THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Electromagnétisme et Systèmes Haute Fréquence

Présentée et soutenue par :

Mme PRISCILLIA DAQUIN

le vendredi 20 octobre 2017

Titre :

Méthodes quasi-optimales pour la résolution des équations intégrales de frontière en électromagnétisme

---

Ecole doctorale :

Génie Electrique, Electronique, Télécommunications (GEET)

Unité de recherche :

Laboratoire Plasma et Conversion d'Energie (LAPLACE)

Directeur(s) de Thèse :

M. JEAN RENE POIRIER

M. RONAN PERRUSSEL

Rapporteurs :

M. FRANCESCO ANDRIULLI, TELECOM BRETAGNE CAMPUS DE BREST

M. MARION DARBAS, UNIVERSITE AMIENS

Membre(s) du jury :

M. OLIVIER CHADEBEC, INP DE GRENOBLE, Président

M. FREDERIC MESSINE, INP TOULOUSE, Membre

Remerciements

Avec cette thèse s'achèvent mes années d'études supérieures, et je suis ravie et fière qu'elles se terminent de cette manière.

Je tiens donc à remercier en tout premier lieu mes deux directeurs de thèse : d'une part Jean-René Poirier, qui a eu confiance en moi dès le début de ma dernière année à l'INP-ENSEEIHT ; d'autre part Ronan Perrussel, qui a su éclairer mes travaux sous une lumière différente. Ce double encadrement, dont j'ai pu avoir un aperçu durant mon projet long en fin d'études d'ingénieur, a été déterminant dans l'atteinte de mes objectifs grâce à vos expertises respectives et vos qualités humaines. Merci à vous deux pour votre disponibilité, votre complémentarité et votre patience.

Je remercie chaleureusement les deux rapporteurs de ce manuscrit pour leur lecture attentive et rigoureuse de ce document : merci à Marion Darbas pour les corrections très détaillées dont vous avez pris le temps de me faire part ; merci à Francesco Andriulli pour les axes d'ouverture évoqués et les interrogations soulevées lors de la soutenance.

Je remercie également Olivier Chadebec pour avoir endossé la présidence du jury et animé la phase de questions avec dynamisme, ainsi que Frédéric Messine qui en tant qu'examinateur a pu nous apporter un regard extérieur sur les problématiques traitées lors de cette thèse.

Je tiens de même à remercier les membres invités du jury : d'une part, Pascal de Resseguier pour sa vision industrielle sur la modélisation en électromagnétisme, et d'autre part, Alfredo Buttari pour son expertise quant au calcul parallèle : un immense merci pour m'avoir aidée au cours de ma thèse lorsque que ces problématiques s'y sont introduites.

Je remercie profondément Julien Vincent, avec qui j'ai pu travailler sur la thématique des Fibres à Cristaux Photoniques et qui a su s'adapter à mon emploi du temps chargé. Ce fût un plaisir de travailler avec toi.

Je remercie tous les acteurs de la recherche au laboratoire LAPLACE pour leur accueil chaleureux et bienveillant, et en particulier les membres du groupe de recherche GRE : Nathalie Raveu, Olivier Pascal, Olivier Pigaglio, Gaëtan Prigent, Junwu Tao, Jacques David, avec une mention spéciale pour Anne-Laure Franc avec qui j'ai pu avoir de longues et agréables discussions sur des sujets divers.

Merci à Jacques Benaioun et à David Bonnafous d'avoir répondu présents à chaque fois que j'ai eu le moindre souci informatique, et désolée de vous avoir sollicités si souvent.

Merci à Jessica Toscano, Valérie Schwarz, Carine Alibert et Catherine Moll Mazella pour leur disponibilité et leur efficacité.

Merci également à toutes les personnes avec qui j'ai pu travailler dans le cadre d'enseignements : Han-Cheng Seat, Olivier Bernal, Danielle Andreu, Emmanuelle Peuch, Hélène Tap. Il était déjà agréable de vous avoir comme enseignants, il l'est plus encore d'enseigner à vos côtés.

Un immense merci à Catherine Montels pour sa gentillesse, son efficacité et sa patience.

Je remercie très chaleureusement tous les doctorants et post-doctorants que j'ai eu la chance de côtoyer durant ces années de thèse : mes collègues de bureau (Caroline, Luciana, Wu, Asma, Jing-Yi, Wencong, Mohammad et Nicolas), mes camarades du GRE (Benedikt, Lucille, Pedro), ainsi que ceux du GREM3 (Julien, Alexandre, Gurvan, Jordan, Youness, Khaled, Alberto) et des autres groupes de recherche, voire d'autres laboratoires (Kamil, Andrea, Joseph, Abdelkader, Andallah, Yann, Andy, Olivier, Mickaël, Laura, Malik, Thomas, Clément, William, et tous ceux que j'oublie). Merci d'avoir fait de notre lieu de travail un endroit si agréable à vivre.

Un immense merci à toutes les personnes qui m'ont permis de conserver un équilibre entre le travail prenant de la thèse et ma vie personnelle. Merci à ma mère, mon frère et ma tante pour leur soutien indéfectible, leur touchante fierté et leur amour inconditionnel.

Je remercie aussi à mes amis qui m'ont soutenue au quotidien, et notamment lorsqu'en début de parcours, ma vie personnelle a été bouleversée. Tenter de tous vous citer serait prendre le risque d'en oublier beaucoup beaucoup trop, mais je tiens à nommer ceux sans qui je n'y serais peut-être pas parvenue : Amouda, Blaise, Caroline. Vous avez été mes piliers, je ne vous en remercierai jamais assez.

Enfin, il y a une dernière personne pour qui ma gratitude n'a pas de limite. Romain, merci pour tout.

Liste des acronymes

$\mathcal{H}$-matrice	Matrice hiérarchique (*hierarchical matrix*).
r_k-matrice	Matrice de rang faible (*low rank matrix*).
ACA	Approximation en Croix Adaptative (*Adaptive Cross Approximation*).
BEM	Méthode des éléments finis de frontière (*Boundary Element Method*).
CFIE	Équation Intégrale en Champs Combinés (*Combined Field Integral Equation*).
CEM	Compatibilité électromagnétique.
DDL	Degré de liberté.
EFIE	Équation Intégrale en Champ Électrique (*Electric Field Integral Equation*).
FMM	Méthode multipôle rapide (*Fast Multipole Method*).
GMRES	Généralisation de la méthode de Minimisation du Résidu (*Generalized Minimal Residual method*).
HCA	Approximation en Croix Hybride (*Hybrid Cross Approximation*).
MoM	Méthode des moments (*Method of Moments*).
PCF	Fibres à cristaux photoniques (*Photonic-Crystal Fibers*).
RAM	Mémoire vive (*Random Access Memory*).
SVD	Décomposition en Valeurs Singulières (*Singular Value Decomposition*).
SER	Surface Équivalente Radar.

Conventions de notations

Dans la suite, nous nous attacherons à respecter les quelques conventions de notations suivantes.

Soient m et n deux entiers naturels, et $\mathbb{K}$ un corps ($\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$). Les vecteurs de $\mathbb{K}^n$ seront représentés en gras italique. Les matrices denses seront représentées en majuscules italiques (exemple : $M \in \mathbb{C}^{m\times n}$). Nous distinguerons les matrices denses des matrices hiérarchiques en ponctuant ces dernières d'un tilde (exemple : $\widetilde{M}$ est la représentation hiérarchique de $M \in \mathbb{C}^{m\times n}$).

Le plan $\mathbb{R}^2$ (resp. l'espace $\mathbb{R}^3$) est supposé muni de la base canonique euclidienne $(\boldsymbol{u}_x, \boldsymbol{u}_y)$ (resp. $(\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_z)$) et de l'origine $\boldsymbol{o}$ de coordonnées toutes nulles. Nous pourrons également utiliser la base canonique polaire $(\boldsymbol{u}_r, \boldsymbol{u}_\theta)$ dans le plan et la base canonique sphérique $(\boldsymbol{u}_r, \boldsymbol{u}_\theta, \boldsymbol{u}_\phi)$ dans l'espace, comme indiqué en figures 1 et 2.

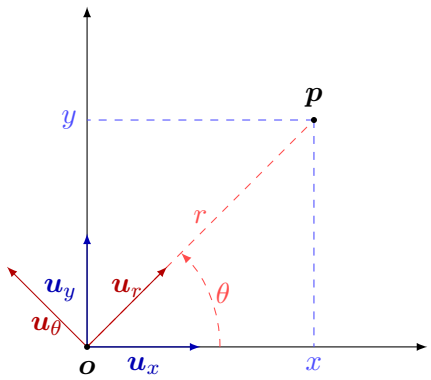


FIGURE 1 – Systèmes de coordonnées du plan.

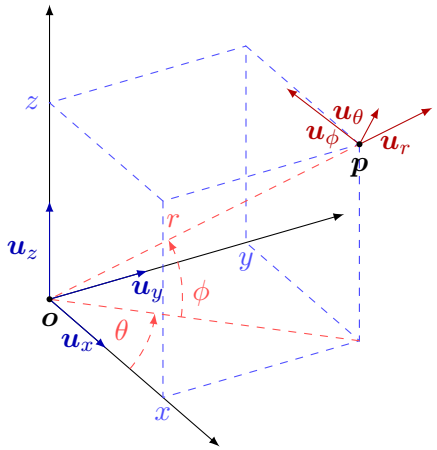


FIGURE 2 – Systèmes de coordonnées de l'espace.

Constantes

ϵ_0	Permittivité électrique du vide.	$(\mu_0 c^2)^{-1}\mathrm{F.m}^{-1}$
i	Unité imaginaire.	$\mathrm{i}^2 = -1$
μ_0	Perméabilité magnétique du vide.	$4\pi \times 10^{-7}\mathrm{N.A}^{-2}$

Grandeurs usuelles

$\boldsymbol{B}$	Induction magnétique.	T ou $\mathrm{N.A^{-1}.m^{-1}}$
$\boldsymbol{D}$	Induction électrique.	$\mathrm{C.m^{-2}}$
$\boldsymbol{E}$	Champ électrique.	$\mathrm{V.m^{-1}}$ ou $\mathrm{N.C^{-1}}$
f	Fréquence de travail.	Hz
$\boldsymbol{H}$	Champ magnétique.	$\mathrm{A.m^{-1}}$
$\boldsymbol{J}$	Courant électrique.	$\mathrm{A.m^{-2}}$
k_0	Nombre d'onde. $k_0 = \frac{2\pi}{\lambda} = \frac{\omega}{c}$.	$\mathrm{m^{-1}}$
λ	Longueur d'onde. $\lambda = \frac{c}{f}$.	m
ω	Pulsation de l'onde. $\omega = 2\pi f$.	$\mathrm{rad.s^{-1}}$

Symboles

$\mathcal{D}$	Opérateur potentiel de double couche.
$\delta^{(\mathrm{a})}$	Erreur absolue.
$\delta^{(\mathrm{r})}$	Erreur relative.
$\mathcal{P}$	Ensemble des parties d'un ensemble. Exemple : $\mathcal{P}(E) = \{A \vert A \subseteq E\}$.
$\mathcal{S}$	Opérateur potentiel de simple couche.

Sommaire

Introduction

Contexte

L'évolution des calculateurs actuels conduit à la modélisation numérique de phénomènes physiques de plus en plus complexes. Parmi les applications les plus communes en électromagnétisme, on peut notamment citer la télédétection radar, la furtivité, ou encore la compatibilité électromagnétique. Ces applications sont régies par les équations de Maxwell, qui décrivent les relations qui existent entre les différentes grandeurs électromagnétiques.

Pour résoudre les équations de Maxwell, on peut écrire une équation intégrale de frontière et la discrétiser par la méthode des éléments finis de frontière (en anglais *Boundary Element Method*, notée BEM), aussi appelée méthode des moments (en anglais *Method of Moments*, notée MoM) [1]. Il s'agit, connaissant le champ incident, de calculer les courants surfaciques électriques et magnétiques, et d'en déduire le champ diffracté en tout point de l'espace par intégration sur la surface de l'objet diffractant.

Cette méthode fait partie des méthodes exactes en domaine fréquentiel, et n'engendre donc pas d'approximations autres que celles issues du modèle de départ, de la discrétisation et de la précision de l'intégration numérique. La BEM a déjà fait ses preuves en électromagnétisme, et est appréciée car seule la surface de l'objet diffractant doit être maillée. Cependant, cette méthode peut se révéler coûteuse car elle conduit à la résolution d'un système linéaire de taille n^2, où n est le nombre de degrés de liberté (DDLs) du problème traité. La résolution directe d'un tel système a alors une complexité en $\mathcal{O}(n^3)$ qu'il n'est pas souhaitable de conserver. Deux stratégies sont alors possibles. La première consiste en l'utilisation d'un solveur itératif reposant sur des produits matrice-vecteur. Si le nombre d'itérations est important, un préconditionnement du système, notamment de type LU incomplet [2], peut être mis en place. La seconde stratégie consiste à compresser le système total par des méthodes adaptées, afin de réduire la complexité de l'inversion. Une factorisation LU de la matrice compressée permettra cette fois une résolution directe du système.

Si certaines méthodes de compression, telles que la méthode multipôle rapide (en anglais *Fast Multipole Method*, notée FMM) [3, 4, 5], ou encore l'Approximation en Croix Adaptative (en anglais *Adaptive Cross Approximation*, notée ACA) [6, 7, 8], ont déjà prouvé leur efficacité, d'autres, telles que l'Approximation en Croix Hybride (en anglais *Hybrid Cross Approximation*, notée HCA) [9, 10, 11], sont encore rarement utilisées et permettent de pallier à certains des inconvénients des deux autres méthodes. En particulier les méthodes ACA et HCA peuvent être utilisées dans le cadre d'un format matriciel particulier, appelé matrice hiérarchique (en anglais *hierarchical matrix*, notée $\mathcal{H}$-matrix). Ce format nécessite toutefois un certain nombre de développements algorithmiques.

Déroulement de la thèse

Les travaux de ce doctorat ont eu pour buts de développer, valider et appliquer à des cas concrets un code de résolution rapide des équations de Maxwell. Ce code s'appuie sur des équations intégrales de frontière et utilise les $\mathcal{H}$-matrices associées à des méthodes de compression afin d'accélérer la résolution de grands systèmes linéaires. Ce travail s'est

déroulé en quatre grandes étapes.

Dans un premier temps, nous avons implémenté le format $\mathcal{H}$-matrice ainsi que l'arithmétique associée. Ce format a alors pu être validé par application de la méthode de compression ACA sur des exemples canoniques. La seconde partie de cette thèse a été consacrée à l'étude et à l'implémentation de différentes méthodes de résolution d'un système linéaire hiérarchique. La troisième étape est focalisée sur l'application de la méthode de compression HCA à la formulation Équation Intégrale en Champ Électrique (en anglais *Electric Field Integral Equation*, notée EFIE). Enfin, nous avons amorcé un travail d'optimisation par parallélisation de certaines opérations.

Les développements effectués utilisent le langage Fortran 90 et s'appuient sur les routines des bibliothèques Blas et Lapack pour adapter l'algèbre linéaire matricielle au cas des $\mathcal{H}$-matrices, ainsi que sur des directives OpenMP pour les développements parallèles.

Plan du manuscrit

Au chapitre 1, nous définissons la formulation EFIE que nous discrétisons par la BEM. Nous précisons également les grandeurs d'intérêt que nous pourrons ensuite calculer, et nous expliquons les fondements de la méthode de compression ACA, qui nous servira de référence dans le chapitre suivant.

Le chapitre 2 est consacré aux définitions et propriétés des matrices de rang faible (en anglais *low rank matrices*, notées r_k-matrices), des $\mathcal{H}$-matrices et des *block cluster trees* représentant la hiérarchie des matrices à compresser. Nous définissons notamment la notion d'*admissibilité*, permettant de déterminer si un bloc matriciel peut ou non être compressé. Nous explicitons alors certaines opérations fondamentales de l'arithmétique des $\mathcal{H}$-matrices et nous vérifions la complexité numérique de celles-ci.

Dans le chapitre 3, nous validons la méthode de compression ACA en terme de précision, et nous déterminons des règles quant au paramétrage de cette méthode. Nous développons également la méthode de compression HCA afin de remédier à certaines des limitations de la méthode ACA. Nous comparons alors les deux méthodes, tant en terme de précision que de temps d'assemblage. Nous étudions enfin plusieurs techniques de résolution d'un système linéaire hiérarchique que nous comparons en termes de temps CPU et de mémoire.

Le chapitre 4 est un chapitre applicatif, dans lequel nous traitons dans un premier temps le cas d'un avion maillé avec un nombre important de triangles. Pour cette première application, nous parallélisons quelques unes des opérations mises en jeu. Nous traitons ensuite des cas de surfaces rugueuses, en nous focalisant en particulier sur les surfaces de Weierstrass. Une étude préliminaire en dimension 2 nous permet de généraliser certains résultats à la dimension 3. Pour finir, nous utilisons le formalisme des $\mathcal{H}$-matrices pour accélérer la recherche de singularités dans des fibres à cristaux photoniques (en anglais *Photonic-Crystal Fibers*, notées PCFs).

Chapitre 1

Résolution rapide des équations de Maxwell par méthodes intégrales

Les équations de Maxwell, postulat de base de l'électromagnétisme, permettent l'étude de la diffraction d'une onde par un obstacle. Bien qu'elles expriment une interdépendance des champs électrique et magnétique, il est possible en régime stationnaire d'étudier indépendamment le comportement de ces deux champs. Cela mène à des équations que l'on peut exprimer sous forme intégrale, puis résoudre par la méthode des éléments finis de frontière (en anglais Boundary Element Method*, notée BEM). Cette méthode, également appelée méthode des moments (ou MoM) dans la communauté hyperfréquences, est toutefois limitante en terme de coûts, car elle génère des systèmes linéaires denses. Il peut donc être nécessaire de réduire ces coûts en rendant ces systèmes parcimonieux, au prix de certaines approximations.*

Dans ce chapitre, nous développerons le cheminement entre les équations de Maxwell et les équations intégrales. Nous détaillerons en particulier la BEM dans le cas de la diffraction d'une onde électromagnétique par un objet parfaitement conducteur. Nous discuterons alors de quelques méthodes de compression matricielle, en vue d'une compression hiérarchique. Nous établirons enfin une liste des possibles origines des erreurs de précision que nous rencontrerons par la suite.

Mots clés

Équations de Maxwell, équation de Helmholtz, BEM, EFIE, compression, erreur

---

1.1 Étude de la diffraction d'une onde électromagnétique

Dans cette section, nous présentons les équations de Maxwell harmoniques, qui permettent de modéliser un problème de diffraction électromagnétique. Ces équations sont ensuite discrétisées, et nous appliquons la méthode des éléments finis de frontière (en anglais *Boundary Element Method*, notée BEM).

1.1.1 Équations de Maxwell harmoniques

Conducteur parfait

Considérons un champ électromagnétique $(\boldsymbol{E}, \boldsymbol{H})$, où $\boldsymbol{E}$ (resp. $\boldsymbol{H}$) désigne le champ électrique (resp. magnétique). Soit un domaine Ω représentant un objet diffractant borné parfaitement conducteur, comme illustré en figure 1.1.1. On note Γ la frontière fermée de l'objet diffractant, $\boldsymbol{n}$ la normale à Γ sortante de Ω, et $\Omega_{\text{ext}} = \mathbb{R}^3 \backslash \overline{\Omega}$ l'extérieur de Ω.

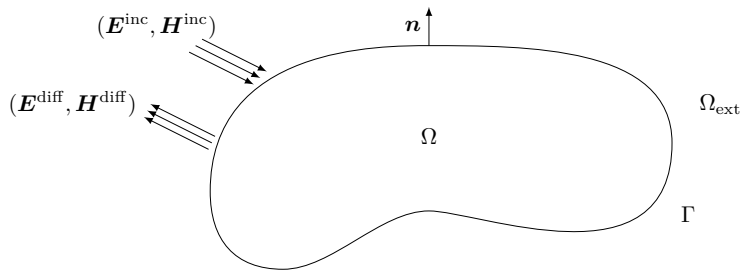


FIGURE 1.1.1 – Diffraction d'une onde électromagnétique incidente $(\boldsymbol{E}^{\text{inc}}, \boldsymbol{H}^{\text{inc}})$ par un objet diffractant parfaitement conducteur.

On se place dans l'espace libre, caractérisé par la permittivité ϵ_0 et la perméabilité μ_0. Connaissant le champ incident $(\boldsymbol{E}^{\text{inc}}, \boldsymbol{H}^{\text{inc}})$, on souhaite calculer le champ diffracté $(\boldsymbol{E}^{\text{diff}}, \boldsymbol{H}^{\text{diff}})$. Le champ électromagnétique total $(\boldsymbol{E}, \boldsymbol{H})$ est alors la somme du champ incident et du champ diffracté, soit

$$(\boldsymbol{E}(\boldsymbol{x}, t), \boldsymbol{H}(\boldsymbol{x}, t)) = (\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}, t), \boldsymbol{H}^{\text{inc}}(\boldsymbol{x}, t)) + (\boldsymbol{E}^{\text{diff}}(\boldsymbol{x}, t), \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}, t)), \tag{1.1.1}$$

où $\boldsymbol{x}$ est un point de $\mathbb{R}^3$ et t désigne le temps. Définissons également pour $\boldsymbol{x} \in \mathbb{R}^3$ les *inductions* électrique et magnétique

$$\boldsymbol{D}(\boldsymbol{x}, t) = \epsilon_0 \boldsymbol{E}(\boldsymbol{x}, t) \qquad \text{Induction électrique,} \tag{1.1.2}$$

$$\boldsymbol{B}(\boldsymbol{x}, t) = \mu_0 \boldsymbol{H}(\boldsymbol{x}, t) \qquad \text{Induction magnétique,} \tag{1.1.3}$$

et la *densité surfacique de courant* électrique

$$\boldsymbol{J}(\boldsymbol{x}, t) = \boldsymbol{n}(\boldsymbol{x}) \wedge \boldsymbol{H}(\boldsymbol{x}, t). \tag{1.1.4}$$

Pour $\boldsymbol{x}_\Gamma \in \Gamma$, la *densité volumique de charge électrique* $\rho(\boldsymbol{x}_\Gamma, t)$ est liée à la densité surfacique de courant électrique par l'équation de *conservation de la charge*

$$\partial_t \rho(\boldsymbol{x}_\Gamma, t) + \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{J}(\boldsymbol{x}_\Gamma, t) = 0. \tag{1.1.5}$$

Pour tout $\boldsymbol{x} \in \mathbb{R}^3$, les *équations de Maxwell* s'écrivent alors

$$\begin{cases} \boldsymbol{\nabla} \cdot \boldsymbol{D}(\boldsymbol{x}, t) = \rho(\boldsymbol{x}, t) & \text{Équation de Maxwell-Gauss,} \\ \boldsymbol{\nabla} \cdot \boldsymbol{B}(\boldsymbol{x}, t) = 0 & \text{Équation du flux magnétique,} \\ \boldsymbol{\nabla} \wedge \boldsymbol{H}(\boldsymbol{x}, t) - \partial_t \boldsymbol{D}(\boldsymbol{x}, t) = \boldsymbol{J}(\boldsymbol{x}, t) & \text{Équation de Maxwell-Ampère,} \\ \boldsymbol{\nabla} \wedge \boldsymbol{E}(\boldsymbol{x}, t) + \partial_t \boldsymbol{B}(\boldsymbol{x}, t) = \boldsymbol{0} & \text{Équation de Maxwell-Faraday.} \end{cases} \tag{1.1.6}$$

Régime harmonique

On distingue le *régime temporel*, dans lequel on étudie l'évolution du champ électromagnétique au cours du temps, du *régime harmonique*. Dans ce modèle, la dépendance en temps du champ est supposée sinusoïdale à une fréquence f. On considère la pulsation ω associée, définie par $\omega = 2\pi f$.

