul - Comparaison en temps solveur direct et FMG.

Solveur	Grossier	Fin	Renum.	Facto.	Solve	10V	Conv.	Total
Direct	–	21 125 245	137	114	0.66	–	–	251
FMG(2,2)	1 288 825	21 125 245	23	60	0.10	95	1e-6	178

exécuté : le FMG n'exécute le direct que sur le niveau grossier alors que l'on travaille sur 21.1 millions d'inconnues dans l'autre cas. Les deux colonnes *10V* et *Conv* ne concernent que le FMG. La première de ces colonnes représente le temps mis pour exécuter 10 V-cycles complets (temps de lissages et de corrections grossières inclus). Notons que dans les 95 secondes sont inclus les temps de 10 résolutions directes grossières. La seconde colonne représente quant à elle le niveau de convergence (norme du résidu relatif) atteint par la méthode multigrille en 10 itérations.

La méthode multigrille montre un avantage certain par rapport à la méthode directe : la consommation mémoire du FMG est bien inférieure à celle du solveur direct et le temps de calcul est meilleur. Alors que la méthode directe sur le niveau fin nécessite les non-zéros d'un système à 21.1 millions d'inconnues, la méthode multigrille nécessite pour sa part les non-zéros du système à 1.2 millions d'inconnues, la diagonale de la matrice du niveau de maillage fin et le vecteur itéré sur le maillage fin (vecteurs de taille 21.1 millions).

Stabilité de la convergence

On a testé le fait que la méthode multigrille donne les mêmes résultats indépendemment du nombre de cœurs utilisés. Pour cela, on a regardé l'évolution du résidu relatif pour le même cas test (haltère) sur 12, 32, 64 et 128 cœurs avec trois niveaux de raffinement (calculs avec 1.2 millions d'inconnues sur le niveau grossier et 84.6 millions sur le niveau fin). Tous ces tests ont donné les mêmes résultats, en convergeant tous vers 1e-6 au bout de 10 itérations de V-cycles.

Scalabilité forte

On teste ici la scalabilité forte : pour un problème donné, on fait varier le nombre de cœurs. Dans un premier temps, on utilise le cas test de l'haltère sur lequel on tourne un FMG à trois niveaux. Les résultats obtenus sont présentés dans le tableau 6.4. Dans ce test, le maillage du niveau grossier comporte 1.2 millions d'inconnues, alors que le maillage du niveau fin comporte 84.6 millions d'inconnues.

Note : Les tableaux 6.4, 6.7 et 6.8 comportent des données provenant de différentes exécutions du code. Selon le niveau d'optimisation, les performances du code sont variables. L'instabilité de la machine (phase de démarrage de TERA100) sur laquelle ces temps ont été mesurés elle aussi contribue aux écarts observables entre ces exécutions. Par contre, les données contenues au sein d'un même tableau ont été exécutées dans les mêmes conditions (machine et compilation identiques) et sont alors comparables. Pour s'assurer d'une quantité de mémoire suffisante pour chaque exécution, tous ces tests n'utilisent que deux cœurs par nœud. Les nœuds TERA100 comportant 32 cœurs, les 30 cœurs restant sont inactifs.

Les temps présentés dans ce tableau sont les suivants :

TABLE 6.4: Haltère 84.6 millions d'inconnues (grossier : 1.2 millions) - FMG(2,2) 3 niveaux - temps en secondes.

# cœurs	Préparation	Calcul CSC	Assemblage	Factorisation	Résolution	10V
16	123.98	0.48	0.42	79.62	0.55	2285.58
32	118.67	0.21	0.23	82.72	0.30	1369.26
64	103.10	0.15	0.12	68.18	0.19	710.57
128	102.69	0.08	0.06	60.94	0.11	399.23
256	130.32	0.03	0.03	75.27	0.21	194.42
512	170.97	0.01	0.01	78.81	0.06	100.17

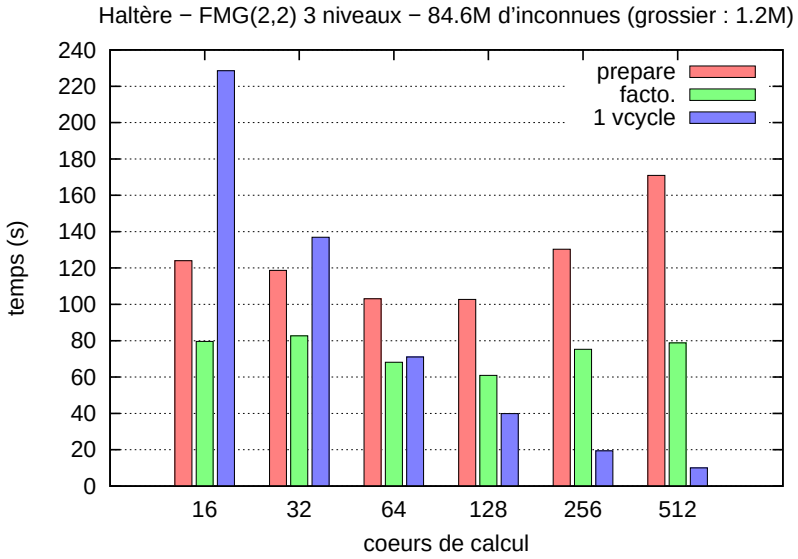


FIGURE 6.14: Cas test de l'haltère : strong scaling.

– Préparation : représente le temps mis pour effectuer l'assemblage, la renumérotation des inconnues, ainsi que tous les traitements annexes permettant de calculer les informations utiles au solveur direct ;
– Calcul CSC : temps de calcul de la structure de la matrice après renumérotation et redistribution des inconnues ;
– Assemblage : temps de l'assemblage de la matrice renumérotée ;
– Factorisation : temps de factorisation du système linéaire relatif au maillage du niveau grossier ;
– Résolution directe : temps de résolution par méthode directe du système sur le niveau de maillage grossier ;
– 10V : temps de calcul de 10 itérations de V-cycles.

De toutes ces informations, seule la colonne 10V concerne le niveau fin. Toutes les autres informations concernent le prétraitement de la méthode, et plus particulièrement les étapes de préparation nécessaires à la résolution directe. Le temps de 10 itérations est mesuré de la façon suivante : le *timer* est démarré une fois la solution du maillage de niveau grossier interpolée et il est stoppé une fois le dernier post-lissage de la dernière itération atteint : outre le temps des lissages sur le niveau fin, il contient aussi le temps de 10 résolutions directes

sur le niveau grossier ainsi que le temps passé sur les niveaux intermédiaires (restrictions, prolongations et lissages).

La figure 6.14 est la représentation graphique des données contenues dans le tableau 6.4. Elle permet de mettre en évidence plusieurs choses :
– une bonne scalabilité des cycles en V ;
– une mauvaise scalabilité de la méthode directe sur le niveau grossier.
Le comportement de la méthode directe sur le niveau grossier peut s'expliquer par la combinaison de plusieurs facteurs : un nombre d'inconnues par cœur trop faible et une quantité de communications trop importante. Le tableau 6.5 représente le nombre d'inconnues par cœur pour la matrice à 1.2 millions de colonnes qui est clairement un problème trop petit au delà de 32 cœurs de calcul. Dans la dernière ligne de ce tableau, on peut voir que l'un des 512 cœurs ne calcule que 130 colonnes. Le déséquilibre de charge et le faible nombre de colonnes font que les performances de la méthode directe ne sont pas optimales. Bien que la factorisation soit exécutée une seule fois par résolution multigrille, ce point sera à améliorer.

TABLE 6.5: Inconnues sur le niveau grossier après distribution (1.2 millions d'inconnues).

# cœurs	min.	max.	moy.
16	49232	96414	80551
32	1112	58438	40275
64	651	30627	20137
128	300	17390	10068
256	268	8323	5034
512	130	4001	2517

D'un autre côté, l'augmentation du nombre de cœurs augmente aussi le nombre de cellules partagées (partiellement locales). Le tableau 6.6 montre bien l'évolution de la répartition des éléments avec le nombre de cœurs : plus le nombre de cœurs est important, moins il y a d'éléments purement locaux. Certains cœurs ont même aucun élément local, ce qui implique qu'au moins une communication est nécessaire par élément partiellement local. De plus, le nombre minimum d'éléments calculés localement (sélectionnés par la méthode de la répartition aléatoire décrite dans le chapitre 5.4.2) décroît rapidement avec l'augmentation du nombre de cœurs. Dans les cas extrêmes, il se peut que certains cœurs ne disposent d'aucun élément à calculer, pénalisant lourdement la scalabilité de la méthode. Il faudra donc améliorer notre technique d'équilibrage de charge.