Plaçons nous en un point $\boldsymbol{x} \in \mathbb{R}^3$ quelconque et en un instant t fixé. Le champ $\boldsymbol{U} = \boldsymbol{E}$ ou $\boldsymbol{H}$ s'écrit alors

$$\boldsymbol{U}(\boldsymbol{x}, t) = \boldsymbol{U}_0(\boldsymbol{x}) \cos(\omega t - \varphi(\boldsymbol{x})), \tag{1.1.7}$$

où $\boldsymbol{U}_0$ est le module de $\boldsymbol{U}$ et φ sa phase. Le problème peut alors être projeté dans le plan complexe en écrivant

$$\boldsymbol{U}(\boldsymbol{x}, t) = \mathrm{Re}(\boldsymbol{U}_0(\boldsymbol{x})\mathrm{e}^{-\mathrm{i}\varphi(\boldsymbol{x})}\mathrm{e}^{-\mathrm{i}\omega t}) = \mathrm{Re}(\underline{\boldsymbol{U}}(\boldsymbol{x})\mathrm{e}^{-\mathrm{i}\omega t}). \tag{1.1.8}$$

La quantité $\underline{\boldsymbol{U}}(\boldsymbol{x})$ est appelée le *phaseur* du champ $\boldsymbol{U}(\boldsymbol{x}, t)$. Cette quantité est intéressante car elle exprime totalement le champ $\boldsymbol{U}(\boldsymbol{x}, t)$ à t fixé et est arithmétiquement pratique. En effet, la somme $\boldsymbol{W}(\boldsymbol{x}, t)$ de deux champs $\boldsymbol{U}(\boldsymbol{x}, t)$ et $\boldsymbol{V}(\boldsymbol{x}, t)$ revient à la somme des phaseurs associés, c'est-à-dire

$$\boldsymbol{W}(\boldsymbol{x}, t) = \mathrm{Re}\Big((\underline{\boldsymbol{U}}(\boldsymbol{x}) + \underline{\boldsymbol{V}}(\boldsymbol{x}))\mathrm{e}^{-\mathrm{i}\omega t}\Big). \tag{1.1.9}$$

Par ailleurs la dérivation temporelle $\partial_t \cdot$ du champ $\boldsymbol{U}(\boldsymbol{x}, t)$ revient à une multiplication par $-\mathrm{i}\omega$ du phaseur $\underline{\boldsymbol{U}}(\boldsymbol{x})$ associé, soit

$$\partial_t \boldsymbol{U}(\boldsymbol{x}, t) = \mathrm{Re}(-\mathrm{i}\omega \underline{\boldsymbol{U}}(\boldsymbol{x})\mathrm{e}^{-\mathrm{i}\omega t}). \tag{1.1.10}$$

Les équations en régime harmonique sont donc posées uniquement avec les phaseurs, la dépendance en temps étant alors implicite. Dans la suite, nous appliquons ce formalisme aux champs $\boldsymbol{E}$ et $\boldsymbol{H}$, et nous notons plus simplement $\boldsymbol{U}(\boldsymbol{x})$ le phaseur du champ $\boldsymbol{U}(\boldsymbol{x}, t)$. Nous obtenons alors pour $\boldsymbol{x} \in \mathbb{R}^3$, les *équations de Maxwell harmoniques*

$$\begin{cases} \boldsymbol{\nabla} \cdot \boldsymbol{D}(\boldsymbol{x}) = \rho(\boldsymbol{x}) & \text{Équation de Maxwell-Gauss,} \\ \boldsymbol{\nabla} \cdot \boldsymbol{B}(\boldsymbol{x}) = 0 & \text{Équation du flux magnétique,} \\ \boldsymbol{\nabla} \wedge \boldsymbol{H}(\boldsymbol{x}) + \mathrm{i}\omega \boldsymbol{D}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x}) & \text{Équation de Maxwell-Ampère,} \\ \boldsymbol{\nabla} \wedge \boldsymbol{E}(\boldsymbol{x}) - \mathrm{i}\omega \boldsymbol{B}(\boldsymbol{x}) = \boldsymbol{0} & \text{Équation de Maxwell-Faraday.} \end{cases} \tag{1.1.11}$$

Conditions aux limites

Au système d'équations de Maxwell s'ajoutent des conditions aux limites sur la frontière Γ. En effet, le champ électrique intérieur est nul pour un conducteur parfait. Or le champ électrique est nécessairement continu dans l'espace. Cette condition de continuité se traduit, pour $\boldsymbol{x}_\Gamma \in \Gamma$, par la condition aux limites sur le champ électrique

$$\boldsymbol{E}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n}(\boldsymbol{x}_\Gamma) = \boldsymbol{0}. \tag{1.1.12}$$

Par ailleurs, afin d'assurer l'unicité de la solution, nous appliquons la *condition de radiation de Silver-Müller à l'infini*, définie par l'équation

$$\lim_{|\boldsymbol{x}| \to +\infty} |\boldsymbol{x}| \left(\sqrt{\epsilon_0} \boldsymbol{E}(\boldsymbol{x}) - \sqrt{\mu_0} \boldsymbol{H}(\boldsymbol{x}) \wedge \frac{\boldsymbol{x}}{|\boldsymbol{x}|} \right) = \boldsymbol{0}. \tag{1.1.13}$$

Nous obtenons ainsi, pour le champ total $(\boldsymbol{E}, \boldsymbol{H})$, pour $\boldsymbol{x} \in \mathbb{R}^3$ et $\boldsymbol{x}_\Gamma \in \Gamma$, le problème

$$\left\{\begin{array}{l} \boldsymbol{\nabla} \wedge \boldsymbol{H}(\boldsymbol{x}) + \mathrm{i}\omega\epsilon_0 \boldsymbol{E}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x}), \\ \boldsymbol{\nabla} \wedge \boldsymbol{E}(\boldsymbol{x}) - \mathrm{i}\omega\mu_0 \boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{0}, \\ \boldsymbol{E}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} = \boldsymbol{0}, \\ \lim\limits_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}| \left(\sqrt{\epsilon_0}\boldsymbol{E}(\boldsymbol{x}) - \sqrt{\mu_0}\boldsymbol{H}(\boldsymbol{x}) \wedge \dfrac{\boldsymbol{x}}{|\boldsymbol{x}|} \right) = \boldsymbol{0}. \end{array}\right. \tag{1.1.14}$$

Ce système constitue un problème bien posé [12, 13].

Par la suite, nous nous placerons en *propagation libre*, c'est-à-dire $\boldsymbol{J} = \boldsymbol{0}$ dans Ω_{ext}.

Problèmes en champs séparés

Le problème précédent exprime l'interdépendance des champs $\boldsymbol{E}$ et $\boldsymbol{H}$, déjà perceptible dans les équations de Maxwell-Ampère et de Maxwell-Faraday.

Il est toutefois possible, en propagation libre, de montrer que chacun de ces champs est solution de l'*équation de Helmholtz*, qui s'écrit pour le champ $\boldsymbol{U}=$ $\boldsymbol{E}$ ou $\boldsymbol{H}$ et pour $\boldsymbol{x} \in \mathbb{R}^3$

$$(\Delta + k_0^2)\boldsymbol{U}(\boldsymbol{x}) = 0, \tag{1.1.15}$$

où k_0 désigne le nombre d'onde, défini par

$$k_0 = \frac{2\pi}{\lambda} = \frac{\omega}{c}, \tag{1.1.16}$$

λ étant la longueur d'onde à fréquence de travail f ($\lambda = \frac{c}{f}$), ω la pulsation de l'onde correspondant à la fréquence f ($\omega = 2\pi f$) et c la vitesse de la lumière.

Ceci permet de séparer les champs $\boldsymbol{E}$ et $\boldsymbol{H}$ et de passer ainsi d'un problème couplé à deux problèmes distincts.

Si le champ $\boldsymbol{U}$ est solution de l'équation de Helmholtz, alors il vérifie la *condition de radiation de Sommerfeld*, qui s'écrit

$$\lim_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}| \left(\partial_{|\boldsymbol{x}|}\boldsymbol{U}(\boldsymbol{x}) - \mathrm{i}k_0 \boldsymbol{U}(\boldsymbol{x}) \right) = \boldsymbol{0}. \tag{1.1.17}$$

Cette condition traduit le fait qu'aucune énergie ne peut être rayonnée de l'infini vers la source.

En considérant le cas d'un conducteur parfait, le système d'équations en champ électrique diffracté s'écrit alors, pour $\boldsymbol{x} \in \Omega_{\text{ext}}$ et $\boldsymbol{x}_\Gamma \in \Gamma$,

$$\left\{\begin{array}{l} \Delta \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}) + k_0^2 \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}) = \boldsymbol{0}, \\ \boldsymbol{\nabla} \cdot \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}) = 0, \\ \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} = -\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n}, \\ \lim\limits_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}| \left(\partial_{|\boldsymbol{x}|}\boldsymbol{E}^{\text{diff}}(\boldsymbol{x}) - \mathrm{i}k_0 \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}) \right) = \boldsymbol{0}. \end{array}\right. \tag{1.1.18}$$

Sur le même modèle, nous pouvons établir le problème en champ magnétique diffracté

$$\left\{\begin{array}{l} \Delta \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}) + k_0^2 \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}) = \boldsymbol{0}, \\ \boldsymbol{\nabla} \cdot \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}) = 0, \\ \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} = -\boldsymbol{H}^{\text{inc}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} - \boldsymbol{J}(\boldsymbol{x}_\Gamma), \\ \lim\limits_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}| \left(\partial_{|\boldsymbol{x}|}\boldsymbol{H}^{\text{diff}}(\boldsymbol{x}) - \mathrm{i}k_0 \boldsymbol{H}^{\text{diff}}(\boldsymbol{x}) \right) = \boldsymbol{0}. \end{array}\right. \tag{1.1.19}$$

Réduction du problème en dimension 2

Considérons un champ incident $\boldsymbol{E}^{\text{inc}}$ invariant par translation selon l'axe $\boldsymbol{u}_z$. Soit $\boldsymbol{x} = (\boldsymbol{x}\cdot\boldsymbol{u}_x, \boldsymbol{x}\cdot\boldsymbol{u}_y, \boldsymbol{x}\cdot\boldsymbol{u}_z) \in \mathbb{R}^3$. Supposons que la géométrie Ω présente une invariance par translation selon l'axe $\boldsymbol{u}_z$.

Le champ $\boldsymbol{E}^{\text{inc}}$ est alors lui aussi invariant selon cet axe et on peut écrire, par abus de notation,

$$\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}\cdot\boldsymbol{u}_x, \boldsymbol{x}\cdot\boldsymbol{u}_y, \boldsymbol{x}\cdot\boldsymbol{u}_z) = \boldsymbol{E}^{\text{inc}}(\boldsymbol{x}\cdot\boldsymbol{u}_x, \boldsymbol{x}\cdot\boldsymbol{u}_y), \tag{1.1.20}$$

réduisant un problème de l'espace $\mathbb{R}^3$ en un problème du plan orthogonal à $\boldsymbol{u}_z$. Ceci implique en particulier que la normale $\boldsymbol{n}$ à Γ est contenue dans le plan orthogonal à $\boldsymbol{u}_z$, soit

$$\boldsymbol{n} = n_x\boldsymbol{u}_x + n_y\boldsymbol{u}_y. \tag{1.1.21}$$

Cette simplification nous permet de formuler le problème en dimension 2, ce qui facilite sa résolution. Il est d'ailleurs usuel de traiter un exemple présentant ce type d'invariance en préliminaire à l'étude de cas plus complexes. Ce problème réduit peut se décliner selon les deux modes de propagation *Transverse Magnétique* (ou mode TM) et *Transverse Électrique* (ou mode TE).

Mode TM. Ce mode de propagation est tel que le champ électrique est polarisé perpendiculairement au plan de propagation, soit

$$\boldsymbol{E} = E_z\boldsymbol{u}_z. \tag{1.1.22}$$

Dans ce cas, la condition aux limites devient une condition de *Dirichlet*

$$\boldsymbol{E}\wedge\boldsymbol{n} = E_z\boldsymbol{u}_z\wedge\boldsymbol{n} = -E_z n_y\boldsymbol{u}_x + E_z n_x\boldsymbol{u}_y = \boldsymbol{0}, \tag{1.1.23}$$

ce qui implique que la composante scalaire E_z est nulle.

Le problème est alors simplifié et s'écrit en fonction de la composante selon $\boldsymbol{u}_z$ du champ diffracté, notée E_z^{diff}, pour $\boldsymbol{x}\in\Omega_{\text{ext}}$ et $\boldsymbol{x}_\Gamma\in\Gamma$,

$$\begin{cases} \Delta E_z^{\text{diff}}(\boldsymbol{x}) + k_0^2 E_z^{\text{diff}}(\boldsymbol{x}) = 0, \\ E_z^{\text{diff}}(\boldsymbol{x}_\Gamma) = -E_z^{\text{inc}}(\boldsymbol{x}_\Gamma), \\ \lim\limits_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}|^{\frac{1}{2}}\left(\partial_{|\boldsymbol{x}|}E_z^{\text{diff}}(\boldsymbol{x}) - \mathrm{i}k_0 E_z^{\text{diff}}(\boldsymbol{x})\right) = 0. \end{cases} \tag{1.1.24}$$

Mode TE. Dans ce second mode, c'est le champ magnétique qui est orthogonal à la direction de propagation, soit

$$\boldsymbol{H} = H_z\boldsymbol{u}_z. \tag{1.1.25}$$

Or, l'équation de Maxwell-Ampère harmonique (*cf.* système (1.1.11)) donne

$$\boldsymbol{\nabla}\wedge\boldsymbol{H} + \mathrm{i}\omega\epsilon_0\boldsymbol{E} = \boldsymbol{0} \implies \partial_x H_z\boldsymbol{u}_y - \partial_y H_z\boldsymbol{u}_x = \mathrm{i}\omega\epsilon_0\boldsymbol{E}, \tag{1.1.26}$$

Donc le champ électrique est contenu dans le plan de propagation et s'écrit sous la forme

$$\boldsymbol{E} = E_x\boldsymbol{u}_x + E_y\boldsymbol{u}_y, \tag{1.1.27}$$

avec

$$E_x = \frac{\mathrm{i}}{\omega\epsilon_0}\partial_y H_z \text{ et } E_y = -\frac{\mathrm{i}}{\omega\epsilon_0}\partial_x H_z. \tag{1.1.28}$$

Ce cas de figure implique une condition aux limites de *Neumann*, traduisant le fait que le conducteur est parfait. On a alors

$$\boldsymbol{E} \wedge \boldsymbol{n} = (E_x \boldsymbol{u}_x + E_y \boldsymbol{u}_y) \wedge \boldsymbol{n} = (E_x n_y - E_y n_x)\boldsymbol{u}_x = \boldsymbol{0}. \tag{1.1.29}$$

Ainsi

$$n_y \partial_y H_z + n_x \partial_x H_z = 0, \tag{1.1.30}$$

soit

$$\partial_{\boldsymbol{n}} H_z = 0. \tag{1.1.31}$$

Le problème s'écrit donc, pour $\boldsymbol{x} \in \Omega_{\text{ext}}$ et $\boldsymbol{x}_\Gamma \in \Gamma$,

$$\begin{cases} \Delta H_z^{\text{diff}}(\boldsymbol{x}) + k_0^2 H_z^{\text{diff}}(\boldsymbol{x}) = 0, \\ \partial_{\boldsymbol{n}} H_z^{\text{diff}}(\boldsymbol{x}_\Gamma) = -\partial_{\boldsymbol{n}} H_z^{\text{inc}}(\boldsymbol{x}_\Gamma), \\ \lim\limits_{|\boldsymbol{x}|\to+\infty} |\boldsymbol{x}|^{\frac{1}{2}} \left(\partial_{|\boldsymbol{x}|} H_z^{\text{diff}}(\boldsymbol{x}) - \mathrm{i}k_0 H_z^{\text{diff}}(\boldsymbol{x})\right) = 0. \end{cases} \tag{1.1.32}$$

Dans la suite nous n'étudierons pas ce mode en détail.

1.1.2 Équation Intégrale en Champ Électrique (EFIE)

Comme nous l'avons évoqué dans l'introduction, l'étude de la diffraction d'une onde électromagnétique peut être menée en utilisant différents types de méthodes numériques. Cependant, les applications visées (objets électriquement grands, nécessitant une discrétisation fine, parfaitement conducteurs) nous poussent à nous porter sur le choix d'une méthode exacte intégrale, et en particulier sur la BEM (ou Méthode des moments).

Afin de pouvoir appliquer cette méthode numérique, la première nécessité est d'exprimer le problème à résoudre sous forme d'équations intégrales. C'est ce que nous allons faire dans cette partie.

Formulation intégrale des équations de Maxwell

Notre étude sera principalement consacrée au cas d'une formulation EFIE en dimension 3. Nous traiterons cependant le cas $2D$ en préliminaires.

Noyau de Green. Considérons l'équation de Helmholtz prenant la distribution de Dirac δ_0 pour terme de source, soit pour un champ solution $\boldsymbol{U}$,

$$(\Delta + k_0^2)\boldsymbol{U} = -\delta_0. \tag{1.1.33}$$

La solution de cette équation satisfaisant la condition de rayonnement de Sommerfeld à l'infini est appelée *noyau de Green*, noté G. L'expression du noyau de Green dépend du cas considéré. En particulier, pour $\boldsymbol{x} \in \Omega_{\text{ext}}$ et $\boldsymbol{x}_\Gamma \in \Gamma$,

$$G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) = \frac{1}{4\mathrm{i}} H_0^{(1)}(k_0 |\boldsymbol{x} - \boldsymbol{x}_\Gamma|) \qquad \text{en dimension 2,} \tag{1.1.34}$$

$$G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) = \frac{1}{4\pi} \frac{\mathrm{e}^{\mathrm{i}k_0|\boldsymbol{x}-\boldsymbol{x}_\Gamma|}}{k_0 |\boldsymbol{x} - \boldsymbol{x}_\Gamma|} \qquad \text{en dimension 3.} \tag{1.1.35}$$

$H_0^{(1)}$ désigne la *fonction de Hankel* de première espèce.

Décomposition des champs en potentiels. Considérons l'équation du flux magnétique (*cf.* système (1.1.11)).

Les propriétés des opérateurs différentiels nous assurent alors l'existence d'un vecteur $\boldsymbol{A}$ tel que

$$\boldsymbol{B} = \boldsymbol{\nabla} \wedge \boldsymbol{A}. \tag{1.1.36}$$

On déduit alors de l'équation de Maxwell-Faraday harmonique

$$\boldsymbol{\nabla} \wedge \boldsymbol{E} - \mathrm{i}\omega \boldsymbol{\nabla} \wedge \boldsymbol{A} = \boldsymbol{0}, \tag{1.1.37}$$

ce qui implique

$$\boldsymbol{\nabla} \wedge (\boldsymbol{E} - \mathrm{i}\omega \boldsymbol{A}) = \boldsymbol{0}. \tag{1.1.38}$$

Il existe donc un *potentiel scalaire*, noté Φ, qui vérifie

$$\boldsymbol{E} = \nabla\Phi + \mathrm{i}\omega \boldsymbol{A}. \tag{1.1.39}$$

Le vecteur $\boldsymbol{A}$ est également un potentiel, appelé *potentiel vecteur*.

Les potentiels scalaire et vecteur sont alors liés par la *jauge de Lorenz*,

$$\boldsymbol{\nabla} \cdot \boldsymbol{A} = -\mathrm{i}\omega\epsilon_0\mu_0\Phi. \tag{1.1.40}$$

Représentation intégrale du potentiel scalaire. L'équation de Maxwell-Gauss (*cf.* système (1.1.11)) et la relation d'existence du potentiel vecteur (1.1.36) donnent

$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0} = \boldsymbol{\nabla} \cdot (\boldsymbol{\nabla}\Phi + \mathrm{i}\omega \boldsymbol{A}), \tag{1.1.41}$$

ce qui implique, par utilisation de la jauge de Lorenz (1.1.40),

$$(\Delta + k_0^2)\Phi = \frac{\rho}{\epsilon_0}. \tag{1.1.42}$$

Le potentiel scalaire Φ est donc solution de l'équation de Helmholtz (1.1.15), et sa *représentation intégrale* peut s'écrire, pour $\boldsymbol{x} \in \Omega_{\text{ext}}$, sous la forme

$$\Phi(\boldsymbol{x}) = \int_\Gamma \frac{\rho(\boldsymbol{x}_\Gamma)}{\epsilon_0} G(\boldsymbol{x}, \boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.43}$$

Or l'*équation de conservation de la charge* (1.1.5) s'écrit, pour $\boldsymbol{x}_\Gamma \in \Gamma$ et en régime harmonique,

$$\rho(\boldsymbol{x}_\Gamma) = \frac{1}{\mathrm{i}\omega} \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{J}(\boldsymbol{x}_\Gamma). \tag{1.1.44}$$

Le potentiel scalaire Φ peut donc également s'écrire sous la forme

$$\Phi(\boldsymbol{x}) = \frac{1}{\mathrm{i}\omega} \int_\Gamma G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{J}(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.45}$$

Représentation intégrale du potentiel vecteur. L'équation de Maxwell-Ampère (*cf.* système (1.1.11)) et la relation d'existence du potentiel vecteur (1.1.36) donnent

$$\boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} \wedge \boldsymbol{A}) + \mathrm{i}\omega\mu_0\epsilon_0 \boldsymbol{E} = \mu_0 \boldsymbol{J}. \tag{1.1.46}$$

En appliquant la relation d'existence du potentiel scalaire (1.1.39), on en déduit la relation

$$\boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} \wedge \boldsymbol{A}) + \mathrm{i}\omega\mu_0\epsilon_0 (\boldsymbol{\nabla}\Phi + \mathrm{i}\omega \boldsymbol{A}) = \mu_0 \boldsymbol{J}. \tag{1.1.47}$$

Or, les formules d'analyse vectorielle donnent la relation

$$\boldsymbol{\nabla} \wedge (\boldsymbol{\nabla} \wedge \boldsymbol{A}) = \boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \boldsymbol{A}) - \boldsymbol{\nabla}^2 \boldsymbol{A}. \tag{1.1.48}$$

De plus, la jauge de Lorenz (1.1.40) nous permet de nous affranchir du terme en Φ, soit

$$\boldsymbol{\nabla}\Phi = -\frac{1}{\mathrm{i}\omega\mu_0\epsilon_0}\boldsymbol{\nabla}(\boldsymbol{\nabla} \cdot \boldsymbol{A}). \tag{1.1.49}$$

On en déduit finalement

$$(\Delta + k_0^2)\boldsymbol{A} = -\mu_0 \boldsymbol{J}. \tag{1.1.50}$$

Le potentiel vecteur est donc lui aussi solution de l'équation de Helmholtz (1.1.15), et sa *représentation intégrale* peut s'écrire, pour $\boldsymbol{x} \in \Omega_{\text{ext}}$, sous la forme

$$\boldsymbol{A}(\boldsymbol{x}) = -\mu_0 \int_\Gamma G(\boldsymbol{x}, \boldsymbol{x}_\Gamma)\boldsymbol{J}(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.51}$$

Équation intégrale en champ électrique (EFIE). Soit pour $\boldsymbol{x}_\Gamma \in \Gamma$ le champ électrique diffracté donné par

$$\boldsymbol{E}^{\text{diff}}(\boldsymbol{x}_\Gamma) = \boldsymbol{\nabla}_\Gamma \Phi(\boldsymbol{x}_\Gamma) + \mathrm{i}\omega \boldsymbol{A}(\boldsymbol{x}_\Gamma). \tag{1.1.52}$$

Alors d'après les écritures intégrales du potentiel scalaire (1.1.45) et du potentiel vecteur (1.1.51), l'écriture intégrale du champ électrique est donnée par

$$\boldsymbol{E}^{\text{diff}}(\boldsymbol{x}_\Gamma) = \boldsymbol{\nabla}_\Gamma \frac{1}{\mathrm{i}\omega} \int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{J}(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma - \mathrm{i}\omega\mu_0 \int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\boldsymbol{J}(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma. \tag{1.1.53}$$

Ceci peut s'écrire sous la forme de l'équation

$$\boldsymbol{E}^{\text{diff}}(\boldsymbol{x}_\Gamma) = \mathcal{L}(\boldsymbol{J}(\boldsymbol{x}_\Gamma)), \tag{1.1.54}$$

où $\mathcal{L}$ est l'opérateur linéaire traduisant la forme intégrale des potentiels vecteur et scalaire.

Reprenons à présent la condition aux limites sur le champ électrique, qui s'écrit

$$\boldsymbol{E}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} = \boldsymbol{0}, \text{ soit } \boldsymbol{E}^{\text{diff}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n} = -\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n}. \tag{1.1.55}$$

On en déduit alors l'équation intégrale pour $\boldsymbol{x}_\Gamma \in \Gamma$,

$$\left(\boldsymbol{\nabla}_\Gamma \frac{1}{\mathrm{i}\omega} \int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{J}(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma - \mathrm{i}\omega\mu_0 \int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\boldsymbol{J}(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma\right) \wedge \boldsymbol{n} = -\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma) \wedge \boldsymbol{n}. \tag{1.1.56}$$

L'équation (1.1.56) définit la *formulation EFIE*.

Remarque 1.1.1 (Formulation MFIE)**.** *Il est également possible d'établir une formulation intégrale du champ magnétique $\boldsymbol{H}$. On obtient alors pour $\boldsymbol{x}_\Gamma \in \Gamma$, l'équation intégrale*

$$\frac{1}{2}\boldsymbol{J}(\boldsymbol{x}_\Gamma) + \boldsymbol{n}_{\boldsymbol{x}_\Gamma} \wedge \int_\Gamma \boldsymbol{\nabla}_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\boldsymbol{J}(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma = \boldsymbol{n}_{\boldsymbol{x}_\Gamma} \wedge \boldsymbol{H}^{\text{inc}}(\boldsymbol{x}_\Gamma), \tag{1.1.57}$$

qui exprime la formulation Équation Intégrale en Champ Magnétique (en anglais Magnetic Field Integral Equation*, notée MFIE).*

Remarque 1.1.2 (Formulation CFIE)**.** *Les formulations EFIE et MFIE sont mal posées pour un nombre fini de fréquences f (fréquences de résonance) [14]. On peut alors définir la formulation Équation Intégrale en Champs Combinés (en anglais* Combined Field Integral Equation*, notée CFIE) : il s'agit d'une combinaison linéaire des deux formulations précédentes, écrite sous la forme*

$$CFIE = \alpha EFIE + (1-\alpha)\frac{\mathrm{i}}{k_0} MFIE, \tag{1.1.58}$$

où $0 < \alpha < 1$*. Cette formulation combinée n'admet pas de courant parasite, et ce quelle que soit la valeur du nombre d'onde* k_0*.*

Réduction de la formulation intégrale en dimension 2

Représentation intégrale. La formulation EFIE des problèmes de diffraction en dimension 2 (problèmes (1.1.24) pour le mode TM et (1.1.32) pour le mode TE) nécessite l'énoncé du théorème suivant.

Théorème 1.1.1 (Représentation [15, 16])**.** *Soit* $\boldsymbol{x} \in \mathbb{R}^2 \backslash \Gamma$*. Considérons le champ diffracté* u^{diff}*. Alors* u^{diff} *admet la représentation*

$$u^{\mathrm{diff}}(\boldsymbol{x}) = \int_\Gamma \partial_{\boldsymbol{n}_{\boldsymbol{x}_\Gamma}} G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) u(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma - \int_\Gamma G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) \partial_{\boldsymbol{n}_{\boldsymbol{x}_\Gamma}} u(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma \tag{1.1.59}$$

On définit alors les opérateurs *potentiel de simple couche* et *potentiel de double couche.*

Définition 1.1.1 (Potentiels de simple et de double couche [15, 16])**.** *Soit* φ *et* ψ *deux potentiels et soit* $\boldsymbol{x}$ *un point de* $\mathbb{R}^2 \backslash \Gamma$*. On appelle alors* potentiel de simple couche*, noté* $\mathcal{S}$*, l'opérateur linéaire qui à* φ *associe*

$$\mathcal{S}(\varphi)(\boldsymbol{x}) = \int_\Gamma G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) \varphi(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.60}$$

De même, on appelle potentiel de double couche*, noté* $\mathcal{D}$*, l'opérateur linéaire qui à* ψ *associe*

$$\mathcal{D}(\psi)(\boldsymbol{x}) = \int_\Gamma \partial_{\boldsymbol{n}_{\boldsymbol{x}}} G(\boldsymbol{x}, \boldsymbol{x}_\Gamma) \psi(\boldsymbol{x}_\Gamma)\, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.61}$$

Le théorème 1.1.1 s'écrit alors plus simplement, en posant $\varphi = -\partial_{\boldsymbol{n}_{\boldsymbol{x}_\Gamma}} u$ et $\psi = u$,

$$u^{\mathrm{diff}}(\boldsymbol{x}) = \mathcal{S}(\varphi)(\boldsymbol{x}) + \mathcal{D}(\psi)(\boldsymbol{x}). \tag{1.1.62}$$

Conditions aux limites sur les potentiels. Les potentiels précédemment définis présentent des propriétés particulières au voisinage de la frontière Γ qu'il est important de prendre en compte.

Les potentiels de simple et de double couche sont définis pour des points $\boldsymbol{x}$ de $\mathbb{R}^2 \backslash \Gamma$. L'étude de leur comportement sur la frontière Γ implique un passage à la limite.

Propriété 1.1.1 (Condition aux limites sur le potentiel de simple couche [15, 16])**.** *Soit un potentiel* φ*, soit* $\boldsymbol{x} \in \Omega_{ext}$ *et* $\boldsymbol{x}_\Gamma \in \Gamma$*. Notons* $\mathcal{S}_\Gamma$ *l'opérateur défini sur l'ensemble des potentiels ayant* Γ *pour ensemble de définition et qui à* φ *associe*

$$\mathcal{S}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma) = \int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma) \varphi(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma. \tag{1.1.63}$$

Alors le potentiel de simple couche est continu sur la frontière Γ *:*

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_\Gamma} \mathcal{S}(\varphi)(\boldsymbol{x}) = \mathcal{S}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma). \tag{1.1.64}$$

Propriété 1.1.2 (Condition aux limites sur le potentiel de double couche)**.** *Soit un potentiel* ψ*, soit* $\boldsymbol{x} \in \Omega_{ext}$ *et* $\boldsymbol{x}_\Gamma \in \Gamma$*. Notons* $\mathcal{D}_\Gamma$ *l'opérateur défini sur l'ensemble des potentiels ayant* Γ *pour ensemble de définition et qui à* ψ *associe*

$$\mathcal{D}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma) = \int_\Gamma \partial_{\boldsymbol{n}_{\boldsymbol{y}_\Gamma}} G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\varphi(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma. \tag{1.1.65}$$

Alors le potentiel de double couche est discontinu sur la frontière Γ *:*

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_\Gamma} \mathcal{D}(\psi)(\boldsymbol{x}) = \frac{1}{2}\psi(\boldsymbol{x}_\Gamma) + \mathcal{D}_\Gamma(\psi)(\boldsymbol{x}_\Gamma). \tag{1.1.66}$$

Ces propriétés nous permettent enfin d'établir l'équation intégrale en champ électrique en dimension 2 (mode TM).

EFIE réduite en dimension 2. À partir du théorème de représentation 1.1.1, en posant $\psi = u$ et $\varphi = -\partial_{\boldsymbol{n}_{\boldsymbol{x}_\Gamma}} u$, on a pour $\boldsymbol{x} \in \mathbb{R}^2\backslash\Gamma$,

$$u^{\mathrm{diff}}(\boldsymbol{x}) = \mathcal{S}(\varphi)(\boldsymbol{x}) + \mathcal{D}(\psi)(\boldsymbol{x}). \tag{1.1.67}$$

Ainsi, par passage à la limite, d'après les propriétés 1.1.1 et 1.1.2, on obtient pour $\boldsymbol{x}_\Gamma \in \Gamma$

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_\Gamma} u^{\mathrm{diff}}(\boldsymbol{x}) = u^{\mathrm{diff}}(\boldsymbol{x}_\Gamma) = \mathcal{S}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma) + \frac{1}{2}\psi(\boldsymbol{x}_\Gamma) + \mathcal{D}_\Gamma(\psi)(\boldsymbol{x}_\Gamma). \tag{1.1.68}$$

De même, en appliquant à (1.1.67) une dérivation et en passant à la limite suivant les propriétés 1.1.1 et 1.1.2, on obtient pour $\boldsymbol{x}_\Gamma \in \Gamma$

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_\Gamma} \partial_{\boldsymbol{n}_{\boldsymbol{x}}} u^{\mathrm{diff}}(\boldsymbol{x}) = \partial_{\boldsymbol{n}_{\boldsymbol{x}}} u^{\mathrm{diff}}(\boldsymbol{x}_\Gamma) = -\frac{1}{2}\varphi(\boldsymbol{x}_\Gamma) + \mathcal{D}'^{*}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma) + \mathcal{N}_\Gamma(\psi)(\boldsymbol{x}_\Gamma). \tag{1.1.69}$$

Ainsi en mode TM, on pose $E_z = u = \psi$. On rappelle la condition aux limites du problème (1.1.24), qui s'écrit en fonction de u et pour $\boldsymbol{x}_\Gamma \in \Gamma$,

$$u^{\mathrm{diff}}(\boldsymbol{x}_\Gamma) = -u^{\mathrm{inc}}(\boldsymbol{x}_\Gamma), \text{ soit } u(\boldsymbol{x}_\Gamma) = 0. \tag{1.1.70}$$

Ainsi, $\psi(\boldsymbol{x}_\Gamma) = u(\boldsymbol{x}_\Gamma) = 0$ donc $\mathcal{D}_\Gamma(\psi)(\boldsymbol{x}_\Gamma) = 0$ (par linéarité de l'opérateur $\mathcal{D}_\Gamma$). On déduit de (1.1.68) l'équation intégrale

$$\mathcal{S}_\Gamma(\varphi)(\boldsymbol{x}_\Gamma) = -u^{\mathrm{inc}}(\boldsymbol{x}_\Gamma), \tag{1.1.71}$$

soit

$$\int_\Gamma G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma)\varphi(\boldsymbol{y}_\Gamma)\, \mathrm{d}\boldsymbol{y}_\Gamma = -E^{\mathrm{inc}}(\boldsymbol{x}_\Gamma). \tag{1.1.72}$$

L'équation (1.1.72) constitue la formulation EFIE réduite à la dimension 2 en mode TM.

Remarque 1.1.3. *De même qu'en* (1.1.54)*, on peut écrire* (1.1.72) *sous la forme*

$$\boldsymbol{E}^{\mathrm{diff}}(\boldsymbol{x}_\Gamma) = \mathcal{L}(\varphi(\boldsymbol{x}_\Gamma)), \tag{1.1.73}$$

où $\mathcal{L}$ *est l'opérateur linéaire exprimant l'opposé du potentiel de simple couche.*

1.1.3 Méthode des éléments finis de frontière (BEM)

La BEM permet de limiter l'étude à un maillage surfacique, réduisant par conséquent le nombre d'éléments à considérer dans le calcul. Il s'agit, connaissant le champ incident, de calculer les densités surfaciques de courant électrique sur chaque élément du maillage choisi, et d'en déduire le champ diffracté en tout point de l'espace par intégration sur la surface de l'objet diffractant. Cette méthode conduit à la résolution d'un système linéaire dense, ce qui peut devenir problématique dans le cas d'un grand nombre d'éléments puisque la complexité d'un tel système est *a priori* quadratique. Nous verrons dans la suite comment améliorer cette complexité.

Formulation variationnelle

Considérons le potentiel $\boldsymbol{J}$ et une fonction test $\boldsymbol{J}_t$. En partant de (1.1.54) pour le cas général et de (1.1.73) pour la réduction du problème en dimension 2 en mode TM, on a pour $\boldsymbol{x}_\Gamma \in \Gamma$

$$\mathcal{L}(\boldsymbol{J}(\boldsymbol{x}_\Gamma)) = -\boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma). \tag{1.1.74}$$

Appliquons la *méthode de Galerkine*, qui consiste à multiplier chaque terme par la fonction test $\boldsymbol{J}_t$ appliquée en $\boldsymbol{x}_\Gamma$, puis à intégrer sur Γ. On obtient alors la *formulation variationnelle* de l'EFIE, soit

- dans le cas général,

$$\frac{1}{\mathrm{i}\omega}\int_{\Gamma\times\Gamma} G(\boldsymbol{x}_\Gamma,\boldsymbol{y}_\Gamma)(\boldsymbol{\nabla}_\Gamma\cdot\boldsymbol{J}(\boldsymbol{y}_\Gamma)\boldsymbol{\nabla}_\Gamma\cdot\boldsymbol{J}_t(\boldsymbol{x}_\Gamma) - \mathrm{i}\omega\mu_0\boldsymbol{J}(\boldsymbol{y}_\Gamma)\cdot\boldsymbol{J}_t(\boldsymbol{x}_\Gamma))\ \mathrm{d}\boldsymbol{y}_\Gamma\ \mathrm{d}\boldsymbol{x}_\Gamma = -\int_\Gamma \boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma)\cdot\boldsymbol{J}_t(\boldsymbol{x}_\Gamma)\ \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.75}$$

- pour le problème réduit en dimension 2 en mode TM, on considère un potentiel scalaire J, ainsi qu'une fonction test J_t également scalaire, et on a cette fois

$$\int_{\Gamma\times\Gamma} J_t(\boldsymbol{x}_\Gamma)G(\boldsymbol{x}_\Gamma,\boldsymbol{y}_\Gamma)J(\boldsymbol{y}_\Gamma)\ \mathrm{d}\boldsymbol{y}_\Gamma\ \mathrm{d}\boldsymbol{x}_\Gamma = -\int_\Gamma \boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma)J_t(\boldsymbol{x}_\Gamma)\ \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.76}$$

Discrétisation du problème réduit en dimension 2 (mode TM)

La surface Γ est approchée par une surface formée de segments droits tels que l'intersection de deux segments distincts se réduise soit au vide, soit à un sommet commun.

Nous admettrons dans la suite que Γ est suffisamment approchée par l'ensemble $\mathcal{M}_\Gamma$ de segments s tels que nous puissions considérer

$$\Gamma = \bigcup_{s\in\mathcal{M}_\Gamma} s. \tag{1.1.77}$$

Notons n le nombre de segments du maillage $\mathcal{M}_\Gamma$ ainsi choisi. Fixons ensuite un espace vectoriel de n fonctions tests, dont on sélectionne une base $(\varphi_i)_{i\in[\![1,n]\!]}$. Considérons alors la densité surfacique de courant J (*cf.* définition (1.1.4)) tangente à Γ. Pour $\boldsymbol{x}_\Gamma \in \Gamma$, il existe un système de scalaires $(j_i)_{i\in[\![1,n]\!]}$ tel que

$$J(\boldsymbol{x}_\Gamma) = \sum_{i=1}^{n} j_i\varphi_i(\boldsymbol{x}_\Gamma). \tag{1.1.78}$$

Le coefficient scalaire j_i pour $i \in [\![1,n]\!]$ constitue un degré de liberté (DDL) du problème. Il est localisé sur s_i, le $i^{\text{ème}}$ segment du maillage, et représente le courant J à travers ce segment. Soit $\boldsymbol{x}_\Gamma$ un point de Γ. On définit la $i^{\text{ème}}$ fonction de base φ_i ($i \in [\![1,n]\!]$) par

$$\begin{cases} \varphi_i(\boldsymbol{x}_\Gamma) = 1 & \text{si } \boldsymbol{x}_\Gamma \in s_i, \\ \varphi_i(\boldsymbol{x}_\Gamma) = 0 & \text{sinon.} \end{cases} \tag{1.1.79}$$

On a alors, en partant de (1.1.76), et en prenant pour fonction test φ_j pour $j \in [\![1,n]\!]$,

$$\sum_{i=1}^{n} \left(\int_{\Gamma\times\Gamma} \varphi_j(\boldsymbol{x}_\Gamma) G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma) \varphi_i(\boldsymbol{y}_\Gamma) \, \mathrm{d}\boldsymbol{y}_\Gamma \, \mathrm{d}\boldsymbol{x}_\Gamma \right) j_j = -\frac{\mathrm{i}}{\omega\mu_0} \int_\Gamma E^{\text{inc}}(\boldsymbol{x}_\Gamma) \varphi_j(\boldsymbol{x}_\Gamma) \, \mathrm{d}\boldsymbol{x}_\Gamma \tag{1.1.80}$$

ce qui peut s'écrire sous la forme du système linéaire

$$Z\boldsymbol{j} = \boldsymbol{e}, \tag{1.1.81}$$

avec $Z = (z_{ij})_{i,j\in[\![1,n]\!]}$, $\boldsymbol{j} = (j_j)_{j\in[\![1,n]\!]}$, $\boldsymbol{e} = (e_j)_{j\in[\![1,n]\!]}$, et pour $i,j \in [\![1,n]\!]$,

$$z_{ij} = \int_{\Gamma\times\Gamma} \varphi_j(\boldsymbol{x}_\Gamma) G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma) \varphi_i(\boldsymbol{y}_\Gamma) \, \mathrm{d}\boldsymbol{y}_\Gamma \, \mathrm{d}\boldsymbol{x}_\Gamma, \tag{1.1.82}$$

$$e_j = -\frac{\mathrm{i}}{\omega\mu_0} \int_\Gamma \boldsymbol{E}^{\text{inc}}(\boldsymbol{x}_\Gamma) \varphi_j(\boldsymbol{x}_\Gamma) \, \mathrm{d}\boldsymbol{x}_\Gamma. \tag{1.1.83}$$

Remarque 1.1.4. *Dans le système linéaire* (1.1.81)*, par analogie à la* loi d'Ohm*, la matrice $Z \in \mathbb{C}^{n\times n}$ est appelée matrice impédance du problème, n désignant le nombre de DDLs du problème. Ce système linéaire peut être vu comme une* loi d'Ohm électromagnétique.

Discrétisation en dimension 3

La surface Γ est cette fois-ci approchée par une surface formée de triangles plans tels que l'intersection de deux triangles distincts se réduise soit au vide, soit à un sommet commun, soit à une arête commune. Le maillage $\mathcal{M}_\Gamma$ est alors constitué de l'ensemble des arêtes a communes à exactement deux triangles.

Nous admettrons dans la suite que Γ est suffisamment approchée par l'ensemble des arêtes a de $\mathcal{M}_\Gamma$ pour que nous puissions considérer

$$\Gamma = \bigcup_{a\in\mathcal{M}_\Gamma} a. \tag{1.1.84}$$

Notons n le nombre de d'arêtes du maillage et fixons un espace vectoriel de n fonctions tests dont on choisit une base $(\boldsymbol{\varphi}_i)_{i\in[\![1,n]\!]}$. La densité surfacique de courant $\boldsymbol{J}$ vérifie ici aussi (1.1.78).

Le DDL j_i pour $i \in [\![1,n]\!]$ correspond au courant traversant la $i^{\text{ème}}$ arête intérieure a_i (c'est-à-dire séparant exactement deux triangles distincts) du maillage. Ce courant est compté positivement dans un sens arbitrairement choisi sur l'arête et négativement dans le sens inverse. On définit alors en fonction de ce choix, un triangle $T_i^{(1)}$ contenant le courant dirigé positivement vers l'arête a_i, ainsi que le triangle adjacent $T_i^{(2)}$. On note $|T_i^{(k)}|$, $k \in [\![1,2]\!]$, l'aire du triangle $T_i^{(k)}$, $\boldsymbol{s}_i^{(k)}$ le sommet de $T_i^{(k)}$ qui n'est pas dans a_i, et pour $\boldsymbol{x}_\Gamma \in T_i^{(k)}$, $\boldsymbol{\rho}_i^{(k)}$ le vecteur liant $\boldsymbol{x}_\Gamma$ au sommet $\boldsymbol{s}_i^{(k)}$ et orienté tel que

$$\begin{cases} \boldsymbol{\varphi}_i(\boldsymbol{x}_\Gamma) = \frac{1}{2|T_i^{(1)}|} \boldsymbol{\rho}_i^{(1)}(\boldsymbol{x}_\Gamma) & \text{si } \boldsymbol{x}_\Gamma \in T_i^{(1)}, \\ \boldsymbol{\varphi}_i(\boldsymbol{x}_\Gamma) = \frac{-1}{2|T_i^{(2)}|} \boldsymbol{\rho}_i^{(2)}(\boldsymbol{x}_\Gamma) & \text{si } \boldsymbol{x}_\Gamma \in T_i^{(2)}, \\ \boldsymbol{\varphi}_i(\boldsymbol{x}_\Gamma) = \boldsymbol{0} & \text{sinon.} \end{cases} \tag{1.1.85}$$

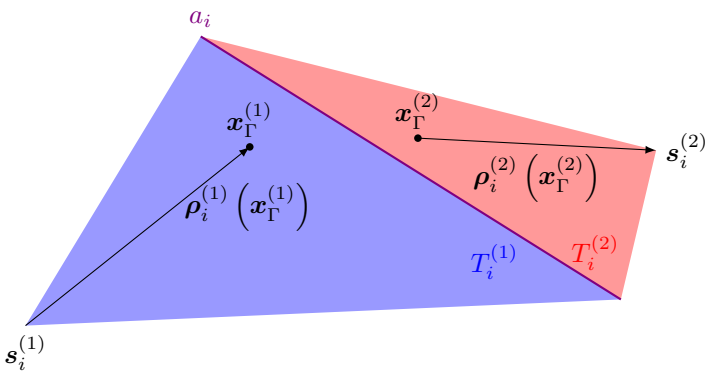


FIGURE 1.1.2 – Schéma d'une arête intérieure a_i et de ses deux triangles adjacents $T_i^{(1)}$ et $T_i^{(2)}$.

On a cette fois, en partant de (1.1.75), et en prenant pour fonction test $\boldsymbol{\varphi}_j$ pour $j \in [\![1, n]\!]$,

$$\sum_{i=1}^{n} \left(\int_{\Gamma\times\Gamma} G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma) \left(\boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) - \frac{1}{k_0^2} \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) \right) \mathrm{d}\boldsymbol{y}_\Gamma \, \mathrm{d}\boldsymbol{x}_\Gamma \right) j_j = \frac{\mathrm{i}}{\omega\mu_0} \int_\Gamma \boldsymbol{E}^{\mathrm{inc}}(\boldsymbol{x}_\Gamma) \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) \, \mathrm{d}\boldsymbol{x}_\Gamma, \quad (1.1.86)$$

ce qui peut également s'écrire sous la forme du système linéaire (1.1.81), avec cette fois-ci

$$z_{ij} = \int_{\Gamma\times\Gamma} G(\boldsymbol{x}_\Gamma, \boldsymbol{y}_\Gamma) \left(\boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) - \frac{1}{k_0^2} \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) \right) \mathrm{d}\boldsymbol{x}_\Gamma \, \mathrm{d}\boldsymbol{y}_\Gamma, \quad (1.1.87)$$

$$e_j = \frac{\mathrm{i}}{\omega\mu_0} \int_\Gamma \boldsymbol{E}^{\mathrm{inc}}(\boldsymbol{x}_\Gamma) \cdot \boldsymbol{\varphi}_j(\boldsymbol{x}_\Gamma) \, \mathrm{d}\boldsymbol{x}_\Gamma. \quad (1.1.88)$$

Remarque 1.1.5. *Que ce soit dans le cas général en dimension 3 ou pour la réduction du problème en dimension 2, les formulations EFIE définies ci-avant ne sont pas valables si $\boldsymbol{x}_\Gamma = \boldsymbol{y}_\Gamma$ $(i = j)$, le noyau de Green n'étant alors pas défini. Il existe alors diverses stratégies pour générer la partie singulière de la matrice, que nous ne détaillerons pas ici.*

Plusieurs méthodes de résolution sont alors disponibles pour le système linéaire (1.1.81) [17, 18, 19, 20]. L'annexe A en fait un rapide inventaire.

1.1.4 Deux cas de référence

Pour la suite, nous proposons de nous appuyer sur deux cas de référence pour tester la formulation EFIE : d'une part un problème réduit en dimension 2 en mode TM, et d'autre part un problème classique en dimension 3. Dans les illustrations à venir, et sauf mention contraire, nous considérerons ces deux cas avec la fréquence $f = 0{,}6\,\mathrm{GHz}$, soit une longueur d'onde $\lambda \approx 0{,}5\,\mathrm{m}$. Nous testerons alors des maillages dont le pas ira de $\lambda/5$ à $\lambda/200$.

Les cas de référence ont été choisis car nous disposons d'une solution analytique associée, ce qui nous permettra de vérifier la précision des méthodes que nous utiliserons ainsi que leur fidélité à la réalité physique.

Cylindre ($2D$)

Le cas du cylindre parfaitement conducteur à base circulaire et de hauteur infinie est intéréssant car son étude se réduit à un cercle en dimension 2. Par ailleurs, la solution

analytique du problème (1.1.72) est connue et met en oeuvre une somme infinie de *fonctions de Bessel*.

On considère la section transversale circulaire d'un cylindre à paroi métallique parfaitement conductrice, discrétisée en n segments de longueur égale, comme représenté en figure 1.1.3. Dans notre cas, nous considérerons un cylindre de rayon $r = 0{,}1\,\mathrm{m}$.

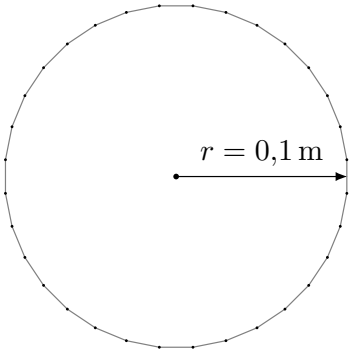


FIGURE 1.1.3 – Discrétisation de la section transversale d'un cylindre infini ($n = 32$).

Sphère ($3D$)

On considère une sphère métallique de rayon $r = 1\,\mathrm{m}$, discrétisée par un maillage surfacique triangulaire, comme représenté en figure 1.1.4. Comme dans le cas du cylindre, nous disposons de la solution analytique de ce cas de référence, qui s'exprime cette fois-ci en fonction des *séries de Mie*.

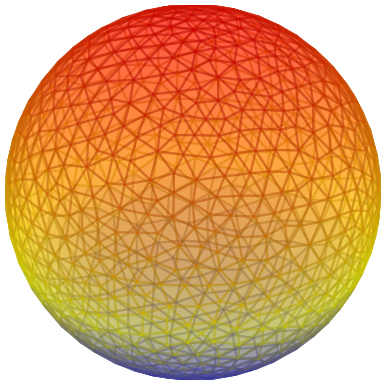


FIGURE 1.1.4 – Discrétisation d'une sphère ($n = 2673$).

1.2 Calcul des grandeurs d'intérêt

Quelle que soit la méthode de résolution choisie, l'équation (1.1.81) permet d'obtenir les courants surfaciques grâce auxquels il devient possible d'accéder à l'expression du champ électromagnétique diffracté, mais aussi à la Surface Équivalente Radar (SER). Dans cette section nous proposons de définir les grandeurs que nous serons amenés à calculer une fois le système (1.1.81) résolu.

1.2.1 Calcul du champ électrique diffracté

Une source rayonne des champs électrique et magnétique dont la configuration spatiale et les amplitudes diffèrent selon la distance entre la source et le point d'observation. En général, on prend en considération deux zones distinctes : la zone en *champ proche* (ou *zone de Fresnel)* et la zone *en champ lointain* (ou *zone de Fraunhofer).*

Définition 1.2.1 (Champ proche, champ lointain)**.** *Soit une source* Ω *de dimension maximale* D*. On suppose par ailleurs que* $D \leqslant \lambda$*. Soit* $\boldsymbol{x}_{\text{obs.}} \in \Omega_{ext}$ *un point d'observation extérieur à la source.*

- *On parle de* calcul en champ proche *(ou* Zone de Fresnel*) si*

$$\min_{\boldsymbol{x}\in\Omega} |\boldsymbol{x} - \boldsymbol{x}_{\text{obs.}}| \leqslant \frac{2D^2}{\lambda}. \tag{1.2.1}$$

- *Inversement, on parle de* calcul en champ lointain *(ou* Zone de Fraunhofer*) si*

$$\min_{\boldsymbol{x}\in\Omega} |\boldsymbol{x} - \boldsymbol{x}_{\text{obs.}}| \geqslant \frac{2D^2}{\lambda}. \tag{1.2.2}$$

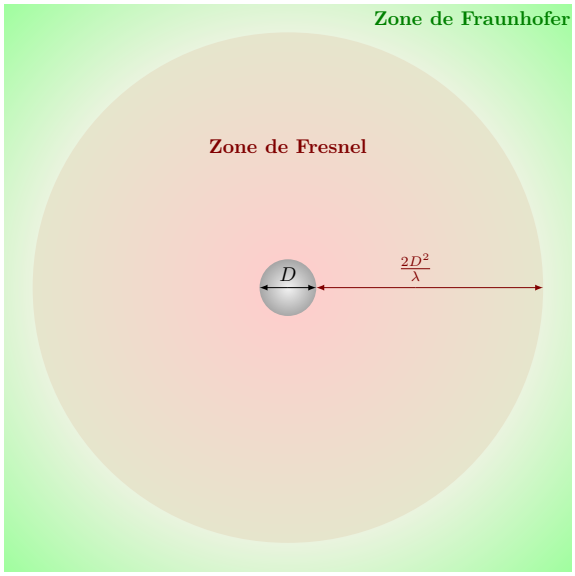


FIGURE 1.2.1 – Zones de calcul en champ proche (Fresnel) et en champ lointain (Fraunhofer) en fonction de la longueur d'onde et des dimensions de l'objet ($D = 1\,\text{m}$, $\lambda = 0{,}5\,\text{m}$).

Pour calculer le champ électrique diffracté en champ proche ou lointain, il suffit d'utiliser la formule de représentation 1.1.1 en distinguant le cas général de la réduction du problème en dimension 2. Ainsi :

- dans le cas général, à partir du courant $\boldsymbol{J}$ décomposé sur la base des vecteurs $(\boldsymbol{\varphi}_i)_{i\in[\![1,n]\!]}$ telle que

$$\forall \boldsymbol{x}_\Gamma \in \Gamma \quad \boldsymbol{J}(\boldsymbol{x}_\Gamma) = \sum_{i=1}^{n} j_i \boldsymbol{\varphi}_i(\boldsymbol{x}_\Gamma), \tag{1.2.3}$$

  on obtient pour $\boldsymbol{x} \in \Omega_{\text{ext}}$:

$$\boldsymbol{E}(\boldsymbol{x}) = \sum_{i=1}^{n} j_i \left(\int_\Gamma G(\boldsymbol{x}, \boldsymbol{y}_\Gamma) \boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \, \mathrm{d}\boldsymbol{y}_\Gamma + \frac{1}{k_0^2} \boldsymbol{\nabla}_{\boldsymbol{x}} \int_\Gamma G(\boldsymbol{x}, \boldsymbol{y}_\Gamma) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{y}_\Gamma) \, \mathrm{d}\boldsymbol{y}_\Gamma \right), \tag{1.2.4}$$

$$\boldsymbol{H}(\boldsymbol{x}) = \sum_{i=1}^{n} j_i \left(\boldsymbol{\nabla} \wedge \int_{\Gamma} G(\boldsymbol{x}, \boldsymbol{y}_{\Gamma}) \boldsymbol{\varphi}_i(\boldsymbol{y}_{\Gamma}) \, \mathrm{d}\boldsymbol{y}_{\Gamma} \right). \tag{1.2.5}$$

- en dimension 2, le courant J scalaire décomposé sur la base $(\varphi_i)_{i \in [\![1,n]\!]}$ s'écrit

$$\forall \boldsymbol{x}_{\Gamma} \in \Gamma \quad J(\boldsymbol{x}_{\Gamma}) = \sum_{i=1}^{n} j_i \varphi_i(\boldsymbol{x}_{\Gamma}), \tag{1.2.6}$$

  et on obtient cette fois pour $\boldsymbol{x} \in \Omega_{\text{ext}}$:

$$\boldsymbol{E}(\boldsymbol{x}) = \sum_{i=1}^{n} j_i \left(\int_{\Gamma} G(\boldsymbol{x}, \boldsymbol{y}_{\Gamma}) \varphi_i(\boldsymbol{y}_{\Gamma}) \, \mathrm{d}\boldsymbol{y}_{\Gamma} \right), \tag{1.2.7}$$

$$\boldsymbol{H}(\boldsymbol{x}) = \sum_{i=1}^{n} j_i \left(\boldsymbol{\nabla} \wedge \int_{\Gamma} G(\boldsymbol{x}, \boldsymbol{y}_{\Gamma}) \varphi_i(\boldsymbol{y}_{\Gamma}) \, \mathrm{d}\boldsymbol{y}_{\Gamma} \right). \tag{1.2.8}$$

1.2.2 Surface Équivalente Radar (SER)

Un radar est un capteur électromagnétique qui permet l'étude de l'interaction entre une onde électromagnétique (le signal radar) et un objet, appelé *cible* dans ce contexte. Cette interaction est évaluée par le calcul de la Surface Équivalente Radar (SER). Le calcul de cette quantité peut ensuite être utilisé dans des contextes variés, tels que la télédétection ou l'analyse de risques, et s'applique à divers types de cibles (végétation, surface de la mer, avions, bâteaux, *etc.*).