Pour éviter ce problème de cœurs n'ayant pas suffisamment de travail à réaliser, on a choisi de se baser sur le maillage raffiné deux fois afin de pouvoir augmenter le nombre de cœurs. Ainsi le niveau de maillage grossier contient suffisamment d'éléments et d'inconnues pour pouvoir être utilisé avec plus de 1000 cœurs.

En se basant sur le maillage de l'haltère raffiné une fois comme niveau grossier (5.2 millions d'inconnues), le FMG à 2 niveaux a été exécuté pour un nombre de cœurs allant de 128 à 2048 (en *batch*). Le niveau fin comporte alors 84.6 millions d'inconnues. Les résultats obtenus sont présentés dans le tableau 6.7. La colonne "# col" présente le nombre de colonnes moyen par cœur et son écart type. Les résultats présentent une scalabilité des itérations de V-cycles quasi parfaite. Toutefois, on retrouve le comportement de la méthode directe qui ne passe pas à l'échelle pour un nombre de cœurs entre 1024 et 2048.

TABLE 6.6: Répartition des éléments sur le niveau grossier après distribution (991,821 éléments).

	purement locaux			calculés localement		
# cœurs	min	max	moy	min	max	moy
16	34798	69057	58622	38145	74217	61988
32	0	40929	28320	800	44989	30994
64	0	20675	13549	439	23405	15497
128	0	11731	6392	254	13321	7748
256	0	4806	2934	194	6425	3874
512	0	2345	1308	84	3112	1937

TABLE 6.7: Haltère 84.6 millions d'inconnues (grossier : 5.2 millions) - FMG(2,2) 2 niveaux - temps en secondes.

# cœurs	Prépa.	Stock.	Assem.	Facto.	Résol.	# col. (σ)		10V
128	229.34	0.36	0.24	103.04	0.26	41063	(11259)	354.23
256	286.71	0.10	0.12	95.40	0.20	20531	(5556)	174.91
512	369.69	0.05	0.06	98.56	0.19	10265	(2534)	85.89
1024	480.22	0.02	0.03	98.96	0.26	5132	(1212)	65.88
2048	1331.51	0.01	0.02	107.95	0.27	2566	(533)	47.29

En suivant la même logique que pour les tests précédents, on a désormais utilisé le maillage de l'haltère raffiné deux fois comme niveau de maillage grossier. Dans ces conditions, le solveur direct doit résoudre un système comportant 21.1 millions d'inconnues alors que le FMG calcule une solution sur 338.5 millions d'inconnues. Les résultats obtenus sont présentés dans le tableau 6.8. Comme dans les cas précédents, la scalabilité de la partie FMG est très bonne. Par contre, on retrouve encore le problème de scalabilité de la méthode directe bien que le nombre moyen de colonnes par cœur soit suffisant.

Afin d'atténuer le problème de scalabilité de la méthode directe, une amélioration peut être apportée : bien que le solveur direct utilisé dans notre travail (PaStiX) puisse utiliser deux niveaux de parallélisme (MPI et threads), seule la version exclusivement MPI a été utilisée ici à cause de la non portabilité (pour l'instant) sur TERA100. En plus d'une faible consommation mémoire, cette version hybride du solveur PaStiX devrait conduire à une meilleure scalabilité en temps.

Un dernier test sur l'haltère a été mené pour passer la barre du milliard d'inconnues sur le niveau fin. Comme dans le cas précédent, le maillage de l'haltère raffinée deux fois a été

TABLE 6.8: Haltère 338.5 millions d'inconnues (grossier : 21.1 millions) - FMG(2,2) 2 niveaux - temps en secondes.

# cœurs	Prépa.	Stock.	Assem.	Facto.	Résol.	# col. (σ)		10V
256	662.22	0.43	0.52	117.96	0.64	82520	(18029)	628.74
512	881.20	0.20	0.23	125.56	0.56	41260	(8642)	345.00
1024	1186.71	0.13	0.14	151.56	0.56	20630	(4473)	213.49

TABLE 6.9: Haltère 1.3 milliards d'inconnues (grossier : 21.1 millions) - FMG 3 niv.

# cœurs	Prépa.	Stock.	Assem.	Facto.	Résol.	# col. (σ)		20V
1024	1160.28	0.12	0.14	147.73	0.55	20630	(4473)	1715.38

utilisé comme maillage grossier. Le FMG à trois niveaux résout alors un système à 1.3 milliards d'inconnues. Pour valider ce calcul, le nombre d'itérations a été poussé à 20 afin d'atteindre le plateau de convergence. Le tableau 6.9 présente les temps obtenus avec une exécution en *batch*.