La définition de la SER découle de l'*équation radar*, qui décrit le bilan de puissance d'une liaison radar. Elle s'écrit

$$P_{\text{r}} = \frac{P_{\text{e}} G_{\text{e}} G_{\text{r}} \lambda \sigma}{(4\pi)^3 R_{\text{e}}^2 R_{\text{r}}^2}, \tag{1.2.9}$$

avec P_{e} (resp. P_{r}) la puissance émise (resp. reçue), G_{e} (resp. G_{r}) le gain de l'antenne d'émission (resp. de réception), R_{e} (resp. R_{r}) la distance entre la cible et l'antenne d'émission (resp. de réception), λ la longueur d'onde, et enfin σ la SER.

La SER s'exprime en m^2 et désigne le rapport entre l'énergie diffusée dans une direction par unité d'angle solide, et la densité d'énergie incidente. Elle est définie par l'expression

$$\sigma(\theta) = \lim_{R \to +\infty} 4\pi R^2 \frac{|\boldsymbol{E}^{\text{diff}}|^2}{|\boldsymbol{E}^{\text{inc}}|^2}, \tag{1.2.10}$$

où R désigne la distance entre la cible et le radar.

On distingue les radars *monostatiques* des radars dits *bistatiques*. On parle de radar monostatique lorsque l'antenne d'émission et l'antenne de réception sont localisées au même endroit par rapport à la cible, et de radar bistatique dans le cas inverse (*cf.* figure 1.2.2).

Pour un radar bistatique, les directions d'incidence et d'observation diffèrent et forment un plan à ϕ constant dans lequel on se place pour calculer la SER bistatique. On peut notamment fixer le plan de calcul et l'angle d'incidence, puis observer la SER en fonction de l'angle d'observation θ qui pourra varier entre 0 et 360°.

Pour un radar monostatique, l'angle d'incidence et celui d'observation sont les mêmes. Il n'y a cette fois-ci pas de plan ϕ prédéfini par les directions d'incidence et d'observation. Ainsi pour un angle d'observation θ donné, on doit calculer la SER dans toutes les directions ϕ.

La SER dépend des caractéristiques géométriques et diélectriques de l'objet, de la fréquence de travail et de la polarisation de l'onde. La figure 1.2.3 donne le tracé de la SER bistatique pour un cylindre ($2D$) ainsi que pour une sphère ($3D$) à différentes fréquences f de travail.

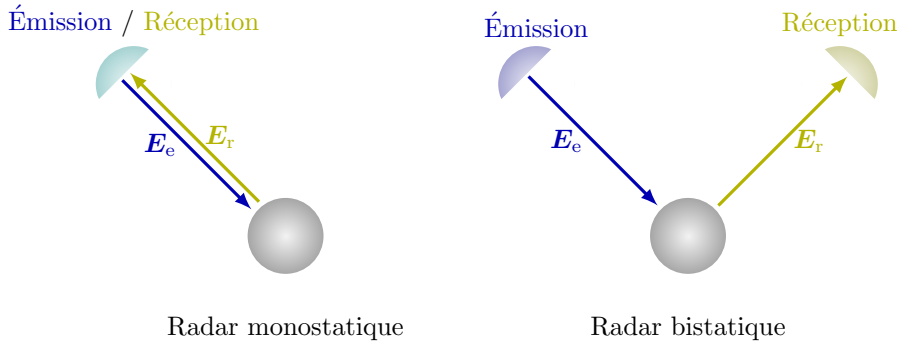


FIGURE 1.2.2 – Configurations possibles d'un radar. $\boldsymbol{E}_e$ (resp. $\boldsymbol{E}_r$) désigne le champ électrique émis (resp. reçu).

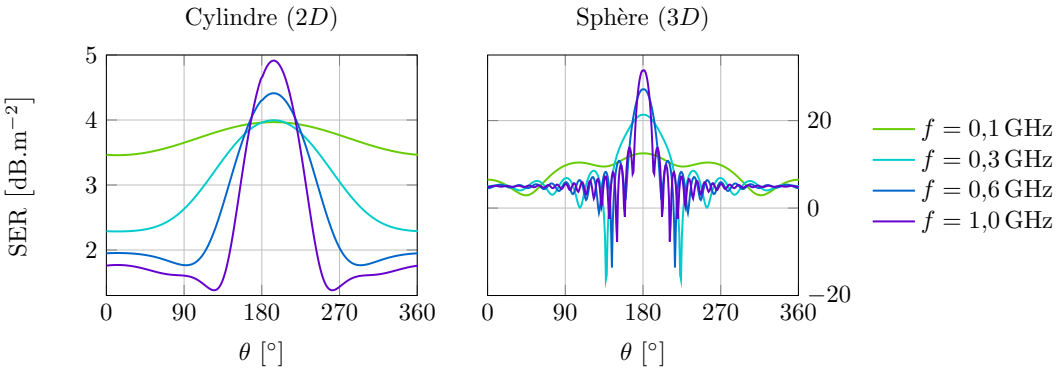


FIGURE 1.2.3 – SER bistatique pour différentes valeurs de la fréquence f (incidence normale, $\phi = 0°$).

Remarque 1.2.1. *L'équation* (1.1.81) *ne fait état que d'un unique second membre* $\boldsymbol{e}$*, vecteur de* $\mathbb{C}^n$*. On peut étendre cette équation au cas de la résolution d'une équation à* p *seconds membres* ($p \in \mathbb{N}$) *telle que*

$$ZI = V, \tag{1.2.11}$$

avec $Z \in \mathbb{C}^{n\times n}$*,* $I \in \mathbb{C}^{n\times p}$ *et* $V\mathbb{C}^{n\times p}$*. Cette extension peut s'avérer utile pour le calcul de la SER monostatique, puisque ce calcul nécessite autant de seconds membres que de plans d'incidence considérés pour le champ électrique incident* $\boldsymbol{E}^{\text{inc}}$*.*

1.3 Méthodes usuelles de compression matricielle

1.3.1 Nécessité des méthodes de compression

Comme nous avons pu l'observer dans la partie 1.1, la discrétisation des opérateurs intégraux mène à la résolution d'un système matriciel. Ce système est dense car l'évaluation du noyau de Green sur deux élements de la surface Γ est *a priori* non nulle. Bien que le fait de ne considérer que la frontière Γ de l'objet permette de réduire le nombre de DDLs à considérer, le système à résoudre n'en est pas moins dense, ce qui peut vite devenir problématique.

En notant n le nombre de DDLs à considérer, le coût mémoire d'un tel système est en $\mathcal{O}(n^2)$, et la complexité algorithmique du produit matrice-vecteur est également en $\mathcal{O}(n^2)$. Cette complexité est encore plus élevée pour l'inversion matricielle, proportionnelle à n^3 (*cf.* annexe B).

Pour limiter ces complexités algorithmiques trop importantes, il est possible d'approcher ces matrices denses par des matrices *parcimonieuses* (ou *data sparse*). Il existe pour cela plusieurs méthodes de compression, dont certaines sont très populaires. On peut notamment citer les techniques d'interpolation [9], le *Panel Clustering* [21, 22] ou encore la compression QR [23, 24]. Toutes ces méthodes se basent sur le même principe qui consiste à négliger les interactions dites *lointaines* pour privilégier les interactions *proches*, et sont des méthodes *a posteriori*.

Ces dernières années, deux méthodes semblent tirer leur épingle du jeu, toutes deux associées à une forme de hiérarchisation des matrices à compresser. Il s'agit d'une part de la FMM [3] qui est une méthode de compression apparue il y a une trentaine d'années, et d'autre part l'ACA [25], développée dans les années 2000. Ces deux méthodes permettent notamment une accéleration de la résolution itérative de grands systèmes linéaires. Nous allons entrer un peu plus dans les détails pour ces deux méthodes de compression *a priori* toujours très populaires.

1.3.2 Méthode Multipôle Rapide Multiniveau (MLFMM)

La FMM utilise une expansion multipolaire du noyau de Green G, solution de l'équation de Helmholtz. Ceci a pour conséquence de regrouper les interactions proches et de les considérer comme une seule et même source contenue dans une boîte de calcul [3].

Dans le cas de la FMM simple, toutes les boîtes élémentaires de calcul sont de la même taille. En revanche, la MLFMM permet d'avoir des boîtes élémentaires de diverses tailles selon l'importance des interactions considérées.

En électromagnétisme on utilise la MLFMM pour accélérer la multiplication matrice-vecteur, et donc les solveurs itératifs dans la BEM [4]. En traitant les interactions entre des fonctions très différentes par la MLFMM, il n'est pas nécessaire de stocker les éléments de matrice correspondants, ce qui réduit considérablement le coût mémoire. Si la MLFMM est appliquée de manière hiérarchique [5], elle améliore la complexité algorithmique des produits matrice-vecteur dans un solveur itératif en la faisant passer de $\mathcal{O}(n^2)$ à $\mathcal{O}(n\log(n))$, avec une tolérance sur la précision $\varepsilon_{\text{MLFMM}} > 0$.

La MLFMM a été l'une des premières méthodes permettant de traiter des problèmes électriquement grands avec des temps de calculs et des coûts mémoire raisonnables en utilisant la BEM. En cela elle constitue un tournant dans le domaine des méthodes numériques. Cependant cette méthode n'est pas applicable à tous les problèmes, et notamment les problèmes périodiques.

1.3.3 Approximation en Croix Adaptative (ACA)

À la MLFMM ont succédé plusieurs autres méthodes de compression matricielle, telles que le *Panel Clustering* [21, 22] ou encore la compression QR [24, 23]. Il s'agit cependant de méthodes de compression *a posteriori*.

La technique ACA pallie à ce problème en permettant la construction d'une approximation *a priori* d'un bloc matriciel. Cette méthode a été introduite par Mario Bebendorf en 2000 [6]. La matrice Z est séparée en blocs de différentes tailles selon un algorithme de *clustering* qui regroupe les DDLs proches. On utilise souvent une structure de $\mathcal{H}$-matrice, dont nous détaillerons la mise en place au chapitre 2. Chaque bloc ainsi obtenu représente les interactions entre deux régions d'un milieu : s'il existe peu d'interactions entre deux milieux, on peut se permettre de perdre de l'information et donc de compresser le bloc correspondant, en l'approchant par une r_k-matrice (*cf.* partie 2.1).

Tout comme dans le cas de la MLFMM, la matrice n'est pas stockée dans sa totalité, mais l'algorithme d'assemblage permet d'en obtenir une approximation par blocs, ce qui réduit également l'espace de stockage. Cette méthode améliore aussi la complexité algorithmique des produits matrice-vecteur, la faisant passer de $\mathcal{O}(n^2)$ à $\mathcal{O}(n\log(n))$ lorsque l'ACA est associée à une structure de $\mathcal{H}$-matrice (*cf.* section 2.3). Cette opération étant nécessaire dans les solveurs itératifs, l'amélioration de la complexité implique une résolution plus rapide par le biais de ces solveurs.

L'ACA peut notamment être utilisée pour des problèmes de compatibilité électromagnétique [7], ou plus généralement pour des problèmes comportant un très grand nombre de DDLs [8]. Elle peut également être associée à la Méthode de Décomposition de Domaine pour des calculs de rayonnement d'antennes ou de diffraction par des objets électriquement grands [26]. La méthode d'estimation de l'erreur est fiable si le noyau est dit *asymptotiquement lisse.*

Définition 1.3.1 (Noyau asymptotiquement lisse.)**.** *Soit g un noyau utilisé dans une discrétisation par éléments finis de frontière. Il est dit* asymptotiquement lisse *si il existe des constantes c_1 et c_2 réelles et un degré de singularité $\sigma \in \mathbb{N}$ tels que, en désignant $\boldsymbol{u} = \boldsymbol{x}$ ou $\boldsymbol{y}$, l'inégalité*

$$\forall n \in \mathbb{N} \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3 \quad |\partial_{\boldsymbol{u}}^n g(\boldsymbol{x}, \boldsymbol{y})| \leqslant n! c_1 (c_2 \|\boldsymbol{x} - \boldsymbol{y}\|)^{-n-\sigma} \tag{1.3.1}$$

soit vérifiée.

Par application de l'algorithme ACA à une matrice $Z \in \mathbb{C}^{m\times n}$, on obtient une approximation de Z sous la forme

$$Z \approx UV^H, \tag{1.3.2}$$

avec $U \in \mathbb{C}^{m\times k}$ et $V \in \mathbb{C}^{n\times k}$. L'algorithme est itératif et ne s'arrête que si un critère d'arrêt dépendant de la précision ε_{ACA} souhaitée est vérifié. Il est en effet possible de calculer une estimation de la norme de Frobenius (*cf.* définition 1.4.2) de l'approximation au fur et à mesure de sa construction.

La première version de l'ACA repose sur la recherche de pivots totaux à chaque itération, ce qui la rend difficilement applicable aux problèmes de diffraction. En effet elle nécessite l'évaluation d'une matrice dite de *résidu*, de même taille que Z, afin de déterminer le pivot total maximal, ce qui augmente considérablement le coût de la méthode et nécessite une connaissance totale de la matrice. C'est pourquoi nous avons préféré implémenter l'ACA avec recherche de pivot partiel, proposée par Rjasanow [16] et dont la description est donnée dans l'algorithme 1.3.1.

Il existe également une version multi-niveaux de l'ACA, appelée *Multilevel* ACA [27], et pouvant également s'appuyer sur un format hiérarchique. Nous ne détaillerons pas cet algorithme ici.

L'ACA est une méthode purement algébrique, agissant directement au niveau de la matrice. Son implémentation est relativement aisée, quelle que soit la formulation utilisée. Il suffit en effet, pour un bloc matriciel $Z \in \mathbb{C}^{m\times n}$ donné, de parvenir à générer, pour tout $i \in [\![1, m]\!]$ (resp. $j \in [\![1, n]\!]$), la $i^{\text{ème}}$ ligne (resp. la $j^{\text{ème}}$ colonne) de Z. Par conséquent l'ACA a l'avantage de ne pas souffrir de problèmes de robustesse de la résolution en basse fréquence, puisqu'elle ne dépend pas de la fonction de Green utilisée.

Cependant l'ACA est sujette à certaines limitations du fait de sa nature heuristique. En effet, la méthode d'estimation de l'erreur n'assure pas la convergence de la méthode vers une approximation suffisamment précise du bloc à compresser. Ce sont ces limitations

Algorithme 1.3.1 *Adaptive Cross Approximation* (ACA) avec recherche de pivot partiel.

procedure ACA(Z,ε_{ACA},U,V)
  $k_{\text{lim}} = \min(m, n)$
  $\mathcal{I} = \emptyset$
  $\mathcal{J} = \emptyset$
  $k = 0$
  $l = 0$
  $c = 0$
  while size($\mathcal{I}$) $< k_{\text{lim}}$ **and** size($\mathcal{J}$) $< k_{\text{lim}}$ **do**
    $V_k = Z_l^T$ ▷ Calcul de la $l^{\text{ème}}$ ligne de Z.
    $\mathcal{I} = \mathcal{I} \cup \{l\}$
    $V_k = V_k - \sum_{m=0}^{k-1} u_{mr} V_m$
    $c = \max_{j \notin \mathcal{J}} |v_{kj}|$
    $\gamma = v_{ki}^{-1}$
    $U_k = Z_c$ ▷ Calcul de la $c^{\text{ème}}$ colonne de Z.
    $\mathcal{J} = \mathcal{J} \cup \{c\}$
    $U_k = U_k - \sum_{m=0}^{k-1} v_{mc} U_m$
    $r = \max_{i \notin \mathcal{I}} |u_{ki}|$
    $V_k = \gamma V_k$
    $\|\Sigma_k\|_{\text{F}}^2 = \|\Sigma_{k-1}\|_{\text{F}}^2 + 2 \sum_{i=0}^{k-1} U_k^T U_i V_i^H V_k + \|U_k\|_2^2 \|V_k\|_2^2$ ▷ Évaluation de la norme de Frobenius.
    if $\|U_k\|_2 \|V_k\|_2 < \varepsilon_{\text{ACA}} \|\Sigma_k\|_{\text{F}}$ **then**
      stop
    else
      $k \leftarrow k + 1$
    end if
  end while
end procedure

qui ont motivé le développement d'une méthode de compression hybride, l'HCA, dont nous détaillerons l'implémentation en partie 3.2.

Dans la suite, tous les exemples numériques réalisés avec compression et apparaissant avant la section 3.2 sont effectués par ACA.

1.4 Évaluation de la précision

Dans la suite de ce travail, nous nous attacherons en diverses occasions à évaluer les méthodes employées. Pour ce faire, nous étudierons la mémoire vive (en anglais *Random Access Memory*, notée RAM) utilisées par chacune d'entre elles, leur temps d'exécution et leur précision, ce dernier critère étant prioritaire sur les deux premiers. Les erreurs de précision que nous rencontrerons peuvent avoir différentes origines et ne s'évaluent pas de la même manière selon les cas. Il convient donc d'établir une liste des erreurs que nous pourrons rencontrer, et de préciser dans chaque cas comment évaluer l'erreur faite sur la précision.

Dans tous les cas, la notion d'erreur est liée à une norme que nous noterons $\|\cdot\|$ et dont la définition dépendra du type d'objets desquels nous évaluerons la précision. En particulier, et sauf mention contraire, nous considérerons la *norme* 2 pour les vecteurs et la *norme de Frobenius* pour les matrices. Ces deux normes sont définies ci-dessous.

Définition 1.4.1 (Norme 2 d'un vecteur)**.** *Soit $\boldsymbol{x} \in \mathbb{C}^n$. La* norme 2 *de $\boldsymbol{x}$ est le réel positif $\|\boldsymbol{x}\|_2$ défini par la relation*

$$\|\boldsymbol{x}\|_2 \overset{\text{déf.}}{=} \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}. \tag{1.4.1}$$

Définition 1.4.2 (Norme de Frobenius d'une matrice)**.** *Soit $A \in \mathbb{C}^{m\times n}$. La* norme de Frobenius *de A est le réel positif $\|A\|_F$ défini par la relation*

$$\|A\|_F \overset{\text{déf.}}{=} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}. \tag{1.4.2}$$

Grâce à ces normes, nous pourrons évaluer l'erreur d'une quantité x donnée par rapport à une quantité de référence $x_{\text{réf.}}$ de deux manières différentes :

- soit en évaluant l'erreur absolue $\delta^{(\text{a})}$ entre x et $x_{\text{réf.}}$, définie par

$$\delta^{(\text{a})}(x, x_{\text{réf.}}) \coloneqq \|x - x_{\text{réf.}}\|, \tag{1.4.3}$$

- soit en calculant l'erreur relative $\delta^{(\text{r})}$ entre x et $x_{\text{réf.}}$, donnée par

$$\delta^{(\text{r})}(x, x_{\text{réf.}}) \coloneqq \frac{\|x - x_{\text{réf.}}\|}{\|x_{\text{réf.}}\|}. \tag{1.4.4}$$

La plupart du temps, nous préférerons évaluer l'erreur relative entre deux données à comparer. Nous considérerons connues et acceptables les deux erreurs suivantes.

Erreur de modélisation. Il s'agit de l'erreur commise entre la réalité physique et la façon dont elle est modélisée. Cette erreur peut apparaître selon la pertinence du modèle et des paramètres choisis. Une fois une solution analytique de référence fixée, le calcul de cette solution, supposée exacte, repose souvent sur le calcul d'une somme infinie (séries de Mie, sommes de fonctions de Bessel, *etc.*). Cette précision dépendra donc également de l'ordre de sommation avec lequel est calculée la solution analytique. Dans la suite, nous considérerons cette erreur négligeable.

Erreur arithmétique. Ce type d'erreur est directement lié à la précision de la machine sur laquelle nous travaillons, c'est-à-dire au nombre de chiffres significatifs disponibles pour exprimer une valeur donnée. Durant toute la phase de développement, nous nous sommes attachés à définir nos variables et constantes en double précision, ce qui, en Fortran 90 sur une machine fonctionnant à 64 bit, correspond à une précision relative de l'ordre de 10^{-15} [28]. Cette valeur sera pour nous une référence, de sorte que dans la suite, toute erreur relative inférieure à $\delta^{(\mathrm{r})}_{\mathrm{mac.}} \coloneqq 10^{-15}$ sera considérée nulle.

Dans la suite, nous pourrons évaluer d'autres erreurs pour lesquelles il reste à définir x et $x_{\mathrm{réf.}}$.

1.4.1 Erreur de discrétisation

Lors d'une simulation en électromagnétisme, il est communément admis qu'une discrétisation à un pas h inférieur au rapport $\lambda/10$ est nécessaire pour s'assurer de la fidélité du résultat obtenu par rapport à la réalité physique. Ce rapport maximal n'est toutefois qu'une limite donnée à titre indicatif, et peut énormément varier selon la géométrie traitée, le milieu étudié, ou plus simplement la tolérance que l'on se fixe sur l'erreur acceptable.

Reprenons le système linéaire (1.1.81) à résoudre. La première étape de la résolution consiste à choisir un pas de discrétisation h, qui impliquera un certain nombre n de DDLs. Un compromis doit donc être fait entre la perspective de calculs précis mais coûteux (en terme de temps et d'espace mémoire), ou d'une simulation grossière mais légère. Une fois ce compromis effectué et le nombre n de DDLs fixés, il est possible d'évaluer l'erreur induite par la discrétisation si l'on dispose d'une grandeur d'intérêt exacte, telle que la SER bistatique ou le champ lointain en certains points (*cf.* section 1.2).

Notons $\boldsymbol{v}_{\mathrm{réf.}}$ un vecteur contenant un nombre n_v de ces valeurs de référence. La résolution du système (1.1.81) sans compression et en utilisant une méthode de résolution qui n'implique pas d'approximation supplémentaire permet d'obtenir le vecteur $\boldsymbol{v}$ contenant les valeurs de la grandeur d'intérêt issue de la discrétisation. L'erreur relative induite par la discrétisation s'écrit alors

$$\delta^{(\mathrm{r})}_{\mathrm{disc.}}(\boldsymbol{v}, \boldsymbol{v}_{\mathrm{réf.}}) \coloneqq \frac{\|\boldsymbol{v} - \boldsymbol{v}_{\mathrm{réf.}}\|_2}{\|\boldsymbol{v}_{\mathrm{réf.}}\|_2}. \tag{1.4.5}$$

Cette erreur se calcule surtout dans les cas pour lesquels nous disposons d'une solution analytique considérée exacte (à condition que la série dont elle résulte soit calculée à un ordre suffisamment important), comme c'est le cas pour la SER bistatique d'une sphère parfaitement conductrice. Elle sera étudiée au chapitre 3 pour nos deux cas de référence.

1.4.2 Erreur sur l'intégration numérique

La BEM implique l'évaluation d'intégrales simples ou doubles que l'on ne peut calculer numériquement avec exactitude. L'intégrale est alors approchée par une somme pondérée

de valeurs correspondant à la valeur de l'intégrale en un certain nombre de points judicieusement choisis, appelés *points de quadrature* [29]. Ainsi, pour une solution régulière, plus l'on considère de points de quadrature sur le domaine d'intégration, plus l'intégration numérique est précise.

Remarque 1.4.1. *Si le maillage choisi est suffisamment fin pour que les éléments soient considérés comme localement confondus avec la surface* Γ*, ajouter des points d'intégration revient à affiner le maillage. En cela, l'erreur commise sur l'intégration peut être vue comme une erreur commise sur la discrétisation.*

La durée d'une intégration est proportionnelle au nombre de points de quadrature choisi dans le cas d'une intégrale simple, et au carré de ce nombre dans le cas d'une intégrale double. De ce fait, un compromis doit être trouvé quant au nombre de points de quadrature à moyenner pour obtenir une intégrale suffisamment précise en un temps raisonnable. Ce compromis peut dépendre de la géométrie traitée et du pas de maillage choisi.

Rappelons le système linéaire (1.1.81) à résoudre,

$$Z\boldsymbol{J} = \boldsymbol{e}. \tag{1.4.6}$$

Choisissons un nombre n_{Gauss} de points de quadrature et assemblons la matrice $Z_{n_{\text{Gauss}}}$. Pour évaluer l'erreur commise sur $Z_{n_{\text{Gauss}}}$ par l'intégration numérique, il convient de résoudre le système linéaire aussi précisément que possible afin d'obtenir un vecteur solution $\boldsymbol{J}_{n_{\text{Gauss}}}$. Si nous disposons d'une solution de référence $\boldsymbol{J}_{\text{réf.}}$ supposée exacte, il suffit alors d'évaluer l'erreur relative

$$\delta^{(\text{r})}_{\text{int.}}(\boldsymbol{J}_{n_{\text{Gauss}}}, \boldsymbol{J}_{\text{réf.}}) \coloneqq \frac{\|\boldsymbol{J}_{n_{\text{Gauss}}} - \boldsymbol{J}_{\text{réf.}}\|_2}{\|\boldsymbol{J}_{\text{réf.}}\|_2}. \tag{1.4.7}$$

La figure 1.4.1 présente le temps d'assemblage de la matrice pleine $Z_{n_{\text{Gauss}}}$, ainsi que l'erreur $\delta^{(\text{r})}_{\text{int.}}$ pour différentes valeurs de n_{Gauss} dans le cas de la sphère.

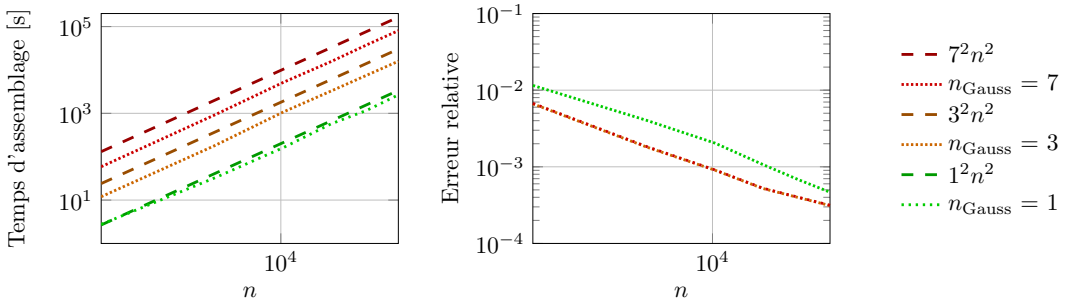


FIGURE 1.4.1 – Comparaison du temps d'assemblage et de l'erreur commise sur la solution en fonction du nombre de DDLs pour différentes valeurs de n_{Gauss}.

Cette figure nous permet de constater que pour $n_{\text{Gauss}} = 3$, la précision atteinte est meilleure que pour $n_{\text{Gauss}} = 1$ mais similaire à la précision obtenue pour $n_{\text{Gauss}} = 7$. En revanche, le temps d'exécution de l'assemblage, bien que proportionnel à n^2 dans tous les cas, ne suit pas tout à fait la dépendance en n^2_{Gauss} attendue en particulier pour $n_{\text{Gauss}} = 1$. On constate toutefois que plus n_{Gauss} est faible, plus le temps d'assemblage est réduit.

Ces observations nous permettent d'établir dès à présent qu'en dimension 3, nous utiliserons 3 points de quadrature de Gauss-Legendre pour le calcul des intégrations. Une opération similaire nous permet de fixer, en dimension 2, $n_{\text{Gauss}} = 2$. Dans la suite, et sauf mention contraire, nous adopterons ces réglages.

Remarque 1.4.2. *Il est également possible de faire varier le nombre de points de Gauss à considérer selon que l'on se situe en champ proche ou en champ lointain. Nous avons toutefois préféré adopter un réglage uniforme du nombre de points de Gauss.*

1.4.3 Erreur de compression

Notons $\widetilde{Z}$ l'approximation par compression d'une matrice $Z \in \mathbb{C}^{m\times n}$ à une précision $\varepsilon_{\text{comp.}}$ près. Cette compression permet de gagner de l'espace de stockage au détriment de la précision. L'erreur relative induite par la compression est directement obtenue par la relation

$$\delta^{(\text{r})}_{\text{comp.}}(\widetilde{Z}, Z) \coloneqq \frac{\|\widetilde{Z} - Z\|_{\text{F}}}{\|Z\|_{\text{F}}}. \tag{1.4.8}$$

Il est également possible d'évaluer approximativement l'erreur due à la compression en définissant un vecteur aléatoire $\boldsymbol{x}$ et en évaluant l'erreur relative

$$\delta^{(\text{r})}_{\text{comp.}}(\widetilde{Z}\boldsymbol{x}, Z\boldsymbol{x}) \coloneqq \frac{\|\widetilde{Z}\boldsymbol{x} - Z\boldsymbol{x}\|_2}{\|Z\boldsymbol{x}\|_2}, \tag{1.4.9}$$

avec les mêmes effets de pondération.

Nous pourrons alors vérifier que ces deux erreurs relatives sont du même ordre que $\varepsilon_{\text{comp.}}$, validant ainsi la précision de la méthode de compression choisie. Nous évaluerons cette erreur en particulier aux sections 3.1 et 3.2.

1.4.4 Erreur de résolution

Pour finir, certaines méthodes de résolution du système (1.1.81) (*cf.* annexe A) peuvent engendrer une erreur de précision supplémentaire. En particulier, on peut distinguer les *méthodes directes* des *méthodes itératives*.

Si les méthodes directes n'impliquent pas, *a priori*, d'approximations supplémentaires (autres que les limitations dues au calcul à virgule flottante), elles sont coûteuses car elles sont synonymes d'une inversion de la matrice Z. Cette inversion peut alors être effectuée de façon approximative, à une précision $\varepsilon_{\text{inv.}}$ donnée. On s'attendra alors à ce que l'erreur relative sur la résolution soit inférieure à cette précision.

Les méthodes itératives quant à elles impliquent une approximation supplémentaire, puisqu'elles s'attachent à approcher la solution exacte avec une tolérance $\varepsilon_{\text{ité.}}$. Plus cette tolérance est fine, plus il faudra d'itérations pour converger vers cette tolérance et plus la méthode sera coûteuse en terme de temps de calcul. Si toutefois la méthode arrive à convergence, on s'attendra également à ce que l'erreur relative sur la résolution soit inférieure à cette tolérance. L'utilisation d'un préconditionneur pourra réduire le nombre d'itérations de la méthode, *a priori* sans influence sur la précision de la solution obtenue.

Quelle que soit le solveur choisi, l'évaluation de cette erreur nécessite de résoudre le système $Z\boldsymbol{J} = \boldsymbol{e}$, qui nous fournira un vecteur solution $\boldsymbol{J}$. Deux méthodes sont alors disponibles.

La première consiste à prédéfinir un vecteur de référence $\boldsymbol{J}_{\text{réf.}}$, puis d'en déduire le vecteur $\boldsymbol{e}_{\text{réf.}}$ défini par $\boldsymbol{e}_{\text{réf.}} \coloneqq Z\boldsymbol{J}_{\text{réf.}}$. On peut alors résoudre le système $Z\boldsymbol{J} = \boldsymbol{e}_{\text{réf.}}$ et calculer l'erreur du vecteur $\boldsymbol{J}$ obtenu par rapport à $\boldsymbol{J}_{\text{réf.}}$, soit

$$\delta^{(\text{r})}_{\text{rés.}}(\boldsymbol{J}, \boldsymbol{J}_{\text{réf.}}) \coloneqq \frac{\|\boldsymbol{J} - \boldsymbol{J}_{\text{réf.}}\|_2}{\|\boldsymbol{J}_{\text{réf.}}\|_2}. \tag{1.4.10}$$

L'autre méthode consiste à prédéfinir le vecteur $\boldsymbol{e}_{\text{réf.}}$ puis à résoudre le système $Z\boldsymbol{J} = \boldsymbol{e}_{\text{réf.}}$ et d'évaluer l'erreur du produit $Z\boldsymbol{J}$ par rapport à $\boldsymbol{e}_{\text{réf.}}$, soit

$$\delta^{(\text{r})}_{\text{rés.}}(Z\boldsymbol{J}, \boldsymbol{e}_{\text{réf.}}) \coloneqq \frac{\|Z\boldsymbol{J} - \boldsymbol{e}_{\text{réf.}}\|_2}{\|\boldsymbol{e}_{\text{réf.}}\|_2}. \tag{1.4.11}$$

En intégrant la matrice Z au calcul de l'erreur, cette seconde méthode se limite en fait au calcul du *résidu* issu de la résolution, et ne permet pas de jauger les éventuels problèmes de conditionnement de la matrice Z, ce qui n'est pas le cas avec la méthode précédente.

Nous évaluerons ce type d'erreurs notamment à la section 3.3.

Conclusion du chapitre

Dans ce chapitre, nous avons présenté, à partir des équations de Maxwell harmoniques, la résolution d'un problème de diffraction par équations intégrales en champ électrique, qui constitue la formulation EFIE. Nous avons en particulier détaillé le cas général en dimension 3 et nous en avons déduit le modèle simplifié en dimension 2 pour le mode TM dans le cas d'une invariance par translation selon une dimension de l'espace. Nous avons alors pu poser le système linéaire qu'il nous faudra résoudre afin d'obtenir la valeur du courant surfacique en tout point du maillage établi. Une fois ce courant calculé, il est alors possible d'évaluer la SER d'un objet diffractant, ainsi que le champ diffracté en champ proche ou en champ lointain.

La résolution d'un système linéaire dense peut vite devenir coûteuse à mesure que le nombre de DDLs augmente. Or nous avons vu qu'un nombre suffisant de DDLs est nécessaire pour obtenir un résultat numérique suffisamment proche de la réalité physique. De ce fait, il paraît indispensable de disposer de méthodes permettant de rendre la résolution de la BEM raisonnable en terme de coûts. L'application d'une méthode de compression semble être une solution appropriée à ce problème, quitte à hiérarchiser la matrice afin d'en déterminer les parties qu'il semble raisonnable de compresser sans perte dommageable de précision sur le résultat global.

Dans le chapitre suivant, nous allons détailler la démarche de hiérarchisation et de compression du système linéaire grâce au formalisme des $\mathcal{H}$-matrices.

Chapitre 2

$\mathcal{H}$-matrices

*Une $\mathcal{H}$-matrice est une représentation hiérarchisée d'une matrice. Ce format permet de considérer la matrice traitée comme un ensemble de blocs matriciels indépendants, que l'on peut éventuellement compresser, s'ils vérifient une condition dite d'*admissibilité*. Les blocs admissibles,* a priori *denses, sont alors remplacés par des blocs de rang faible. Il en résulte une représentation compressée de la matrice totale.*

Dans ce chapitre, nous commencerons par définir les notions de r_k-matrice et d'admissibilité, puis nous expliquerons la façon dont est déterminée la hiérarchie arborescente d'une matrice donnée par clustering *; ces outils nous permettront d'exposer les mécanismes de la construction d'une matrice hiérarchique et de décrire les différentes opérations disponibles dans l'arithmétique des $\mathcal{H}$-matrices. Tous les algorithmes mentionnés ont été implémentés au cours de la thèse en Fortran 90 et s'appuient sur des routines des librairies d'algèbre linéaire* Blas *et* Lapack*.*

Ce chapitre est très largement inspiré des travaux de Bebendorf [30] et Börm [31, 11, 32], qui sont fondamentaux pour la compréhension de la théorie sur laquelle s'appuient les $\mathcal{H}$-matrices. En particulier, la plupart des définitions, propriétés et théorèmes des sections 2.1, 2.2 et 2.3 sont issus de [30].

Mots clés

$\mathcal{H}$-matrice, r_k-matrice, admissibilité, block cluster tree, coarsening, *arithmétique*

2.1 Matrices de rang faible et r_k-matrices

La théorie des $\mathcal{H}$-matrices repose sur la possibilité d'approcher un bloc matriciel, correspondant à des interactions lointaines, par un autre, de rang plus faible. Dans cette partie, nous nous appuierons sur des définitions et théorèmes mathématiques pour comprendre la légitimité d'une telle approximation.

2.1.1 Définitions

Rang d'une matrice

Rappelons dans un premier temps les définitions des notions élémentaires que sont l'*image directe* et le *rang* d'une matrice.

Définition 2.1.1 (Image directe et rang d'une matrice)**.** *Soit $A \in \mathbb{C}^{m\times n}$, $m, n \in \mathbb{N}$. On appelle* image directe de A *et on note* $\mathrm{Im}(A)$ *l'ensemble défini par*

$$\mathrm{Im}(A) \overset{\text{déf.}}{=} \{Ax \in \mathbb{C}^m \,|\, x \in \mathbb{C}^n\}. \tag{2.1.1}$$

Le rang de A*, noté* $\mathrm{rg}(A)$*, est alors défini par*

$$\mathrm{rg}(A) \overset{\text{déf.}}{=} \dim(\mathrm{Im}(A)). \tag{2.1.2}$$

On rappelle aussi les relations suivantes.

Propriété 2.1.1. *Soient $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{m\times n}$, et $C \in \mathbb{C}^{n\times p}$, $m, n, p \in \mathbb{N}$. Alors :*

- $\mathrm{rg}(A) \leqslant \min(m, n)$,
- $\mathrm{rg}(A + B) \leqslant \mathrm{rg}(A) + \mathrm{rg}(B)$,
- $\mathrm{rg}(AC) \leqslant \min(\mathrm{rg}(A), \mathrm{rg}(C))$.

Concrètement, le rang d'une matrice $A \in \mathbb{C}^{m\times n}$ indique le nombre de lignes ou de colonnes linéairement indépendantes de A. Si $\mathrm{rg}(A) < \min(m, n)$ alors la matrice contient de l'information redondante qu'il peut être judicieux d'éliminer. Nous verrons comment procéder dans la suite de cette section.

r_k-matrices

Les matrices dont le rang est inférieur à la plus petite de ses dimensions ont des propriétés intéressantes que nous allons étudier dans cette partie.

Définition 2.1.2. *Pour $k, m, n \in \mathbb{N}$ tels que $k \leqslant \min(m, n)$, on note $\mathbb{C}_k^{m\times n}$ l'ensemble des matrices $A \in \mathbb{C}^{m\times n}$ possédant au plus $k \in \mathbb{N}$ lignes ou colonnes linéairement indépendantes, ou* matrices de rang au plus k*, soit*

$$\mathbb{C}_k^{m\times n} \overset{\text{déf.}}{=} \{A \in \mathbb{C}^{m\times n} \mid \mathrm{rg}(A) \leqslant k\}. \tag{2.1.3}$$

Il serait alors avantageux de pouvoir révéler les redondances de $A \in \mathbb{C}^{m\times n}$. C'est justement la problématique à laquelle répond le théorème suivant, en proposant pour toute matrice $A \in \mathbb{C}^{m\times n}$ de rang au plus k une écriture sous la forme d'un produit de deux matrices chacune de taille inférieure à celle de A.

Théorème 2.1.1. *Soit une matrice $A \in \mathbb{C}^{m\times n}$. Alors $A \in \mathbb{C}_k^{m\times n}$ si et seulement si il existe $U \in \mathbb{C}^{m\times k}$ et $V \in \mathbb{C}^{n\times k}$ telles que*

$$A = UV^H. \tag{2.1.4}$$

La représentation (2.1.4) est intéressante car elle ne cache pas les redondances éventuelles de A, son rang k étant apparent, ce qui n'est pas le cas avec son écriture dense classique. De plus, dès lors que $k(m+n) \leqslant mn$, l'écriture (2.1.4) nécessite de stocker moins de coefficients que l'écriture dense de A.

Définition 2.1.3 (r_k-matrice)**.** *Toute matrice $A \in \mathbb{C}_k^{m\times n}$ est appelée matrice de rang faible (k) ou r_k-matrice si*

$$k(m+n) \leqslant mn. \tag{2.1.5}$$

Ainsi, si $A \in \mathbb{C}_k^{m\times n}$ est une r_k-matrice, l'écriture (2.1.4) constitue une *compression de la matrice A*. Cette première compression n'implique aucune approximation et est donc sans perte de précision. La figure 2.1.1 en donne une représentation graphique.

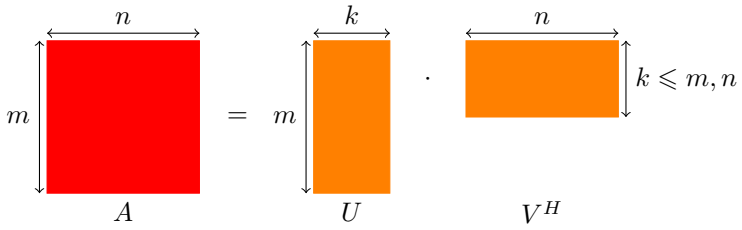


FIGURE 2.1.1 – Compression d'une matrice de rang k sous forme d'un produit UV^H.

Remarque 2.1.1. *Si $A \in \mathbb{C}_k^{n\times n}$, A est une r_k-matrice dès lors que*

$$k \leqslant \frac{n}{2}. \tag{2.1.6}$$

2.1.2 Recompression d'une r_k-matrice par troncature

Une r_k-matrice est donc une matrice pouvant être stockée en utilisant moins d'espace mémoire que sa représentation dense. Il est par ailleurs possible de réduire encore cet espace occupé par recompression. Cette recompression utilise la décomposition en valeurs singulières (en anglais *Singular Value Decomposition*, notée SVD), que nous allons redéfinir ici.

SVD et SVD réduite

Définition 2.1.4 (Matrice unitaire)**.** *Soit $U \in \mathbb{C}^{n\times n}$. U est une matrice unitaire si et seulement si elle vérifie les égalités*

$$UU^H = I_n \text{ et } U^HU = I_n, \tag{2.1.7}$$

où I_n est la matrice identité de $\mathbb{C}^{n\times n}$.

Autrement dit, une matrice unitaire est une matrice carrée inversible et dont l'inverse est sa transconjuguée. De telles matrices présentent des propriétés intéressantes et interviennent dans la définition de la SVD.

Propriété 2.1.2. *Soient $U \in \mathbb{C}^{n\times n}$ et $V \in \mathbb{C}^{n\times n}$ deux matrices unitaires. Alors :*

- U^H *et* V^H *sont unitaires,*
- UV *est unitaire.*

Rappelons à présent le théorème d'existence de la SVD.

Théorème 2.1.2 (Existence de la SVD)**.** *Pour toute matrice* $A \in \mathbb{C}^{m\times n}$*, il existe une factorisation de la forme*

$$A = U\Sigma V^H, \tag{2.1.8}$$

avec :

- $U \in \mathbb{C}^{m\times m}$ *une matrice unitaire,*
- $\Sigma \in \mathbb{R}^{m\times n}$ *la matrice diagonale contenant les* $p = \min(m,n)$ *valeurs singulières (positives ou nulles) de* A*,*
- $V \in \mathbb{C}^{n\times n}$ *une matrice unitaire.*

On appelle cette factorisation la décomposition en valeurs singulières (en anglais Singular Value Decomposition*, notée SVD) de* A*.*

Remarque 2.1.2. *Par convention, la diagonale de* Σ *est rangée dans l'ordre décroissant, soit*

$$\Sigma = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,p]\!]}\right) \text{ avec } \forall i,j \in [\![1,p]\!] \quad i \leqslant j \implies \sigma_i \geqslant \sigma_j, \tag{2.1.9}$$

où $p = \min(m,n)$*. Ceci implique l'unicité de* Σ*.*

Considérons à présent une matrice $A \in \mathbb{C}_k^{m\times n}$ non nulle, avec $k < p$ ($p = \min(m,n)$). A est donc de rang au plus k et possède au plus k valeurs singulières non nulles. D'après la remarque 2.1.2, la SVD de A s'écrit $A = U\Sigma V^H$ avec $\Sigma = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,p]\!]}\right)$ telle que

$$\begin{cases} \sigma_i > 0 & \text{pour } i \in [\![1,k]\!], \\ \sigma_i = 0 & \text{pour } i \in [\![k+1,p]\!]. \end{cases} \tag{2.1.10}$$

Pour $i \in [\![1,m]\!]$ et $j \in [\![1,n]\!]$, le coefficient a_{ij} de A s'écrit alors

$$a_{ij} = \sum_{l=1}^{p} u_{il}\sigma_l v_{lj}^H = \sum_{l=1}^{k} u_{il}\sigma_l v_{lj}^H. \tag{2.1.11}$$

Ainsi seules les k premières colonnes de U et V interviennent véritablement. On a donc

$$A = U_k\Sigma_k V_k^H \text{ avec } \begin{cases} U_k = (u_{ij})_{i\in[\![1,m]\!],j\in[\![1,k]\!]}, \\ \Sigma_k = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,k]\!]}\right), \\ V_k = (v_{ij})_{i\in[\![1,n]\!],j\in[\![1,k]\!]}. \end{cases} \tag{2.1.12}$$

On appelle cette écriture la *SVD réduite (de rang* k*) de* A*.*

Remarque 2.1.3. *La matrice* Σ_k *est alors diagonale carrée ; en revanche les matrices* U_k *et* V_k *ne sont plus nécessairement carrées, mais sont constituées de vecteurs orthogonaux deux à deux.*

SVD réduite d'une r_k-matrice

Considérons à présent une r_k-matrice A de $\mathbb{C}_k^{m\times n}$. En exploitant la représentation (2.1.4), il est possible d'effectuer une SVD réduite de $A = UV^H$ avec une complexité en $\mathcal{O}(k^2(m+n))$ [30].

Cette r_k-SVD consiste tout d'abord à effectuer les décompositions QR réduites

$$U = Q_U R_U \text{ et } V = Q_V R_V, \tag{2.1.13}$$

où $Q_U \in \mathbb{C}^{m\times k}$ et $Q_V \in \mathbb{C}^{n\times k}$ sont des matrices constituées de vecteurs orthogonaux deux à deux, $R_U \in \mathbb{C}^{k\times k}$ et $R_V \in \mathbb{C}^{k\times k}$ des matrices triangulaires supérieures. On a donc

$$A = (Q_U R_U)(Q_V R_V)^H = Q_U (R_U R_V^H) Q_V^H. \tag{2.1.14}$$

On applique alors une SVD réduite au produit $(R_U R_V^H) \in \mathbb{C}^{k\times k}$:

$$R_U R_V^H = U_U \Sigma V_V^H, \tag{2.1.15}$$

avec U_U et V_V des matrices à vecteurs orthogonaux deux à deux, et Σ la matrice diagonale des valeurs singulières du produit $R_U R_V^H$. On en déduit la décomposition

$$A = (Q_U U_U)\Sigma(Q_V V_V)^H, \tag{2.1.16}$$

soit, en posant $\widehat{U} = Q_U U_U$ et $\widehat{V} = Q_V V_V$,

$$A = \widehat{U}\Sigma\widehat{V}^H. \tag{2.1.17}$$

La représentation (2.1.17) est une SVD réduite de A, avec $\Sigma = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,k]\!]}\right) \in \mathbb{R}^{k\times k}$ telle que $0 \leqslant \sigma_k \leqslant \cdots \leqslant \sigma_1$. On peut en calculer le coût à l'aide de l'annexe B :

Factorisation QR de U	$-\frac{4}{3}k^3 + 4mk^2$
Factorisation QR de V	$-\frac{4}{3}k^3 + 4nk^2$
Produit $R_U R_V^H$	$k^2(2k-1)$
SVD de $R_U R_V^H$	$22k^3$
Multiplication $Q_U U_U$	$2mk^2 - mk$
Multiplication $Q_V V_V$	$2nk^2 - nk$
	$\sim \frac{64}{3}k^3 + 6(m+n)k^2$

Par comparaison, le coût d'une SVD réduite classique est de $14mn^2 + 8n^3$ pour une matrice de $\mathbb{C}^{m\times n}$.

On peut également comparer ces coûts pour une r_k-matrice $A \in \mathbb{C}_k^{m\times n}$ en posant avec $m = n$ et $k = n/2^l$ pour $l \in [\![0, 4]\!]$ (A est une r_k-matrice dès que $l \geqslant 1$, *cf.* remarque 2.1.1). Les résultats de cette comparaison sont disponibles en figure 2.1.2.

Troncature d'une r_k-matrice

La norme matricielle sur laquelle reposera la suite de notre étude est la *norme de Frobenius*, donnée en définition 1.4.2.

Le choix de cette norme est à replacer dans le contexte global de cette étude. En particulier, la norme de Frobenius nous permettra d'obtenir aisément la norme d'une matrice hiérarchique totale à partir de la connaissance de la norme de ses blocs individuels. En cela, cette norme semble particulièrement adaptée à la représentation par blocs.

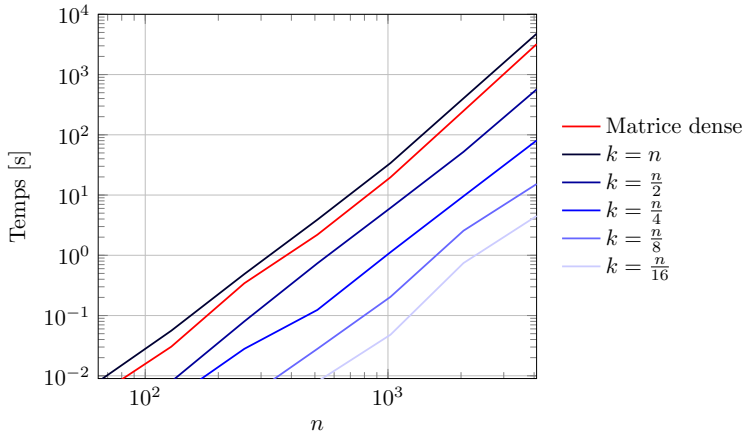


FIGURE 2.1.2 – Temps de calcul de la SVD réduite d'une matrice $A \in \mathbb{C}_k^{n\times n}$ en fonction du nombre de DDLs pour différentes valeurs du rang k.

Propriété 2.1.3. *On a alors les propriétés suivantes.*

- $\|\cdot\|_F$ *est unitairement invariante, c'est-à-dire que pour toutes matrices* $U \in \mathbb{C}^{m\times m}$, $A \in \mathbb{C}^{m\times n}$ *et* $V \in \mathbb{C}^{n\times n}$, *si* U *et* V *sont unitaires, alors*

$$\|A\|_F = \|UA\|_F = \|AV\|_F = \|UAV\|_F. \tag{2.1.18}$$

- *Si* U *est une matrice unitaire de* $\mathbb{C}^{n\times n}$, *alors* $\|U\|_F = \sqrt{n}$.

Remarque 2.1.4. *Soit une matrice* $A \in \mathbb{C}_k^{m\times n}$ *dont la SVD s'écrit* $A = U\Sigma V^H$ *et la SVD réduite* $A = U_k\Sigma_k V_k^H$. *Alors d'après* (2.1.18), *on a*

$$\|A\|_F = \|U\Sigma V^H\|_F = \|\Sigma\|_F. \tag{2.1.19}$$

Or, par définition de la SVD réduite, on a $\|\Sigma\|_F = \|\Sigma_k\|_F$. *Ainsi :*

$$\|A\|_F = \|\Sigma_k\|_F = \left(\sum_{i=1}^{k} \sigma_i^2\right)^{\frac{1}{2}}. \tag{2.1.20}$$

Théorème 2.1.3. *Soient* $A = U\Sigma V^H \in \mathbb{C}^{m\times n}$ *avec* U *et* V *unitaires, et telles que* $m \geqslant n$. *Ainsi* $\Sigma = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,n]\!]}\right) \in \mathbb{R}^{m\times n}$, *avec* $0 \leqslant \sigma_n \leqslant ... \leqslant \sigma_1$.

Soit $\|\cdot\|$ *une norme sur* $\mathbb{C}^{m\times n}$ *unitairement invariante. Alors pour* $k \in [\![0, n]\!]$, *on a :*

$$\min_{M\in\mathbb{C}_k^{m\times n}} \|A - M\| = \|A - A_k\| = \|\Sigma - \Sigma_k^{(n)}\|, \tag{2.1.21}$$

avec $A_k = U\Sigma_k^{(n)}V^H \in \mathbb{C}_k^{m\times n}$ *et* $\Sigma_k^{(n)} = \operatorname{diag}(\sigma_1, ..., \sigma_k, 0, ..., 0) \in \mathbb{R}^{m\times n}$.