La courbe de convergence de ce calcul à 1.3 milliards d'inconnues est présentée dans la figure 6.15 et montre une convergence comparable à celles déjà observées.

6.3.3 Synthèse

Les tests de scalabilité présentés ci-dessus montrent un bon comportement lors du passage à l'échelle de la méthode et plus particulièrement une très bonne scalabilité de la partie multigrille. Par contre, le prétraitement et la factorisation du système linéaire du maillage grossier sont coûteux et passent difficilement à l'échelle. Comme dans le logiciel cible les boucles sur les sous-domaines ne modifient pas les matrices de chacun de ces sous-domaines, les phases de préparation et de factorisation peuvent ne s'effectuer qu'une seule fois dans une étape d'initialisation, alors que le multigrille doit être réalisé à chaque passage dans chaque sous-domaine. Le mécanisme est le même pour d'éventuelles boucles sur les angles d'incidence qui ne modifient pas le premier membre. La parfaite scalabilité du multigrille est donc capitale pour une bonne scalabilité du code. Toutefois, pour obtenir de bonnes performances de la méthode dans sa globalité (prétraitement et multigrille), le nombre de cœurs utilisés est contraint par le solveur direct : le nombre d'inconnues par cœurs sur le niveau de maillage grossier doit être suffisant pour que le ratio calcul/communication du solveur direct soit favorable.