En particulier, pour la norme de Frobenius, l'erreur absolue de A_k par rapport à A s'écrit :

$$\|A - A_k\|_{\mathrm{F}} = \left(\sum_{l=k+1}^{n} \sigma_l^2\right)^{\frac{1}{2}}. \tag{2.1.22}$$

Les valeurs singulières $(\sigma_i)_{1\leqslant i\leqslant n}$ vérifiant $0 \leqslant \sigma_n \leqslant ... \leqslant \sigma_1$, plus k est proche de n et plus $\|A - A_k\|_{\mathrm{F}}$ est proche de 0. A_k est donc une approximation de A, de rang $k \leqslant n$,

dont nous pouvons estimer l'erreur en connaissant les valeurs singulières (ou une SVD) de A.

L'égalité $\|A - A_k\|_{\mathrm{F}} = \|\Sigma - \Sigma_k^{(n)}\|_{\mathrm{F}}$ permet de déterminer le rang k nécessaire pour obtenir une approximation A_k de A à la précision $\varepsilon \geqslant 0$, c'est-à-dire telle que $\|A - A_k\|_{\mathrm{F}} < \varepsilon$. Les $n-k$ valeurs singulières les plus petites sont alors considérées *négligeables* au regard de la précision ε choisie.

En pratique, pour obtenir l'approximation A_k de $A \in \mathbb{C}^{m\times n}$, nous supprimons les $n-k$ valeurs singulières négligeables, ainsi que les $n-k$ dernières colonnes de U et V, ce qui permet de réduire l'espace mémoire nécessaire au stockage de A_k.

Par conséquent nous obtenons l'approximation

$$A \approx A_k = U_k \Sigma_k V_k^H \text{ avec } \begin{cases} U_k = (u_{ij})_{i\in[\![1,m]\!],j\in[\![1,k]\!]}, \\ \Sigma_k = \operatorname{diag}\left((\sigma_i)_{i\in[\![1,k]\!]}\right), \\ V_k = (v_{ij})_{i\in[\![1,n]\!],j\in[\![1,k]\!]}. \end{cases} \tag{2.1.23}$$

L'écriture (2.1.23) est appelée *troncature au rang k de A*. La figure 2.1.3 en donne une représentation graphique.

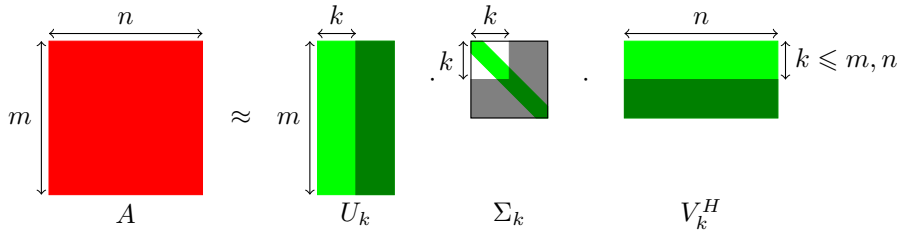


FIGURE 2.1.3 – Compression et troncature d'une matrice quelconque sous forme d'un produit $U_k\Sigma_k V_k^H$.

Nous pouvons, à titre d'exemple, assembler la matrice impédance Z dans le cas du cylindre avec un potentiel de simple couche (2D) ainsi que dans le cas de la sphère ($3D$). Il est alors possible d'appliquer la troncature à différentes précisions ε à un bloc de Z. Considérons la matrice impédance $Z \in \mathbb{C}^{2n\times 2n}$ décomposée en quatre blocs

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}. \tag{2.1.24}$$

Le bloc Z_{12} est, *a priori*, de rang $k = n$. Une fois Z_{12} assemblée, on lui applique une SVD suivie d'une troncature, ce qui nous fournit une approximation $\widetilde{Z}_{12} := U_k\Sigma_k V_k^H$ du bloc, de rang k. Ce test nous permet d'apprécier l'économie de mémoire obtenue par troncature en fonction de la valeur de ε choisie, tout en vérifiant l'erreur relative $\|U_k\Sigma_k V_k^H - Z_{12}\|_{\mathrm{F}} / \|Z_{12}\|_{\mathrm{F}}$ induite par cette troncature sur Z_{12} (*cf.* tableau 2.1.1).

La diminution du rang par troncature nous assure une compression efficace. On constate par ailleurs que l'erreur créée par troncature est de l'ordre de la précision ε choisie.

Dans la suite il sera implicite que toute r_k-matrice créée par compression à une précision $\varepsilon_{\text{comp.}}$ sera suivie d'une troncature à la même précision, sauf mention contraire.

2.1.3 Représentation en r_k-matrice d'une matrice de rang quelconque

D'après la définition 2.1.3, une matrice $A \in \mathbb{C}_k^{m\times n}$ est une r_k-matrice si k, m et n vérifient $k(m+n) \leqslant mn$. La matrice A peut alors s'écrire sous la forme (2.1.4), c'est-à-dire

$$A = UV^H,$$

		$\varepsilon = 10^{-1}$		$\varepsilon = 10^{-2}$		$\varepsilon = 10^{-3}$		$\varepsilon = 10^{-4}$	
Géométrie	n	Rang	Erreur	Rang	Erreur	Rang	Erreur	Rang	Erreur
Cylindre ($2D$)	731	3	$2{,}7 \times 10^{-2}$	5	$4{,}0 \times 10^{-3}$	9	$4{,}8 \times 10^{-4}$	12	$3{,}1 \times 10^{-5}$
	4360	3	$2{,}7 \times 10^{-2}$	5	$4{,}2 \times 10^{-3}$	10	$2{,}5 \times 10^{-4}$	13	$4{,}4 \times 10^{-5}$
	6812	3	$2{,}7 \times 10^{-2}$	5	$4{,}2 \times 10^{-3}$	10	$2{,}6 \times 10^{-4}$	13	$4{,}9 \times 10^{-5}$
Sphère ($3D$)	577	152	$4{,}7 \times 10^{-2}$	249	$4{,}9 \times 10^{-3}$	327	$4{,}9 \times 10^{-4}$	395	$4{,}9 \times 10^{-5}$
	2260	216	$4{,}8 \times 10^{-2}$	372	$4{,}9 \times 10^{-3}$	540	$4{,}9 \times 10^{-4}$	683	$5{,}0 \times 10^{-5}$
	5151	274	$4{,}8 \times 10^{-2}$	464	$5{,}0 \times 10^{-3}$	748	$5{,}0 \times 10^{-4}$	946	$5{,}0 \times 10^{-5}$

TABLEAU 2.1.1 – Rangs et erreurs relatives obtenus après troncatures de r_k-matrices pour différentes valeurs de ε.

avec $U \in \mathbb{C}^{m\times k}$ et $V \in \mathbb{C}^{n\times k}$. Ce format nécessite un espace de stockage en $\mathcal{O}(k(m+n))$.

Après troncature, on peut obtenir une approximation à une précision ε de A au format (2.1.23), c'est-à-dire

$$A \approx A_{k'} = U_{k'}\Sigma_{k'}V_{k'}^H, \tag{2.1.25}$$

avec $U_{k'} \in \mathbb{C}^{m\times k'}$ et $V_{k'} \in \mathbb{C}^{n\times k'}$ deux matrices dont les vecteurs sont orthogonaux deux à deux, $\Sigma_{k'} \in \mathbb{R}^{k'\times k'}$ la matrice diagonale des valeurs singulières de A non négligeables par rapport à la tolérance ε, $k' \leqslant k$. Ce second format sollicite un espace mémoire en $k'(m+n+\frac{1}{2})$.

Les représentations (2.1.4) et (2.1.23) sont donc équivalentes à ε près, et cette équivalence est même une égalité lorsque $k' = k$. On peut donc traiter des r_k-matrices de manière équivalente avec les deux formats à ε près.

Pour que le format (2.1.23) soit plus économe en mémoire que le format (2.1.4), il suffit que la troncature élimine une seule valeur singulière négligeable. En effet,

$$\begin{aligned}
k' \leqslant k-1 &\Leftrightarrow \left(m+n+\frac{1}{2}\right)k' \leqslant \left(m+n+\frac{1}{2}\right)(k-1) \\
&\Leftrightarrow \left(m+n+\frac{1}{2}\right)k' \leqslant (m+n)\,k + \frac{k}{2} - \left(m+n+\frac{1}{2}\right) \\
&\Leftrightarrow \left(m+n+\frac{1}{2}\right)k' - (m+n)\,k \leqslant \frac{k}{2} - \left(m+n+\frac{1}{2}\right) < 0 \text{ car } k \leqslant \min(m,n) \\
k' \leqslant k-1 &\Leftrightarrow \left(m+n+\frac{1}{2}\right)k' < (m+n)\,k.
\end{aligned}$$

Considérons à présent une matrice $A \in \mathbb{C}^{m\times n}$ de rang quelconque k. Alors $A \in \mathbb{C}_k^{m\times n}$. Effectuons la SVD

$$A = U\Sigma V^H,$$

avec U, Σ et V présentant les propriétés exprimées au théorème 2.1.2 et à la remarque 2.1.2. Le rang k correspond au nombre de valeurs singulières non nulles contenues sur la diagonale de Σ ($k \leqslant \min(m,n)$). Alors on a

$$A = A_k = U_k\Sigma_k V_k^H \text{ avec } \begin{cases} U_k = (u_{ij})_{i\in[\![1,m]\!],j\in[\![1,k]\!]}, \\ \Sigma_k = \operatorname{diag}\left((\sigma_i)\right)_{i\in[\![1,k]\!]}, \\ V_k = (v_{ij})_{i\in[\![1,n]\!],j\in[\![1,k]\!]}. \end{cases} \tag{2.1.26}$$

A_k est une troncature au rang k de A exactement égale à A, et il est possible, pour une valeur ε donnée, d'effectuer une troncature de A à un rang $k'(\varepsilon) \leqslant k$.

Définition 2.1.5. *Soit ε un réel strictement positif. Soit $A \in \mathbb{C}_k^{m\times n}$ une matrice de rang exactement k. On appelle* représentation en r_k-matrice de A *toute troncature de A sous la forme*

$$A_{k'} = U_{k'}\Sigma_{k'}V_{k'}^H \text{ avec } 1 \leqslant k' \leqslant k, \tag{2.1.27}$$

et telle que $\|U_{k'}\Sigma_{k'}V_{k'}^H - A\|_F \leqslant \varepsilon$. En particulier,

$$\begin{cases} A = A_{k'} & \text{si } k' = k, \\ A \approx A_{k'} & \text{si } k' < k. \end{cases} \tag{2.1.28}$$

Remarque 2.1.5. *Si A est représentée par une r_k-matrice $A_{k'} = U_{k'}\Sigma_{k'}V_{k'}^H$, alors d'après la remarque 2.1.4*

$$\|A\|_F \approx \|A_{k'}\|_F = \|\Sigma_{k'}\|_F = \sqrt{\sum_{i=1}^{k'} \sigma_i^2}. \tag{2.1.29}$$

Définition 2.1.6 (Taux de compression d'une représentation en r_k-matrice)**.** *Soit $A \in \mathbb{C}_k^{m\times n}$ une matrice de rang k et $A_{k'}$ une troncature au rang $k' \leqslant k$ de A. On appelle taux de compression de $A_{k'}$, le réel noté* $\mathrm{cr}(A_{k'})$ *et défini par*

$$\mathrm{cr}(A_{k'}) \overset{\text{déf.}}{=} \frac{k'(m+n+\frac{1}{2})}{mn}. \tag{2.1.30}$$

Le taux de compression correspond au rapport du nombre d'unités de mémoire occupées par les matrices $U_{k'}$, $\Sigma_{k'}$ et $V_{k'}$, sur le nombre d'unités de mémoire occupées par la matrice A dense.

Remarque 2.1.6. *On peut généraliser la définition du taux de compression. Ainsi :*

- *Si $A \in \mathbb{C}^{m\times n}$ est dense, alors* $\mathrm{cr}(A) = 1$.
- *Si $A \in \mathbb{C}_k^{m\times n}$ est représentée sous le format (2.1.4), alors* $\mathrm{cr}(A) = \frac{k(m+n)}{mn}$.

En particulier, une représentation en r_k-matrice $A_{k'}$ d'une matrice A n'est intéressante que si

$$\mathrm{cr}(A_{k'}) < 1. \tag{2.1.31}$$

Dans la suite de cette section, nous traiterons de r_k-matrices au format (2.1.23), et nous parlerons indifféremment de r_k-matrices et de représentation en r_k-matrices de matrices denses.

2.1.4 Arithmétique des r_k-matrices

Afin de profiter au maximum de la structure de r_k-matrice, il devient nécessaire d'adapter et de redéfinir certaines opérations élémentaires.

Addition arrondie

Soient deux r_k-matrices $A = U_A\Sigma_A V_A^H$ dans $\mathbb{C}_{k_A}^{m\times n}$ et $B = U_B\Sigma_B V_B^H$ dans $\mathbb{C}_{k_B}^{m\times n}$ au format (2.1.23). Nous souhaitons effectuer la somme $C = A + B$ en obtenant C sous la forme d'une r_k-matrice au même format. On a

$$A + B = \begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} \Sigma_A & 0 \\ 0 & \Sigma_B \end{bmatrix} \begin{bmatrix} V_A & V_B \end{bmatrix}^H. \tag{2.1.32}$$

La matrice $\begin{bmatrix}\Sigma_A & 0\\ 0 & \Sigma_B\end{bmatrix}$ est bien diagonale réelle, mais les matrices $\begin{bmatrix}U_A & U_B\end{bmatrix}$ et $\begin{bmatrix}V_A & V_B\end{bmatrix}$ ne sont pas constituées de vecteurs orthogonaux deux à deux, et l'équation (2.1.32) ne constitue donc pas une écriture correspondant au format (2.1.23).

Posons alors

$$Z_U = U_A^H U_B \text{ et } Y_U = U_B - U_A Z_U. \tag{2.1.33}$$

La décomposition QR réduite de Y_U donne $Y_U = Q_U R_U$. Ainsi, nous obtenons

$$\begin{bmatrix}U_A & U_B\end{bmatrix} = \begin{bmatrix}U_A & Q_U\end{bmatrix}\begin{bmatrix}I & Z_U\\ 0 & R_U\end{bmatrix}. \tag{2.1.34}$$

Les mêmes opérations effectuées sur la matrice $\begin{bmatrix}V_A & V_B\end{bmatrix}$ nous donnent

$$\begin{bmatrix}V_A & V_B\end{bmatrix} = \begin{bmatrix}V_A & Q_V\end{bmatrix}\begin{bmatrix}I & Z_V\\ 0 & R_V\end{bmatrix}. \tag{2.1.35}$$

Ainsi

$$A + B = \begin{bmatrix}U_A & Q_U\end{bmatrix}\begin{bmatrix}\Sigma_A + Z_U\Sigma_B Z_V^H & Z_U\Sigma_B R_V^H\\ R_U\Sigma_B Z_V^H & R_U\Sigma_B R_V^H\end{bmatrix}\begin{bmatrix}V_A & Q_V\end{bmatrix}^H. \tag{2.1.36}$$

En effectuant une SVD de la matrice

$$\begin{bmatrix}\Sigma_A + Z_U\Sigma_B Z_V^H & Z_U\Sigma_B R_V^H\\ R_U\Sigma_B Z_V^H & R_U\Sigma_B R_V^H\end{bmatrix} = U\Sigma V^H, \tag{2.1.37}$$

on obtient

$$A + B = \begin{bmatrix}U_A & Q_U\end{bmatrix} U\Sigma V^H \begin{bmatrix}V_A & Q_V\end{bmatrix}^H = \left(\begin{bmatrix}U_A & Q_U\end{bmatrix} U\right)\Sigma\left(\begin{bmatrix}V_A & Q_V\end{bmatrix} V\right)^H. \tag{2.1.38}$$

soit en posant $U_C = \begin{bmatrix}U_A & Q_U\end{bmatrix} U$, $\Sigma_C = \Sigma$ et $V_C = \begin{bmatrix}V_A & Q_V\end{bmatrix} V$,

$$C = U_C\Sigma_C V_C^H. \tag{2.1.39}$$

Les matrices U_C et V_C sont constituées de vecteurs deux à deux orthogonaux, le résultat de cette addition de deux r_k-matrices au format (2.1.23) est donc lui-même une r_k-matrice au même format.

En appliquant cette méthode, la r_k-matrice C ainsi obtenue est exactement égale à la somme de A et B. Cependant, son rang k_C peut croître par rapport à k_A et k_B, puisque $k_C = k_A + k_B$; de même, $\mathrm{cr}(C) = \mathrm{cr}(A) + \mathrm{cr}(B)$.

Pour réduire le rang k_C et le taux de compression associé, il peut être intéressant de compléter la SVD de l'étape (2.1.37) d'une troncature à une précision ε à choisir.

Définition 2.1.7 (Addition arrondie de deux r_k-matrices)**.** *Soient deux r_k-matrices $A = U_A\Sigma_A V_A^H$ dans $\mathbb{C}_{k_A}^{m\times n}$ et $B = U_B\Sigma_B V_B^H$ dans $\mathbb{C}_{k_B}^{m\times n}$ au format (2.1.23). Alors l'opération constituée des étapes (2.1.32) à (2.1.39) fournit une r_k-matrice $C = U_C\Sigma_C V_C^H$ vérifiant*

$$C \coloneqq A + B, \tag{2.1.40}$$

et l'application d'une r_k-SVD à précision $\varepsilon > 0$ sur la r_k-matrice C permet d'obtenir une approximation $\widetilde{C}$ telle que

$$\|C - \widetilde{C}\|_F \leqslant \varepsilon. \tag{2.1.41}$$

$\widetilde{C}$ constitue donc une approximation à ε près de la somme $A+B$. On note alors

$$\widetilde{C} \coloneqq A \oplus B, \tag{2.1.42}$$

*et on appelle cette opération l'*addition arrondie *(à ε près) des r_k-matrices A et B.*

On peut alors calculer le coût de cette opération (sans la troncature) :

Multiplication $Z_U = U_A^H U_B$	$k_A k_B(2m-1)$
Multiplication $Z_V = V_A^H V_B$	$k_A k_B(2n-1)$
Construction de $Y_U = U_B - U_A Z_U$	$2mk_Ak_B$
Construction de $Y_V = V_B - V_A Z_V$	$2nk_Ak_B$
Factorisation QR de $Y_U = Q_U R_U$	$-\frac{4}{3}k_B^3 + 4mk_B^2$
Factorisation QR de $Y_V = Q_V R_V$	$-\frac{4}{3}k_B^3 + 4nk_B^2$
Multiplication $Z_U \Sigma_B Z_V^H$	$2k_Ak_B^2$
Multiplication $Z_U \Sigma_B R_V^H$	$2k_B^3 + k_B(k_B - k_A)$
Multiplication $R_U \Sigma_B Z_V^H$	$2k_B^3 + k_B(k_A - k_B)$
Multiplication $R_U \Sigma_B R_V^H$	$2k_B^3$
Somme $\Sigma_A + Z_U \Sigma_B Z_V^H$	k_A^2
SVD	$22(k_A+k_B)$
Multiplication $\begin{bmatrix} U_A & Q_U \end{bmatrix} U$	$m(k_A+k_B)(2(k_A+k_B)-1))$
Multiplication $\begin{bmatrix} V_A & Q_V \end{bmatrix} V$	$n(k_A+k_B)(2(k_A+k_B)-1))$
	$\sim (m+n)(2k_A^2+6k_B^2+6k_Ak_B)+k_B^3+k_Ak_B^2$

Ce coût est à comparer avec le coût théorique d'une addition de A et B précédée des multiplications $U_A\Sigma_A V_A^H$ et $U_B\Sigma_B V_B^H$, et suivie d'une SVD de la matrice C résultante, soit environ $14mn^2 + 8n^3 + 2m(k_A^2+k_B^2)$, en ne considérant que les termes dominants. Dans l'hypothèse où k_A et k_B sont faibles devant m et n, l'addition arrondie améliore la complexité.

On remarque par ailleurs que k_B intervient plus que k_A dans le coût global de cette opération, aussi il peut être judicieux de s'assurer que $k_B \leqslant k_A$.

Remarque 2.1.7 (Somme de deux r_k-matrices de représentation (2.1.4))**.** *Soient deux r_k-matrices $A = U_A V_A^H \in \mathbb{C}_{k_A}^{m\times n}$ et $B = U_B V_B^H \in \mathbb{C}_{k_B}^{m\times n}$. Si on construit les matrices $U = \begin{bmatrix} U_A & U_B \end{bmatrix} \in \mathbb{C}^{m\times(k_A+k_B)}$ et $V = \begin{bmatrix} V_A & V_B \end{bmatrix} \in \mathbb{C}^{n\times(k_A+k_B)}$, alors*

$$UV^H = \begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} V_A & V_B \end{bmatrix}^H \tag{2.1.43}$$

$$= \begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} V_A^H \\ V_B^H \end{bmatrix} \tag{2.1.44}$$

$$= U_A V_A^H + U_B V_B^H \tag{2.1.45}$$

$$UV^H = A + B \tag{2.1.46}$$

*Cette somme de deux r_k-matrices de représentation (2.1.4) peut s'avérer utile et n'implique aucune approximation supplémentaire. Nous en aurons en particulier l'utilité lors du développement de la méthode de compression HCA (*cf. *section 3.2).*

Produit r_k-matrice-vecteur

Soit $A = U\Sigma V^H$ une r_k-matrice de $\mathbb{C}_k^{m\times n}$ et $\boldsymbol{x}$ un vecteur de $\mathbb{C}^n$. Nous souhaitons réaliser le produit de A par $\boldsymbol{x}$, soit $\boldsymbol{b} = A\boldsymbol{x}$ avec $\boldsymbol{b}$ un vecteur de $\mathbb{C}^m$. Cette opération s'effectue suivant trois étapes, réalisées de droite à gauche, soit

$$\boldsymbol{x}' = V^H\boldsymbol{x}, \qquad \boldsymbol{x}' \in \mathbb{C}^k, \tag{2.1.47}$$
$$\boldsymbol{x}'' = \Sigma\boldsymbol{x}', \qquad \boldsymbol{x}'' \in \mathbb{C}^k, \tag{2.1.48}$$
$$\boldsymbol{b} = U\boldsymbol{x}'', \qquad \boldsymbol{b} \in \mathbb{C}^m. \tag{2.1.49}$$

Le coût théorique de cette opération est alors :

$$\begin{array}{ll} \text{Produit } x' = V^H x & k(2n-1) \\ \text{Produit } x' = V^H x & k \\ \text{Produit } x' = V^H x & m(2k-1) \\ \hline & \sim 2k(m+n), \end{array}$$

soit une compléxité en $\mathcal{O}(k(m+n))$. Ce coût est à comparer avec le coût d'un produit matrice-vecteur classique, impliquant $m(2n-1)$ opérations, soit une complexité en $\mathcal{O}(mn)$, moins intéressante que la complexité précédemment obtenue dès lors que $k \ll m, n$.

Multiplication de deux r_k-matrices

Soient deux r_k-matrices $A = U_A\Sigma_A V_A^H$ dans $\mathbb{C}_{k_A}^{m\times n}$ et $B = U_B\Sigma_B V_B^H$ dans $\mathbb{C}_{k_B}^{n\times p}$ au format (2.1.23). Nous souhaitons effectuer l'opération $C = AB$ telle que C soit une r_k-matrice au format (2.1.23). On pose

$$M = V_A^H U_B. \tag{2.1.50}$$

M est alors de dimension $k_A \times k_B$, *a priori* petite. La SVD de M donne $M = U_M\Sigma_M V_M^H$ avec les propriétés habituelles pour les matrices U_M, Σ_M et V_M. Posons alors

$$U = U_A\Sigma_A U_M, \tag{2.1.51}$$
$$\Sigma = \Sigma_M, \tag{2.1.52}$$
$$V = V_B\Sigma_B V_M. \tag{2.1.53}$$

En appliquant une r_k-SVD à la matrice $U\Sigma V^H$, on obtient la r_k-matrice

$$U\Sigma V^H = U_C\Sigma_C V_C^H = C. \tag{2.1.54}$$

Remarque 2.1.8. *Cette opération est théoriquement exacte, mais tout comme l'addition arrondie $\oplus$, il est possible pour un réel positif ε donné de compléter la r_k-SVD de l'étape (2.1.54) d'une troncature. Cette multiplication devient alors arrondie et notée $\odot$.*

On peut calculer le coût de cette multiplication l'aide de l'annexe B :

$$\begin{array}{ll} \text{Multiplication } V_A^H U_B & k_A k_B(2n-1) \\ \text{SVD de } M = V_A^H U_B & 14k_A k_B^2 + 8k_B^3 \\ \text{Multiplication } U_A\Sigma_A U_M & k_A k_M + mk_A(2k_A-1) \\ \text{Multiplication } V_B\Sigma_B V_M & k_B k_M + pk_B(2k_B-1) \\ \hline & \sim 8k_B^3 + 14k_A k_B^2 + 2nk_A k_B + 2mk_A^2 + 2pk_B^3 \end{array}$$

Dans l'hypothèse où les rangs k_A et k_B sont petits devant m, n et p, ce coût est plus intéressant que celui d'une multiplication classique de matrices denses, soit $mp(2n-1)$.

2.2 Construction d'une partition admissible

2.2.1 Partition admissible

De manière générale, une matrice $A \in \mathbb{C}^{m\times n}$ n'est pas approchée par une r_k-matrice à une précision ε dans sa globalité. Seuls certains blocs bien choisis de A peuvent bénéficier de cette compression sans pertes dommageables de précision sur la matrice totale. Ces blocs sont décrits selon des critères bien déterminés que nous allons développer dans cette section.

Partition et blocs d'indices

Définition 2.2.1 (Partition, blocs)**.** *Soient $I, J \subset \mathbb{N}$. Une sous-partie $P \subset \mathcal{P}(I \times J)$ est une partition de $I \times J$ si*

- $I \times J = \bigcup_{b\in P} b$,
- $\forall b \in P \quad b \neq \emptyset$,
- $\forall b, b' \in P \quad b \cap b' \neq \emptyset \implies b = b'$.

Autrement dit, une partition de $I \times J$ est un ensemble de parties non vides de $I \times J$ deux à deux disjointes et qui recouvrent $I \times J$.

On appelle alors les éléments $b \in P$ blocs d'indices*, ou plus simplement* blocs*.*

Notations. *Notons I (respectivement J) l'ensemble des indices de lignes (resp. colonnes) d'une matrice $A \in \mathbb{C}^{m\times n}$ c'est-à-dire $I \coloneqq [\![1, m]\!]$ et $J \coloneqq [\![1, n]\!]$.*

Soit P une partition de $I \times J$. Soit l (respectivement c) une sous-partie de I (resp. J) vérifiant $b \coloneqq l \times c \in P$. On note alors $A_{|b}$ ou $A_{|l\times c}$ la restriction de la matrice A aux indices du bloc b.