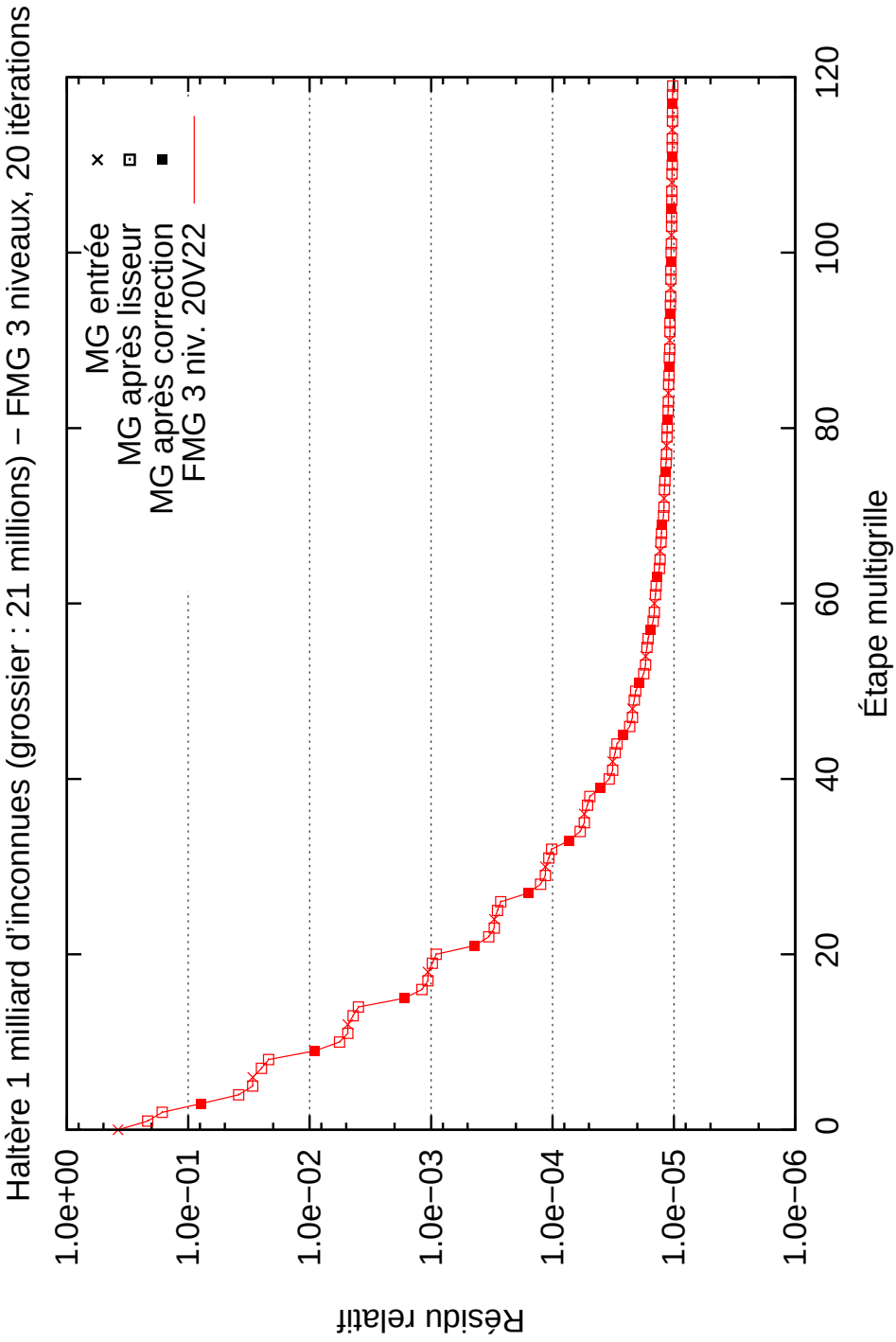


FIGURE 6.15: Convergence de la méthode FMG pour le cas test de l'haltère. Système du niveau fin à 1.3 milliards d'inconnues (grossier : 21.1 millions).

Chapitre 7

Bilan et perspectives

7.1 Bilan

Les travaux de cette thèse nous ont permis de développer un solveur multigrille géométrique parallèle performant. Dans la section 6.2.5, nous avons montré le bon fonctionnement de la méthode pour les problèmes à faibles indices de matériau et à basses fréquences. Dans ces conditions, la convergence du solveur multigrille et de la méthode itérative globale permettent le calcul de SER de qualité satisfaisante. À maillages équivalents en termes d'inconnues, l'utilisation des solveurs multigrille et direct conduit à des SER comparables avec des coûts calculatoire et mémoire clairement à l'avantage de la méthode multigrille : le nombre d'inconnues (et de non-zéros) du système linéaire du niveau grossier est bien inférieur à celui du système linéaire associé au niveau fin et le lisseur itératif *matrix-free* est extrêmement économe en mémoire et de complexité mémoire linéaire (seuls des vecteurs sont stockés). La méthode multigrille peut être utilisée en tant que solveur et permet de réduire le temps CPU et la quantité de mémoire nécessaire à la simulation tout en conservant une qualité de SER satisfaisante.

Pour les problèmes à nombre d'onde plus élevé et sous-maillés sur le niveau grossier, la convergence de la méthode multigrille est dégradée. En comparant les SER calculées par l'emploi de méthodes directe et multigrille, nous avons vu l'amélioration apportée par le raffinement automatique de la méthode multigrille : la SER calculée par le solveur multigrille sur un niveau de maillage fin (basé sur un niveau grossier sous-maillé) s'approche de la solution *réelle* calculée par un solveur direct sur un maillage correctement maillé (cf. section 6.2.6). Le mécanisme de raffinement proposé permet de résoudre le problème lié à la génération de maillages. Toutefois, la méthode de raffinement au centre de gravité des éléments finis employée ici limite le nombre de niveaux de raffinements à deux ou trois. Au delà de cette limite, la forme très écrasée des éléments finis dégrade la convergence de la méthode multigrille et le calcul de la SER.

Pour les cas plus complexes à nombre d'onde important et à matériaux sans perte, la méthode multigrille diverge après une courte phase de convergence (figure 6.5). Cette légère convergence est utilisée afin de préconditionner une méthode de Krylov (GMRES). En comparaison d'un GMRES non préconditionné, la vitesse et le niveau de convergence du GMRES préconditionné par la méthode multigrille sont meilleurs. Pour les cas complexes et à nombre

d'onde important, il est possible de remplacer la méthode directe de l'itération globale par un GMRES préconditionné par la méthode multigrille. Tout comme le solveur multigrille, le GMRES préconditionné consomme peu de mémoire car il est lui aussi *matrix-free*.

Outre la convergence de la méthode et la qualité de la SER calculée, le solveur développé passe à l'échelle et peut être exécuté sur des milliers de cœurs de calcul. Pour les problèmes dont le niveau grossier est de taille raisonnable, la scalabilité du solveur multigrille est quasi-optimale. Toutefois, le bon comportement du solveur multigrille dans sa globalité est contraint par le niveau grossier : les cœurs de calcul utilisés lors de la factorisation du système du maillage grossier doivent avoir suffisamment de calculs à effectuer. Dans le cas contraire, les performances de la factorisation directe sont dégradées et nuisent à la scalabilité de la méthode. Malgré tout, la factorisation ne constitue qu'un prétraitement à la méthode multigrille et la scalabilité de la partie purement multigrille du solveur reste quant à elle quasi-optimale.

Grâce aux performances parallèles et à la faible consommation mémoire de la méthode multigrille, nous avons résolu un problème de Maxwell 3D comptant 1.3 milliards d'inconnues, le record actuel pour une résolution directe étant de 83 millions d'inconnues. Avec le solveur multigrille présenté ici, nous pouvons désormais envisager la résolution de problèmes mettant en jeu des objets beaucoup plus volumineux que ceux actuellement traités par le CEA. À terme, la décomposition de domaine, introduite pour réduire la consommation mémoire de la méthode directe, pourrait être supprimée au profit de la méthode multigrille, permettant alors de s'affranchir de l'itération globale de type Gauss-Seidel.

7.2 Perspectives

Malgré les bonnes performances de la méthode multigrille dans sa globalité, nous avons mis à jour plusieurs points à améliorer :
- au delà d'une certaine limite, les performances du prétraitement se dégradent avec l'augmentation du nombre de cœurs de calcul ;
- le rapport d'aspect des éléments finis influe sur la convergence de la méthode multigrille. De ce fait, la méthode de raffinement actuelle n'autorise qu'un nombre limité de raffinements ;
- pour les matériaux sans perte, la dégradation de la convergence liée au nombre d'onde est très rapide.

Nous présentons ici les améliorations à apporter au solveur multigrille afin de résoudre ou améliorer chacun de ces points. Ces travaux peuvent faire l'objet de travaux futurs pour permettre une utilisation du solveur multigrille pour des calculs de production au CEA.

Équilibrage de charge dynamique

Le problème de la scalabilité du solveur direct mise en évidence lors des tests (tableaux 6.4, 6.7 et 6.8) provient de la faible quantité de calculs à réaliser sur chaque cœur. La solution envisagée consiste alors à utiliser un nombre de cœurs différents pour le prétraitement direct et les étapes des itérations multigrilles.

Un équilibrage de charge dynamique peut se faire au prix d'une redistribution des inconnues à chaque niveau de raffinement. Bien que coûteux en calculs et en communications, cet équilibrage de charge dynamique permettrait de distribuer les inconnues de façon optimale pour chaque méthode (directe ou Jacobi) et ainsi améliorer la scalabilité de la méthode.

Parallélisme à grain fin

Le solveur présenté dans cette thèse n'utilise qu'un seul niveau de parallélisme reposant sur l'utilisation de la bibliothèque de passage de message MPI. Du fait de la consommation mémoire de la méthode directe, les résultats que l'on a présenté ont été réalisés en n'utilisant que deux cœurs par serveur de calcul qui comptent au total 32 cœurs pour avoir la totalité de la mémoire. En tout, 30 cœurs par serveur de calcul sont inactifs pendant la résolution. Il est possible d'utiliser les cœurs inactifs pour accélérer les calculs et réduire la mémoire nécessaire sur le maillage de niveau grossier en utilisant la version hybride MPI-*threads* du solveur PaStiX (et du partitionneur parallèle PT-Scotch).

La principale difficulté de cette amélioration réside dans la cohabitation du solveur direct hybride MPI-threads, du solveur multigrille n'utilisant que MPI et du code de calcul séquentiel.

Modification de la technique de raffinement

Le raffinement actuel a tendance à dégrader le rapport d'aspect des éléments finis en les aplatissant. Pour résoudre ce problème, la méthode de raffinement au centre des arêtes présentée dans le chapitre 5.8.2 doit être employée. Avec cette technique, le rapport d'aspect du maillage initial est conservé au cours de la résolution multigrille, et ce pour chaque niveau de maillage. Toutefois, cette méthode modifie les interfaces entre les sous-domaines de la DDM, rendant le couplage entre le code de calcul et le solveur multigrille plus complexe (maillages non conformes).

Emploi d'opérateurs *translatés* lors du préconditionnement des méthodes de Krylov

La dernière amélioration concerne principalement la gestion des problèmes avec matériaux sans perte. Dans le chapitre 6, nous avons vu que la présence de pertes diélectriques améliore la convergence de la méthode multigrille. En utilisant ce fait, de nombreux articles, dont celui de Erlangga [18], proposent de perturber la diagonale de la matrice par des valeurs à partie imaginaire non nulle. Le système obtenu est alors un opérateur translaté pouvant être utilisé dans le préconditionnement des méthodes de Krylov. Le préconditionneur multigrille est alors utilisé sur le système translaté tandis que la méthode de Krylov utilise l'opérateur original (non translaté). L'introduction de pertes fictives dans l'opérateur permet une meilleure convergence de la méthode multigrille et on peut espérer améliorer la vitesse de convergence du GMRES observée dans la figure 6.6.