Par ailleurs, on pourra noter $\mathbb{C}^I$ l'ensemble des vecteurs de $\mathbb{C}^m$ dont les indices des composantes sont dans I. De même, on notera $\mathbb{C}^{I\times J}$ l'ensemble des matrices de $\mathbb{C}^{m\times n}$ dont les couples d'indices des coefficients sont dans $I \times J$.

Admissibilité d'une partition

Soient $A \in \mathbb{C}^{I\times J}$ et soit P une partition de $I \times J$. Soit $b \coloneqq l \times c \in P$. La restriction $A_{|b}$ est dite *de petite taille* au regard d'une dimension maximale $n_{\max} \in \mathbb{N}$ si l'inégalité

$$\min(\mathrm{Card}(l), \mathrm{Card}(c)) \leqslant n_{\max} \tag{2.2.1}$$

est vérifiée.

La partition P est intéressante en terme de compression si pour tout bloc b de P et pour un $n_{\max}$ donné, $A_{|b}$ est soit de petite taille, soit approchable par une r_k-matrice de taux de compression inférieur à 1.

Définition 2.2.2 (Bloc admissible)**.** *Un bloc $b \in P$ est dit admissible si la restriction $A_{|b}$ peut être approchée par une r_k-matrice.*

L'admissibilité d'un bloc est conditionnée par un *critère d'admissibilité* qui dépend du problème traité.

Ce critère se doit de vérifier les conditions suivantes :

- Si b est un bloc admissible, alors les valeurs singulières de $A_{|b}$ ont une décroissance exponentielle.

- La condition d'admissibilité doit pouvoir être vérifiée pour tout bloc de $l \times c \in \mathcal{P}(I \times J)$ avec une complexité algorithmique de $\mathcal{O}(\mathrm{Card}(l) + \mathrm{Card}(c))$.
- Si b est un bloc admissible, alors toute sous-partie $b' \subset b$ est un bloc admissible.

On peut alors définir la notion d'admissibilité d'une partition $P \in \mathcal{P}(I \times J)$.

Définition 2.2.3 (Partition admissible)**.** *Une partition $P \in \mathcal{P}(I \times J)$ est dite* admissible *si chacun des blocs b de P est soit admissible, soit de petite taille au regard d'une valeur n_{max} choisie.*

2.2.2 Boîte englobante et η-admissibilité

Dans notre cas, les matrices que nous allons chercher à compresser correspondent aux interactions électromagnétiques entre différents points du plan ou de l'espace. La partition choisie aura à cœur de respecter des critères de géométrie. En effet, compresser l'information concernant les interactions entre deux ensembles d'éléments géométriquement proches risquerait de trop nuire à la précision globale de la matrice impédance et ne conduirait pas à un gain mémoire intéressant [30].

Soit $\Omega_I \coloneqq (\boldsymbol{p}_i)_{i\in I}$ une liste de points de $\mathbb{R}^d$ où d est la dimension considérée ($d = 2$ ou 3). On définit alors, en se plaçant dans la base canonique euclidienne $(\boldsymbol{u}_x, \boldsymbol{u}_y)$ pour $d = 2$ et $(\boldsymbol{u}_x, \boldsymbol{u}_y, \boldsymbol{u}_z)$ pour $d = 3$,

$$x_I^- = \min_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_x \text{ et } x_I^+ = \max_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_x, \tag{2.2.2}$$

$$y_I^- = \min_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_y \text{ et } y_I^+ = \max_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_y, \tag{2.2.3}$$

$$z_I^- = \min_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_z \text{ et } z_I^+ = \max_{i\in I} \boldsymbol{p}_i \cdot \boldsymbol{u}_z \text{ si } d = 3, \tag{2.2.4}$$

et les deux points

$$\boldsymbol{p}_I^- = \begin{pmatrix} x_I^- & y_I^- \end{pmatrix}^T \text{ et } \boldsymbol{p}_I^+ = \begin{pmatrix} x_I^+ & y_I^+ \end{pmatrix}^T \qquad \text{pour } d = 2, \tag{2.2.5}$$

$$\boldsymbol{p}_I^- = \begin{pmatrix} x_I^- & y_I^- & z_I^- \end{pmatrix}^T \text{ et } \boldsymbol{p}_I^+ = \begin{pmatrix} x_I^+ & y_I^+ & z_I^+ \end{pmatrix}^T \qquad \text{pour } d = 3. \tag{2.2.6}$$

Alors si $d = 2$ (resp. $d = 3$), la boîte englobante de Ω_I, notée $\mathcal{B}_I$, est le rectangle (resp. le pavé droit) délimité par $\boldsymbol{p}_I^-$ et $\boldsymbol{p}_I^+$. On définit les notions de diamètre d'une boîte et de distance entre deux boîtes.

Définition 2.2.4 (Diamètre d'une boîte englobante, distance entre deux boîtes englobantes)**.** *Soient $\Omega_I \coloneqq (\boldsymbol{p}_i)_{i\in I}$ et $\Omega_J \coloneqq (\boldsymbol{q}_j)_{j\in J}$ deux ensembles de points. Soient $\mathcal{B}_I$ et $\mathcal{B}_J$ leur boîte englobante respective. Le diamètre de $\mathcal{B}_I$ est défini par*

$$\mathrm{diam}(\mathcal{B}_I) \overset{\text{déf.}}{=} \max_{i,j\in I} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2 , \tag{2.2.7}$$

et la distance entre $\mathcal{B}_I$ et $\mathcal{B}_J$ s'écrit

$$\mathrm{dist}(\mathcal{B}_I, \mathcal{B}_J) \overset{\text{déf.}}{=} \min_{i\in I, j\in J} \|\boldsymbol{p}_i - \boldsymbol{q}_j\|_2 . \tag{2.2.8}$$

Notations. *Si P est une partition de $I \times J$ et l (respectivement c) une sous-partie de I (resp. J) vérifiant $b \coloneqq l \times c \in P$, on peut également noter $\mathcal{B}_l$ (resp. $\mathcal{B}_c$) la boîte englobante des points dont les indices sont dans l (resp. dans c).*

On peut alors définir pour le bloc $b \coloneqq l \times c \in P$ une condition d'admissibilité relative à une valeur η.

Définition 2.2.5 (η-admissibilité)**.** *Soit η un réel strictement positif. Alors le bloc $b \coloneqq l \times c$ est dit η-admissible s'il vérifie la condition*

$$\min(\operatorname{diam}(l), \operatorname{diam}(c)) \leqslant \eta \operatorname{dist}(\mathcal{B}_l, \mathcal{B}_c). \tag{2.2.9}$$

Ce critère de η-admissibilité va nous permettre de définir la hiérarchie de la matrice à compresser, et en particulier de déterminer les zones de cette matrice qu'il sera possible de compresser.

2.3 Structure arborescente d'une $\mathcal{H}$-matrice

2.3.1 Notion d'arbre

La hiérarchie d'une $\mathcal{H}$-matrice est décrite par une *structure d'arbre.*

Nous considérons un ensemble de nœuds N contenant le nœud racine r, et une application S qui définit l'ensemble des descendants directs de chaque nœud.

Définition 2.3.1 (Arbre)**.** *Soient N un ensemble de nœuds, r une racine et* S *l'application qui va de N dans l'ensemble des sous-ensembles de $\mathcal{P}(N)$ et définit l'ensemble des descendants directs de chaque nœud. Le triplet $T \coloneqq (N, r, \mathrm{S})$ est un arbre si pour chaque nœud il existe un unique chemin menant à la racine r, c'est-à-dire*

$$\forall t \in N \backslash \{r\} \quad \exists! (t_i)_{i \in [\![0,m]\!]} \in N^{m+1} \quad t_0 = r,\ t_m = t \text{ et } t_{i+1} = \mathrm{S}(t_i). \tag{2.3.1}$$

Dans ce cas, on appelle la suite $(t_i)_{i \in [\![0,m]\!]}$ le chemin du nœud t dans T, et m la profondeur de t dans T.

Notations. *Soient $T \coloneqq (N, r, S)$ un arbre et $t \in N$ un nœud de T de chemin $(t_i)_{i \in [\![0,m]\!]}$. On définit les notations*

- $\mathrm{root}(T)$: *racine de T ($r \coloneqq \mathrm{root}(T)$).*
- $\mathrm{sons}(t)$: *ensemble des descendants directs (ou enfants) du nœud t.*
- $\mathrm{depth}(t)$: *profondeur du nœud t dans l'arbre T ($\mathrm{depth}(r) = 0$ et $\mathrm{depth}(t) = m$).*

La définition 2.3.1 implique les propriétés suivantes.

Propriété 2.3.1. *Soit $T = (N, r, \mathrm{S})$ un arbre.*

- *Soit $t \in N$ de chemin $(t_i)_{i \in [\![0,m]\!]}$ dans T ($r = t_0$ et $t = t_m$). Alors*

$$\forall i, j \in [\![0, m]\!] \quad i \neq j \implies t_i \neq t_j. \tag{2.3.2}$$

- *r ne possède pas d'ascendant, c'est-à-dire*

$$\forall t \in N \quad r \notin \mathrm{sons}(t). \tag{2.3.3}$$

- *Tout nœud différent de la racine r possède un unique ascendant direct (ou parent), c'est-à-dire*

$$\forall t \in N \backslash \{r\} \quad \exists! p \in N \quad t \in \mathrm{sons}(p). \tag{2.3.4}$$

Définition 2.3.2 (Feuille)**.** *Une feuille est un nœud $t \in N$ ne possédant pas de descendant, c'est-à-dire tel que $\mathrm{sons}(t) = \emptyset$. On définit l'ensemble des feuilles de T, noté $\mathcal{L}(T)$, par*

$$\mathcal{L}(T) \overset{\text{déf.}}{=} \{t \in N \mid \mathrm{sons}(t) = \emptyset\}. \tag{2.3.5}$$

Dans la suite, à chaque nœud seront associées des données (ici des ensembles d'indices). Ces données sont appelées l'*étiquette* du nœud et on définit alors la notion d'*arbre à étiquettes*.

Définition 2.3.3 (Arbre à étiquettes)**.** *Soient N et L des sous-ensembles finis non-vides de $\mathbb{N}$, $r \in N$, $\mathrm{S} : N \to \mathcal{P}(N)$ et $l : N \to L$. Alors $T = (N, r, \mathrm{S}, l, L)$ est un arbre à étiquettes si le triplet (N, r, S) est un arbre.*

Les notations précédentes sont conservées pour les arbres à étiquettes. De plus, on définit L comme l'ensemble des étiquettes de T et $l(t) \in L$ est appelée l'étiquette de t, notée $\widehat{t}$.

2.3.2 Construction d'un *cluster tree*

Un *cluster tree* est un objet permettant de construire une partition hiérarchique des indices de lignes et de colonnes d'une matrice. Cette partition est basée sur un algorithme de *clustering* qui regroupe les indices de DDLs sur des critères géométriques liés à la position de ces DDLs.

Définition 2.3.4 (*Cluster tree*)**.** *Soit $T = (N, r, \mathrm{S}, l, L)$ un arbre à étiquettes et I un ensemble d'indices. T est un* cluster tree *s'il vérifie les conditions suivantes :*

- $\widehat{r} = I$,
- $\forall t \in N \backslash \mathcal{L}(T) \quad \widehat{t} = \bigcup_{s \in S(t)} \widehat{s}$,
- $\forall t \in N \quad \forall s_1, s_2 \in \mathrm{sons}(t) \quad s_1 \neq s_2 \implies \widehat{s}_1 \cap \widehat{s}_2 = \emptyset$.

Les nœuds $t \in N$ d'un cluster tree *sont appelés* clusters.

Notations. *Un arbre à étiquette T qui est un* cluster tree *pour l'ensemble d'indices I sera noté T_I, et on utilisera l'abréviation $t \in T_I$ pour $t \in N$.*

Remarque 2.3.1. *En pratique l'ensemble d'indices I contient les indices des fonctions de base, soit $I \coloneqq [\![1, n]\!]$ avec n la dimension de l'espace de projection V_n.*

Numériquement, un *cluster tree* est défini par récursion.

2.3.3 *Block cluster trees*

Définition

Maintenant que nous avons défini une hiérarchie avec les *cluster trees*, nous allons désormais voir une nouvelle structure qui définit la hiérarchie de blocs de la $\mathcal{H}$-matrice. On obtient une décomposition de la matrice en blocs de tailles différentes, et chaque bloc représente les interactions entre deux ensembles de DDLs.

Définition 2.3.5 (*Block cluster tree*)**.** *Soient T_I et T_J deux* cluster trees *pour les ensembles d'indices I et J. T est un* block cluster tree *pour T_I et T_J si il vérifie les conditions suivantes :*

- $\mathrm{root}(T) \coloneqq (\mathrm{root}(T_I), \mathrm{root}(T_J))$.
- *Chaque nœud $b \in T$ s'écrit $b \coloneqq (t, s)$ avec $t \in T_I$ et $s \in T_J$,*

- *Pour tout* $b \coloneqq (t, s) \in T$.

$$\text{sons}(b) \coloneqq \begin{cases} \{(t', s') : t' \in \text{sons}(t), s' \in \text{sons}(s)\} \text{ si } \text{sons}(t) \neq \emptyset \text{ et } \text{sons}(s) \neq \emptyset, \\ \{(t', s) : t' \in \text{sons}(t)\} \text{ si } \text{sons}(t) \neq \emptyset \text{ et } \text{sons}(s) = \emptyset, \\ \{(t, s') : s' \in \text{sons}(s)\} \text{ si } \text{sons}(t) = \emptyset \text{ et } \text{sons}(s) \neq \emptyset, \\ \emptyset \text{ si } \text{sons}(t) = \emptyset \text{ et } \text{sons}(s) = \emptyset. \end{cases} \tag{2.3.6}$$

- *Si* $\text{sons}(b) = \emptyset$ *alors* $b \in \mathcal{L}(T)$ *(*b *est une feuille de* T*).*
- *L'étiquette d'un nœud* $b \coloneqq (t, s)$ *est définie par*

$$\widehat{b} \coloneqq \widehat{t} \times \widehat{s} \subseteq I \times J. \tag{2.3.7}$$

Les nœuds de T *sont appelés* block clusters*, et on note* $T_{I\times J}$ *le* block cluster tree *pour* T_I *et* T_J*.*

Comme pour le *cluster tree*, le *block cluster tree* est défini par récursion. $T_{I\times J}$ peut alors représenter la hiérarchie des interactions entre les deux ensembles de DDLs indexés par I et par J.

Ces définitions nous permettent de parler de façon indifférenciée d'un bloc de $\mathcal{H}$-matrice comme d'un *block cluster*.

Parcimonie

Il est alors possible de formaliser la complexité hiérarchique d'un *block cluster tree* grâce à l'introduction de la *constante de rareté*.

Définition 2.3.6 (Constante de rareté)**.** *Soient* T_I *et* T_J *deux* cluster trees *pour les ensembles d'indices* I *et* J*, et soit* $T_{I\times J}$ *un* block cluster tree *pour* $I \times J$*.*

- *Le nombre de blocs* $(t, s) \in T_{I\times J}$ *associés à un* cluster $t \in T_I$ *donné s'écrit*

$$c^l_{sp}(T_{I\times J}, t) \overset{\text{déf.}}{=} \text{Card}(\{s \subset J \,|\, (t, s) \in T_{I\times J}\}). \tag{2.3.8}$$

- *De même, le nombre de blocs* $(t, s) \in T_{I\times J}$ *associés à un* cluster $s \in T_J$ *donné s'écrit*

$$c^c_{sp}(T_{I\times J}, s) \overset{\text{déf.}}{=} \text{Card}(\{t \subset I \,|\, (t, s) \in T_{I\times J}\}). \tag{2.3.9}$$

- *On définit enfin la* constante de rareté *du* block cluster tree $T_{I\times J}$ *par*

$$c_{sp}(T_{I\times J}) \overset{\text{déf.}}{=} \max\left\{\max_{t\in T_I} c^l_{sp}(T_{I\times J}, t), \max_{s\in T_J} c^c_{sp}(T_{I\times J}, s)\right\}. \tag{2.3.10}$$

Lorsque la constante de rareté est grande, la hiérarchie de l'arbre est "complexe" et il est moins aisé de le parcourir. Ceci se traduit sur la complexité algorithmique des opérations effectuées sur un *block cluster tree* [30], comme nous le verrons dans la suite de ce chapitre.

Admissibilité

La notion d'*admissibilité* permet de déterminer s'il est possible ou non de compresser un bloc en limitant les pertes de précision. Il s'agit d'une extension de la définition de l'η-admissibilité à un bloc matriciel.

Soit $A \in \mathbb{C}^{I\times J}$. On note $A_{|b}$ ou $A_{|t\times s}$ la restriction de A à ses coefficients ayant leurs indices dans l'étiquette $\widehat{b} \coloneqq \widehat{t}\times\widehat{s} \subseteq I\times J$ du bloc $b \coloneqq (t, s)$. On parle alors de η-admissibilité

pour $A_{|b}$ si b est η-admissible. Le bloc matriciel $A_{|b}$ peut alors, s'il est admissible (ou η-admissible), être compressé et représenté sous la forme d'une r_k-matrice.

Par extension, nous noterons $\Omega_{\widehat{t}}$ (resp. $\Omega_{\widehat{s}}$) l'ensemble des domaines contenant les DDLs indexés par $\widehat{t}$ (resp. $\widehat{s}$), et nous dirons que la paire de domaines $(\Omega_{\widehat{t}}, \Omega_{\widehat{s}})$ est une paire de domaines admissible si le bloc $b := (t, s)$ est admissible.

Hiérarchie d'une $\mathcal{H}$-matrice

La hiérarchie d'une $\mathcal{H}$-matrice donnée est alors définie à la fois par le *block cluster tree* qui représente la répartition spatiale des DDLs et par une condition de η-admissibilité entre les nœuds du *block cluster tree*. La matrice résultante représente les interactions entre les différents ensembles de DDLs, comme illustré dans l'exemple en figure 2.3.1.

En pratique nous utilisons une hiérarchie d'*arbre binaire* pour les *cluster trees* T_I et T_J : les ensembles d'indices I et J subissent des bissections successives, chacune d'entre elles correspondant au chemin direct (ou à la branche) entre un nœud et ses deux descendants directs.

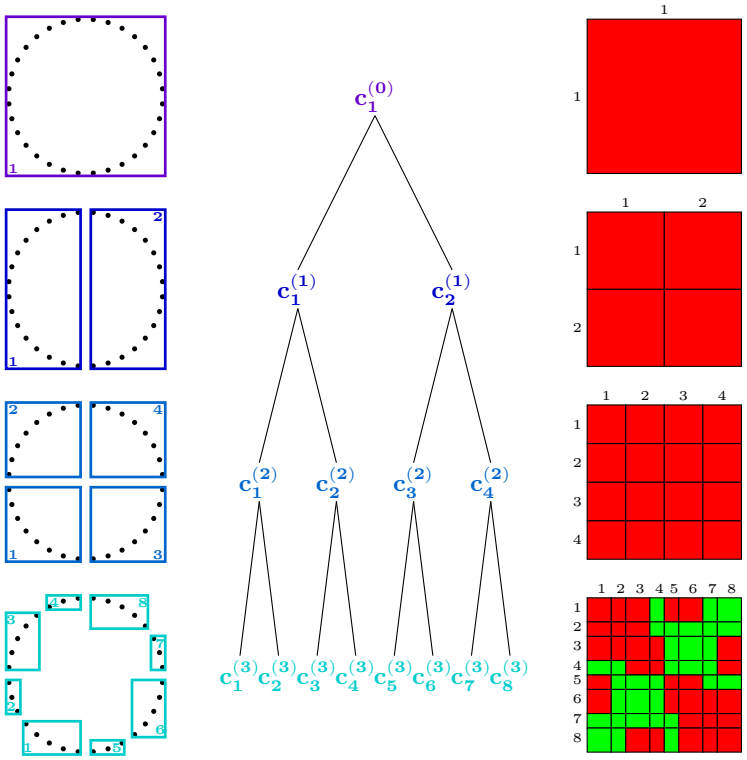


FIGURE 2.3.1 – Construction du *cluster tree* dans le cas d'un cercle discrétisé avec 32 DDLs avec $n_{max} = 5$ (3 niveaux de hiérarchie). Le nœud $c_j^{(i)}$ est le $j^{\text{ème}}$ ensemble d'indices du $i^{\text{ème}}$ niveau de hiérarchie. Les blocs verts (resp. rouges) sont les blocs admissibles (resp. non-admissibles) de la $\mathcal{H}$-matrice.

La $\mathcal{H}$-matrice sera stockée dans un *block cluster tree* qui correspond numériquement à un arbre quaternaire dans lequel les blocs matriciels seront enregistrés au niveau des feuilles. La figure 2.3.2 permet de visualiser la correspondance entre la notion d'arbre quaternaire et la hiérarchie d'une $\mathcal{H}$-matrice.

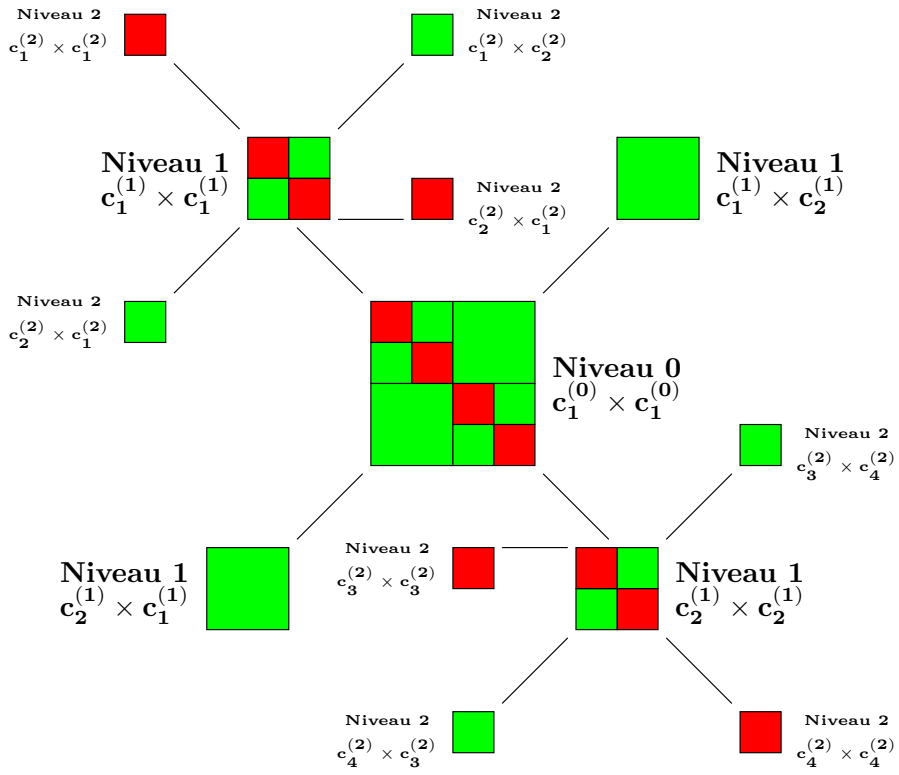


FIGURE 2.3.2 – Vue éclatée d'une $\mathcal{H}$-matrice à 2 niveaux de hiérarchie.

2.4 Construction d'une $\mathcal{H}$-matrice

2.4.1 Processus d'assemblage d'une $\mathcal{H}$-matrice

Une fois le *block cluster tree* construit, il ne reste plus qu'à assembler la $\mathcal{H}$-matrice. L'assemblage est un algorithme récursif qui part de la racine du *cluster tree*. Si le bloc a des descendants, alors on descend dans la hiérarchie, et ainsi de suite, jusqu'à ce que le nœud traité soit une feuille. On teste alors la η-admissibilité du bloc. Ainsi :

- Si le bloc est admissible, on construit le bloc en utilisant une méthode de compression, notée pour l'instant COMP, qui nous fournit une approximation à ε près du bloc à générer. Cette approximation est une r_k-matrice au format (2.1.4), à laquelle on applique la r_k-SVD à précision ε pour en réduire encore le coût mémoire. La nouvelle r_k-matrice ainsi obtenue est donc au format (2.1.23). Si son taux de compression est inférieur à 1, alors on conserve ce format. Sinon, la r_k-matrice est supprimée et remplacée par le bloc dense, construit grâce à la routine consacrée FULL_MATRIX.
- Si le bloc n'est pas admissible, on construit le bloc dense en utilisant la fonction FULL_MATRIX.

Notations. *Soit $A \in \mathbb{C}^{I \times J}$ une matrice à compresser, incarnée par le* block cluster tree *$T = T_{I \times J}$, dont la racine r est telle que $\widehat{r} = I \times J$. On note $\widetilde{A}_\varepsilon$, ou plus simplement $\widetilde{A}$, la représentation de A sous forme d'une $\mathcal{H}$-matrice compressée (à ε près). $b := (l, c)$ étant un nœud (ou bloc) de T tel que $\widehat{l}$ (resp. $\widehat{c}$) soit l'ensemble des indices des lignes (resp. des colonnes) de $A_{|b}$, on note par ailleurs $\widetilde{A}_{|b}$ ou $\widetilde{A}_{|l \times c}$ la représentation compressée de $A_{|b}$.*

L'η-admissibilité d'une feuille b est testée par une fonction notée ADMISSIBLE, qui renvoie un booléen. $b := (l, c) \in T$ possède alors trois paramètres matriciels, notés $U_{|b}$, $\Sigma_{|b}$ et $V_{|b}$ (ou $U_{|l \times c}$, $\Sigma_{|l \times c}$ et $V_{|l \times c}$), tels que :

- $A_{|b} \approx \widetilde{A}_{|b} = U_{|b} V_{|b}^H$ si $A_{|b}$ est admissible, après compression,
- $A_{|b} \approx \widetilde{A}_{|b} = U_{|b} \Sigma_{|b} V_{|b}^H$ si A est admissible, après r_k-SVD,
- $A_{|b} = \widetilde{A}_{|b} = U_{|b}$ si $A_{|b}$ n'est pas admissible ou si la compression suivie de la r_k-SVD donne un taux de compression supérieur à 1.

Dans la suite, nous fixerons la constante d'admissibilité $\eta = \frac{1}{2}$.

L'algorithme 2.4.1 décrit le processus d'assemblage de la $\mathcal{H}$-matrice $\widetilde{A}_{|b}$ représentée par le nœud $b := (l, c)$ dont l'étiquette $\widehat{b} := \widehat{l} \times \widehat{c}$ contient les indices de ses coefficients. $m_{|b}$ désigne l'ensemble des DDLs indexés par $\widehat{b}$.