Bibliographie

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro. Parallel multigrid smoothing : polynomial versus Gauss-Seidel. *Journal of Computational Physics*, 188(2) :593–610, 2003.

[2] E. Anderson, Z. Bai, and C. Bischof. *LAPACK Users' guide*, volume 9. Society for Industrial Mathematics, 1999.

[3] D.N. Arnold, R.S. Falk, and R. Winther. Multigrid in H(div) and H(curl). *Numerische Mathematik*, 85(2) :197–217, 2000.

[4] AH Baker, RD Falgout, T. Kolev, and UM Yang. Multigrid Smoothers for Ultra-Parallel Computing. Technical Report LLNL-JRNL-435315, Lawrence Livermore National Laboratory, 2010.

[5] L.S. Blackford, A. Cleary, J. Choi, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, et al. *ScaLAPACK users' guide*, volume 4. Society for Industrial Mathematics, 1997.

[6] T. Boonen and I.K. Hameyer. *Multigrid algorithms for electromagnetic field computations*. PhD thesis, Katholieke universiteit leuven, 2008.

[7] A. Brandt. Multi-Level Adaptive Solutions. *Mathematics of computation*, 31(138) :333–390, 1977.

[8] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[9] M. Chanaud, D. Goudin, J.J. Pesqué, and J. Roman. A dedicated method for solving large linear systems arising from finite element discretizations of high-frequency Maxwell problems. In *14th Copper Mountain Conference on Multigrid Methods*, Copper Mountain United States, 03 2009.

[10] M. Chanaud, D. Goudin, J.J. Pesqué, and J. Roman. A parallel hybrid multigrid/direct method for large systems arising from finite element discretization of high frequency Maxwell problems. In *Sparse Days 2009*, Toulouse, 06 2009.

[11] M. Chanaud, D. Goudin, J.J. Pesqué, and J. Roman. Hybrid direct-multigrid solver for large 3D electromagnetic problems. In *11th Copper Mountain Conference on Iterative Methods*, Copper Mountain United States, 04 2010.

[12] M. Chanaud, D. Goudin, J.J. Pesqué, and J. Roman. Parallel solver coupling multigrid and direct methods for Maxwell equations. In *SIAM Conference on Parallel Processing for Scientific Computing (PP10)* , Seattle, Washington United States, 02 2010.

[13] Z. Chen, L. Wang, and W. Zheng. An adaptive multilevel method for time-harmonic Maxwell equations with singularities. *SIAM Journal on Scientific Computing*, 29(1) :118–138, 2008.

[14] B. Després. *Méthodes de décomposition de domaine pour les problèmes de propagation d'ondes en régime harmonique.* PhD thesis, Paris 6, 1991.

[15] B. Després. Méthodes itératives pour l'électromagnétisme sur cray T3D. *Chocs*, 16(53) :53–64, 1997.

[16] J.J. Dongarra, J. Du Croz, S. Hammarling, and I.S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, 16(1) :1–17, 1990.

[17] H.C. Elman, O.G. Ernst, and D.P. O'Leary. A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations. *SIAM Journal on scientific computing*, 23(4) :1291–1315, 2002.

[18] Y Erlangga, C.W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing*, 27(4) :1471–1492, 2006.

[19] O.G. Ernst and M.J. Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. In I. Graham, T. Hou, O. Lakkis, and R. Scheichl, editors, *Numerical Analysis of Multiscale Problems*. Springer Verlag, 2011.

[20] R. Falgout and U. Yang. Hypre : A library of high performance preconditioners. *Computational Science-ICCS 2002*, 2331 :632–641, 2002.

[21] M. Faverge. *Ordonnancement hybride statique-dynamique en algèbre linéaire creuse pour de grands clusters de machines NUMA et multi-cœurs.* PhD thesis, Université Bordeaux I, 09 2009.

[22] L. Fox, H.D. Huskey, and J.H. Wilkinson. Notes on the solution of algebraic linear simultaneous equations. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1) :149, 1948.

[23] V. Frayssé, L. Giraud, S. Gratton, and J. Langou. A set of GMRES routines for real and complex arithmetics on high performance computers. *ACM Trans. Math. Softw*, 31(2) :228–238, 2005.

[24] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins Univ. Press, 1996.

[25] J. Gopalakrishnan, J. Pasciak, and L. Demkowicz. Analysis of a a multigrid algorithm for time harmonic maxwell equations. *SIAM Journal on Numerical Analysis*, 42(1) :90–108, 2004.

[26] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Verlag, 1985.

[27] P. Hénon. *Distribution des données et régulation statique des calculs et des communications pour la résolution de grands systèmes linéaires creux par méthode directe.* PhD thesis, Université Bordeaux I, 11 2001.

[28] P. Hénon, P. Ramet, and J. Roman. PASTIX : a high-performance parallel direct solver for sparse symmetric positive definite systems. *Parallel Computing*, 28(2) :301–321, 2002.

[29] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal cf Research of the National Bureau of Standards Vol*, 49(6) :83, 1952.

[30] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM Journal on Numerical Analysis*, 36(1) :204–225, 1998.

[31] R. Hiptmair and J. Xu. Nodal auxiliary space preconditioning in H(curl) and H(div) spaces. *SIAM Journal on Numerical Analysis*, 45 :2483, 2007.

[32] J. Hu, R. Tuminaro, P. Bochev, J. Garasi, and A. Robinson. Toward an h-independent algebraic multigrid method for Maxwell's equations. *SIAM Journal on scientific computing*, 27(5) :1669–1688, 2006.

[33] T.V. Kolev and P.S. Vassilevski. Auxiliary space AMG for H(curl) problems. *Domain Decomposition Methods in Science and Engineering XVII*, pages 147–154, 2008.

[34] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, 5(3) :308–323, 1979.

[35] D. Lecas. Étude des solveurs iteratifs pour l'électromagnétisme. Rapport interne CEA, 2007.

[36] S. Maclachlan and C.W. Oosterlee. Algebraic multigrid solvers for complex-valued matrices. *SIAM Journal on scientific computing*, 30(3) :1548–1571, 2008.

[37] M. Magolu monga Made, R. Beauwens, and G. Warzée. Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix. *Communications in numerical methods in engineering*, 16(11) :801–817, 2000.

[38] G. Meurant. *Computer solution of large linear systems*. North Holland, 1999.

[39] P. Monk. *Finite element methods for Maxwell's equations*. Oxford University Press, USA, 2003.

[40] J.C. Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3) :315–341, 1980.

[41] J.C. Nédélec. A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 50(1) :57–81, 1986.

[42] D. Osei-Kuffuor and Y. Saad. Preconditioning Helmholtz linear systems. *Applied numerical mathematics*, 60(4) :420–431, 2010.

[43] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4) :617–629, 1975.

[44] F. Pellegrini. *SCOTCH 5.1 user's guide*. Laboratoire Bordelais de Recherche en Informatique (LaBRI), 2008.

[45] J.J. Pesqué. Méthodes itératives sur quelques familles de matrices creuses de la variable complexe. Rapport interne CEA, 8 2002.

[46] J.J. Pesqué. Étude sur la méthode multi grille sur des matrices de très grandes tailles provenant de problèmes de Helmholtz, Maxwell, Euler ou Navier-Stockes. Rapport interne CEA, 2011.

[47] X. Pinel. *A perturbed two-level preconditioner for the solution of three-dimensional heterogeneous Helmholtz problems with applications to geophysics*. PhD thesis, Institut National Polytechnique de Toulouse, 2010.

[48] P. Ramet. *Optimisation de la communication et de la distribution des données pour des solveurs parallèles directs en algèbre linéaire dense et creuse*. PhD thesis, Université Bordeaux I, 01 2000.

[49] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numerical linear algebra with applications*, 9(3) :223–238, 2002.

[50] R. Rew, G. Davis, S. Emmerson, and H. Davies. NetCDF user's guide for C. *Unidata Program Center, June*, 1997.

[51] Y. Saad and M.H. Schultz. GMRES : A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3) :856–869, 1986.

[52] T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm for linear systems with exponential convergence. *Journal of Fourier Analysis and Applications*, 15 :431–436, 2009.

[53] A. Toselli. Overlapping Schwarz methods for Maxwell's equations in three dimensions. *Numerische Mathematik*, 86(4) :733–752, 2000.

[54] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[55] M. Van Gijzen, Y. Erlangga, and C. Vuik. Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted laplacian. *SIAM Journal on scientific computing*, 29(5) :1942–1958, 2007.

[56] R.S. Varga. *Matrix iterative analysis*. Prentice Hall, 1962.

[57] Y. Zhu, A.C. Cangellaris, and J. Wiley. *Multigrid finite element methods for electromagnetic field modeling*. Wiley Online Library, 2006.