Algorithme 2.4.1 Assemblage d'une $\mathcal{H}$-matrice $\widetilde{A}_{|b}$.

```
1:  procedure ASSEMBLY(b, m_|b, ε)
2:      if Card(sons(b)) = 0 then                          ▷ Si b est une feuille.
3:          if ADMISSIBLE(b, η) then      ▷ Si b est une feuille admissible, on compresse.
4:              call COMP(b, m_|b, ε)                      ▷ A_|b ≈ Ã_|b = U_|b V_|b^H.
5:              call RK_SVD(b, ε)                          ▷ A_|b ≈ Ã_|b = U_|b Σ_|b V_|b^H.
6:              if cr(Ã_|b) > 1 then  ▷ On n'applique la compression que si elle est efficace.
7:                  call NULLIFY(b)
8:                  call FULL_MATRIX(b, m_|b)              ▷ A_|b = Ã_|b = U_|b.
9:              end if
10:         else          ▷ Si b est une feuille non admissible, on construit la matrice dense.
11:             call FULL_MATRIX(b, m_|b)                  ▷ A_|b = Ã_|b = U_|b.
12:         end if
13:     else          ▷ Si b n'est pas une feuille, on applique la procédure à ses enfants.
14:         for b' ∈ sons(b) do
15:             call ASSEMBLY(b', m_|b', ε)                ▷ Récursion.
16:         end for
17:     end if
18: end procedure
```

On peut alors définir le taux de compression de la $\mathcal{H}$-matrice $\widetilde{A}$ grâce au taux de compression de chaque feuille b de T, tel que

$$\mathrm{cr}(\widetilde{A}) \overset{\text{déf.}}{=} \frac{1}{\mathrm{size}(A)} \sum_{b \in \mathcal{L}(T)} \mathrm{size}(A_{|b}) \, \mathrm{cr}(\widetilde{A}_{|b}). \tag{2.4.1}$$

Remarque 2.4.1 (Norme de Frobenius d'une $\mathcal{H}$-matrice)**.** *La norme de Frobenius (dont l'expression est rappelée en définition 1.4.2 pour une matrice dense et en remarque 2.1.5 pour une r_k-matrice au format (2.1.23)) présente elle aussi l'avantage d'être facilement calculable pour une matrice hiérarchique, car il suffit pour la calculer de connaître la norme de Frobenius de chaque feuille du* block cluster tree.

Considérons une $\mathcal{H}$-matrice $\widetilde{A}$ indexée par une partition de $I \times J$ représentée par le block cluster tree *$T := T_{I \times J}$. Alors la norme de Frobenius de $\widetilde{A}$ s'écrit*

$$\|\widetilde{A}\|_F = \sqrt{\sum_{b \in \mathcal{L}(T)} \|\widetilde{A}_{|b}\|_F^2}. \tag{2.4.2}$$

Dans la suite de cette partie, nous commencerons à valider les complexités algorithmiques théoriques des fonctions développées. Nous considérerons alors le cas particulier

de l'étude de la diffraction d'une onde plane monochromatique de fréquence $f = 0{,}6\,\text{GHz}$ ($\lambda \approx 0{,}5\,\text{m}$) dans les deux cas de référence décrits en partie 1.1.4. La méthode de compression utilisée est l'ACA décrite en partie 1.3.3.

Nous pouvons alors observer en figure 2.4.1 le temps d'exécution de l'algorithme 2.4.1 pour différentes valeurs de $\varepsilon = \varepsilon_{\text{ACA}}$ dans les cas du cylindre infini ($2D$) et de la sphère ($3D$). On vérifie en premier lieu que dans les deux cas la complexité algorithmique de l'assemblage est en $\mathcal{O}(n\log(n))$ quelle que soit la valeur de ε_{ACA}.

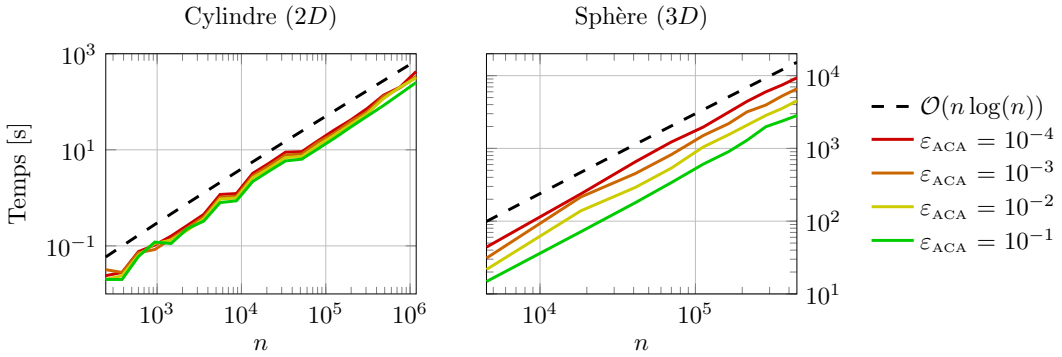


FIGURE 2.4.1 – Temps d'assemblage d'une $\mathcal{H}$-matrice en fonction du nombre de DDLs pour différentes valeurs de ε_{ACA}.

Nous constatons que la tendance est la même en $2D$ et en $3D$, avec une complexité en $\mathcal{O}(n\log(n))$ quelle que soit la valeur de ε_{ACA}. Il est important de noter que pour un nombre de DDLs donné, le temps d'assemblage est toutefois beaucoup plus important dans le cas de la sphère que dans celui du cylindre. Ceci peut s'expliquer d'une part par la plus grande complexité de la formulation EFIE par rapport à la formulation du potentiel de simple couche, mais également par des considérations géométriques. En effet, un DDL du maillage du cylindre est moins susceptible d'avoir beaucoup de proches voisins qu'un DDL appartenant au maillage de la sphère.

Cette hypothèse devrait se traduire par une constante de rareté plus grande, et donc un espace mémoire occupé plus grand pour le cas $3D$ que pour le cas $2D$, pour un nombre de DDLs donné. Nous pourrons le vérifier dans la prochaine partie.

2.4.2 *Coarsening* d'une $\mathcal{H}$-matrice

Nous avons défini en partie 2.3.3 la constante de rareté d'un *block cluster tree*, que l'on peut étendre aux $\mathcal{H}$-matrices. Ainsi, la constante de rareté d'une $\mathcal{H}$-matrice $\widetilde{A}$ est la constante de rareté du *block cluster tree* qui en définit la structure hiérarchique. Cette constante traduit la complexité structurelle de la $\mathcal{H}$-matrice, et il peut être utile de réduire cette constante en simplifiant la structure hiérarchique. C'est la problématique à laquelle répond le processus de *coarsening*.

Le *coarsening* d'une $\mathcal{H}$-matrice est un algorithme récursif qui consiste en l'agglomération de deux ou quatre blocs adjacents en un seul bloc [33]. Il s'agira en particulier de pouvoir créer à partir de deux r_k-matrices adjacentes une seule r_k-matrice, afin de conserver l'économie d'espace mémoire opérée lors de l'assemblage de la $\mathcal{H}$-matrice. Cette agglomération n'est appliquée que si elle réduit effectivement l'espace mémoire global utilisé.

Considérons un bloc matriciel $\widetilde{A} \in \mathbb{C}^{m\times n}$ composé de deux blocs adjacents $\widetilde{A}_1 \in \mathbb{C}^{m\times n_1}$ et $\widetilde{A}_2 \in \mathbb{C}^{m\times n_2}$ ($n = n_1 + n_2$) tels que $\widetilde{A} = \begin{pmatrix}\widetilde{A}_1 & \widetilde{A}_2\end{pmatrix}$. Quels que soient les formats de $\widetilde{A}_1$

et $\widetilde{A}_2$, on peut, soit par une SVD, soit par une r_k-SVD, obtenir ces blocs sous la forme

$$\widetilde{A}_1 = U_1 \Sigma_1 V_1^H \text{ et } \widetilde{A}_2 = U_2 \Sigma_2 V_2^H, \tag{2.4.3}$$

avec $U_i \in \mathbb{C}^{m \times k_i}$, $\Sigma_i \in \mathbb{R}^{k_i \times k_i}$ et $V_i \in \mathbb{C}^{n_i \times k_i}$, pour $i \in [\![1, 2]\!]$. On peut alors créer les deux matrices

$$U = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \text{ et } V = \begin{pmatrix} V_1 & \\ & V_2 \end{pmatrix}, \tag{2.4.4}$$

où $U \in \mathbb{C}^{m \times k}$ et $V \in \mathbb{C}^{n \times k}$ ($k = k_1 + k_2$). On effectue la décomposition QR de U : $U = QR$ où $Q \in \mathbb{C}^{m \times k}$ est une matrice orthogonale et $R \in \mathbb{C}^{k \times k}$ une matrice triangulaire supérieure pouvant se décomposer en deux blocs $R = \left| \begin{array}{cc} R_1 & R_2 \end{array} \right|$, avec $R_i \in \mathbb{C}^{k \times k_i}$, pour $i \in [\![1, 2]\!]$.

On construit alors $\Sigma = \begin{pmatrix} R_1 \Sigma_1 & R_2 \Sigma_2 \end{pmatrix} \in \mathbb{C}^{k \times k}$ et nous en effectuons la SVD suivie d'une troncature à ε près, telle que

$$\Sigma \approx U_\Sigma \Sigma_\Sigma V_\Sigma, \tag{2.4.5}$$

avec $U_\Sigma \in \mathbb{C}^{k \times k'}$, $\Sigma_\Sigma \in \mathbb{R}^{k' \times k'}$ et $V_\Sigma \in \mathbb{C}^{k \times k'}$. Alors, le bloc matriciel $\widetilde{A}$ peut être approché par la r_k-matrice $U_{\widetilde{A}} \Sigma_{\widetilde{A}} V_{\widetilde{A}}^H$ avec $U_{\widetilde{A}} = Q U_\Sigma \in \mathbb{C}^{m \times k'}$, $\Sigma_{\widetilde{A}} = \Sigma_\Sigma \in \mathbb{R}^{k' \times k'}$ et $V_{\widetilde{A}} = V V_\Sigma \in \mathbb{C}^{n \times k'}$.

Le rang de cette approximation k vérifie $k \leqslant k_1 + k_2$. Un algorithme similaire existe pour agglomérer directement quatre blocs matriciels adjacents [30], mais il est également possible d'utiliser deux fois de suite l'algorithme pour deux blocs. En effet, si on a

$$\widetilde{A} = \begin{pmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{pmatrix}, \tag{2.4.6}$$

il suffit d'agglomérer $\widetilde{A}_1 \approx \begin{pmatrix} \widetilde{A}_{11}^T & \widetilde{A}_{21}^T \end{pmatrix}$, $\widetilde{A}_2 \approx \begin{pmatrix} \widetilde{A}_{12}^T & \widetilde{A}_{22}^T \end{pmatrix}$, puis d'agglomérer $\begin{pmatrix} \widetilde{A}_1^T & \widetilde{A}_2^T \end{pmatrix} \approx \widetilde{A}$.

À l'échelle d'une $\mathcal{H}$-matrice, on applique l'agglomération aux blocs du *block cluster tree* dont les enfants sont des feuilles. Si le nœud a deux ou quatre enfants qui sont des feuilles, alors on agglomère les blocs correspondants selon une précision $\varepsilon = \varepsilon_{coars.}$, à l'aide d'une routine que nous appellerons AGGLOMERATE. Ainsi :

- si l'agglomération réduit effectivement l'espace mémoire, on conserve la r_k-matrice ainsi générée ; le nœud sur lequel on se situe devient donc une feuille, on supprime les enfants qu'il possédait ;
- dans le cas contraire, on remonte dans la hiérarchie, le bloc ne sera pas aggloméré.

Ce processus est résumé à l'algorithme 2.4.2, dans lequel la procédure DIAG renvoie un booléen selon que la $\mathcal{H}$-matrice $\widetilde{A}_{|b}$, incarnée par le nœud b, soit ou non située sur la diagonale de la hiérarchie globale.

On peut observer en figure 2.4.2 la différence de structure entre une $\mathcal{H}$-matrice agglomérée et une $\mathcal{H}$-matrice qui ne l'est pas.

On constate que la structure hiérarchique de la $\mathcal{H}$-matrice recompressée par *coarsening* est plus simple que celle de la $\mathcal{H}$-matrice originale, ce qui promet par la suite un parcours plus optimal d'une feuille de la structure arborescente à l'autre (et donc d'un bloc matriciel à l'autre) [30]. Cela se vérifie avec la constante de rareté, qui vaut 14 sur la $\mathcal{H}$-matrice originale contre 4 pour celle ayant bénéficié d'un *coarsening*. Par ailleurs, le *coarsening* fait passer le taux de compression de cette $\mathcal{H}$-matrice de 0,1617 à 0,1053, soit un espace de stockage qui décroît de 2,27 Mo à 1,47 Mo (contre 14,05 Mo si nous avions considéré la

Algorithme 2.4.2 *Coarsening* d'une $\mathcal{H}$-matrice $\widetilde{A}_{|b}$.

```
1: procedure COARSENING(b,ε_coars.)
2:     if Card(sons(b)) = 0 then                          ▷ Si b est une feuille.
3:         return
4:     else                                               ▷ Si b n'est pas une feuille.
5:         for b' ∈ sons(b) do
6:             call COARSENING(b', ε_coars.)              ▷ Récursion sur les enfants de b.
7:         end for
8:     end if
9:     if DIAG(b) then                                    ▷ Si b est un bloc diagonal.
10:        return
11:    end if
12:    if Card(sons(b)) ≠ 0 then                          ▷ Si b n'est pas une feuille.
13:        h = 0
14:        for b' ∈ sons(b) do
15:            if Card(sons(b'))) ≠ 0 then                ▷ On vérifie que les enfants de b sont des feuilles.
16:                h=1
17:            end if
18:        end for
19:        if h = 1 then                                  ▷ Si au moins l'un des enfants de b n'est pas une feuille.
20:            return
21:        else
22:            cr = cr(Ã_|b)
23:            call AGGLOMERATE(Ã_|b, ε_coars.)
24:            if cr < cr(Ã_|b) then                      ▷ Si l'agglomération augmente cr(Ã_|b).
25:                call NULLIFY(b)                        ▷ Suppression de la r_k-matrice créée par agglomération.
26:                return
27:            else                                       ▷ Si l'agglomération réduit cr(Ã_|b).
28:                for b' ∈ sons(b) do
29:                    call NULLIFY(b')                   ▷ Suppression des blocs des enfants de b.
30:                end for
31:            end if
32:        end if
33:    end if
34: end procedure
```

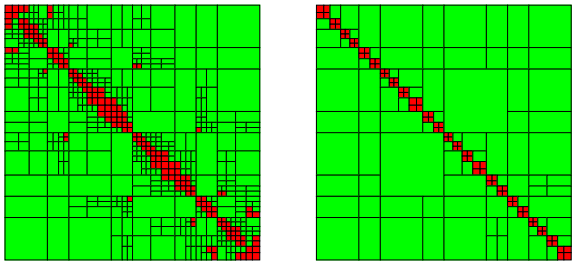


FIGURE 2.4.2 – Représentation hiérarchique de la matrice d'interactions entre les 937 DDLs d'un cylindre (2D) avant et après *coarsening* ($\varepsilon_{coars.} = 10^{-4}$).

matrice dense). Nous pourrons vérifier par la suite qu'en conséquence, les opérations arithmétiques sur les $\mathcal{H}$-matrices sont plus efficaces sur des structures matricielles simplifiées par *coarsening* (*cf.* partie 2.5).

On peut également observer le temps de *coarsening* selon la valeur de $\varepsilon_{coars.}$ en figure 2.4.3.

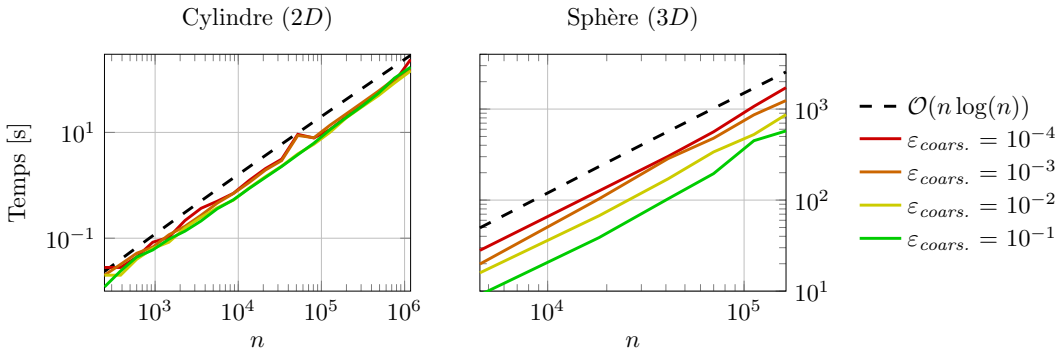


FIGURE 2.4.3 – Temps d'exécution du processus de *coarsening* sur une $\mathcal{H}$-matrice pour différentes valeurs de $\varepsilon_{coars.}$ ($\varepsilon_{\text{ACA}} = 10^{-4}$).

De même, observons l'espace mémoire occupé par une $\mathcal{H}$-matrice selon la valeur de $\varepsilon_{coars.}$ en figure 2.4.4.

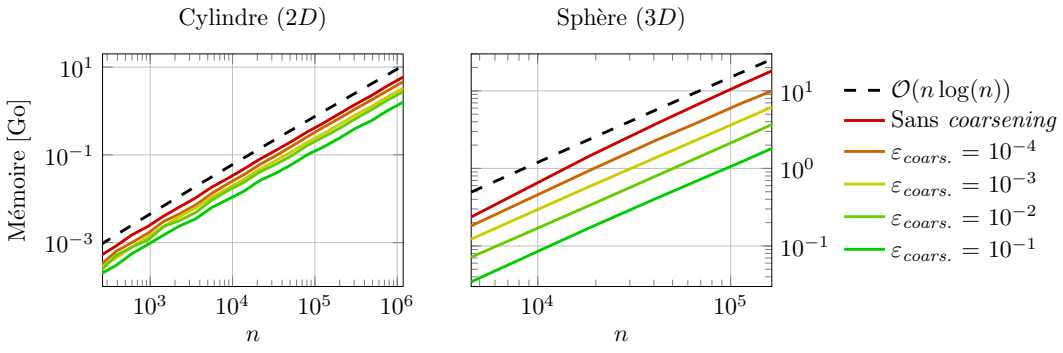


FIGURE 2.4.4 – Coût mémoire d'une $\mathcal{H}$-matrice après *coarsening* pour différentes valeurs de $\varepsilon_{coars.}$.

On constate que le temps de *coarsening* et l'espace mémoire occupé par la $\mathcal{H}$-matrice sont proportionnels à $n\log(n)$ quelle que soit la valeur de $\varepsilon_{coars.}$. Le coût mémoire du cas $3D$ est plus important que pour le cas du cylindre, ce qui se traduit également sur le temps de *coarsening*, quelle que soit la valeur de $\varepsilon_{coars.}$.

Le tableau 2.4.1 présente le rapport $\frac{\text{temps d'assemblage}}{\text{temps de } \textit{coarsening}}$ pour différentes valeurs de $\varepsilon_{\textit{coars.}}$.

	$\varepsilon_{\textit{coars.}}$			
n	10^{-4}	10^{-3}	10^{-2}	10^{-1}
4521	0,637	0,451	0,361	0,207
18 294	0,534	0,437	0,285	0,164
41 046	0,462	0,437	0,262	0,158
69 930	0,460	0,391	0,276	0,160
112 206	0,546	0,440	0,268	0,229
163 500	0,547	0,395	0,277	0,183

TABLEAU 2.4.1 – Proportion du temps de *coarsening* par rapport au temps d'assemblage par ACA ($\varepsilon_{\text{ACA}} = 10^{-4}$) pour différentes valeurs de $\varepsilon_{\textit{coars.}}$ dans le cas d'une sphère ($3D$).

Ce tableau nous permet de constater que le temps de *coarsening* est toujours inférieur au temps d'assemblage. En particulier, plus $\varepsilon_{\textit{coars.}}$ est fin, plus le temps de *coarsening* est long. Cependant cette proportion est relativement constante en fonction du nombre de DDLs pour une précision $\varepsilon_{\textit{coars.}}$ donnée.

Vérifions enfin que le *coarsening*, qui est une compression supplémentaire, ne nuit pas à la précision de la $\mathcal{H}$-matrice. Pour cela, assemblons la $\mathcal{H}$-matrice $\widetilde{Z}$ sans *coarsening* et comparons la par erreur relative en norme de Frobenius avec sa version recompressée par *coarsening*, pour plusieurs valeurs de $\varepsilon_{\textit{coars.}}$ (*cf.* figure 2.4.5).

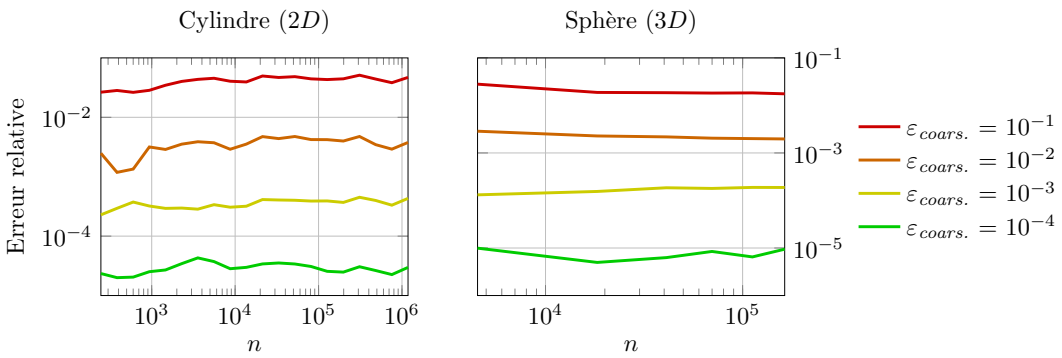


FIGURE 2.4.5 – Erreur relative provoquée par le processus de *coarsening* sur une $\mathcal{H}$-matrice pour différentes valeurs de $\varepsilon_{\textit{coars.}}$.

On constate que l'erreur mesurée est constante en fonction du nombre de DDLs, tout en restant de l'ordre de la valeur $\varepsilon_{\textit{coars.}}$, ce qui valide le processus de *coarsening* en terme de précision.

2.5 Arithmétique des $\mathcal{H}$-matrices

Une fois que nous sommes capables d'assembler des $\mathcal{H}$-matrices, il est possible de développer une arithmétique qui soit adaptée à ce format. Cette arithmétique se compose de plusieurs opérations qui prennent en compte la structure arborescente des $\mathcal{H}$-matrices, et implique donc le développement d'algorithmes récursifs [30].

Une $\mathcal{H}$-matrice comporte à la fois des blocs denses stockés comme des matrices denses, et des blocs compressés stockés comme des r_k-matrices. Ainsi beaucoup d'opérations de cette arithmétique impliquent également des approximations à ε près, ε étant généralement la précision préalablement utilisée pour la compression.

Pour décrire ces opérations, nous nous appuierons sur des $\mathcal{H}$-matrices de structure hiérarchique simple à un ou deux niveaux de hiérarchie. Tous les algorithmes sont ensuite généralisables à des structures hiérarchiques plus complexes et comportant plus de niveaux.

Nous vérifierons que pour chaque opération, la complexité algorithmique est améliorée, et que l'opération de *coarsening* permet de réduire le temps consacré à ces opérations.

2.5.1 Somme arrondie

Considérons deux $\mathcal{H}$-matrices $\widetilde{A}$ et $\widetilde{B}$ de même structure hiérarchique. Nous souhaitons effectuer la somme $\widetilde{C} \coloneqq \widetilde{A} \oplus \widetilde{B}$ comme schématisée en figure 2.5.1 sur laquelle les blocs verts (resp. rouges) représentent des blocs admissibles (resp. non admissibles).

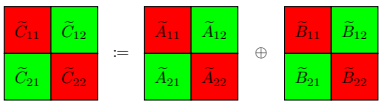


FIGURE 2.5.1 – Somme de deux $\mathcal{H}$-matrices à un niveau de hiérarchie.

Cette addition se décompose en une succession de quatre étapes :

$$\widetilde{C}_{11} \coloneqq \widetilde{A}_{11} + \widetilde{B}_{11}, \tag{2.5.1}$$
$$\widetilde{C}_{12} \coloneqq \widetilde{A}_{12} \oplus \widetilde{B}_{12}, \tag{2.5.2}$$
$$\widetilde{C}_{21} \coloneqq \widetilde{A}_{21} \oplus \widetilde{B}_{21}, \tag{2.5.3}$$
$$\widetilde{C}_{22} \coloneqq \widetilde{A}_{22} + \widetilde{B}_{22}. \tag{2.5.4}$$

Les équations (2.5.1) et (2.5.4) sont exactes tandis que les équations (2.5.2) et (2.5.3) utilisent l'addition arrondie de deux r_k-matrices, décrite en section 2.1.4. De ce fait l'opération globale est approchée à une précision ε près.

On peut généraliser la somme arrondie de deux $\mathcal{H}$-matrices $\widetilde{A}_{|b_A}$ et $\widetilde{B}_{|b_B}$ de même structure hiérarchique, par l'algorithme 2.5.1, RKMAT_ADD désignant la somme arrondie de deux r_k-matrices. La $\mathcal{H}$-matrice résultante $\widetilde{C}_{|b_C}$ sera de même structure hiérarchique que $\widetilde{A}_{|b_A}$ et $\widetilde{B}_{|b_B}$.

On a alors le théorème suivant (dont la démonstration est disponible dans [30]).

Théorème 2.5.1. *Le nombre d'opérations nécessaires à réaliser la somme arrondie de $\widetilde{A}$ et $\widetilde{B}$ est du même ordre que*

$$c_{sp}k^2(\operatorname{depth}(T_I)\operatorname{Card}(I) + \operatorname{depth}(T_J)\operatorname{Card}(J)) + c_{sp}k^3 \min(\operatorname{Card}(I), \operatorname{Card}(J)). \quad (2.5.5)$$

Le temps de réalisation de cette opération est donc plus faible sur des $\mathcal{H}$-matrices ayant subit une opération de *coarsening* au préalable, la constante de rareté étant alors plus faible. Pour le vérifier, nous avons assemblé une $\mathcal{H}$-matrice $\widetilde{Z}$ à précision $\varepsilon_{\text{ACA}} = 10^{-4}$ et nous y avons appliqué le *coarsening* pour différentes valeurs de $\varepsilon_{coars.}$ avant d'effectuer l'opération $2\widetilde{Z} = \widetilde{Z} \oplus \widetilde{Z}$. Le temps d'exécution de cette somme arrondie est visible en figure 2.5.2.

Algorithme 2.5.1 Somme arrondie de deux $\mathcal{H}$-matrices $\widetilde{A}_{|b_A}$ et $\widetilde{B}_{|b_B}$.

1: **procedure** HMAT_ADD(b_A,b_B,ε,b_C)
2: **if** Card(sons(b_A)) $= 0$ **and** Card(sons(b_B)) $= 0$ **then** ▷ Si b_A et b_B sont des feuilles.
3: **if** ADMISSIBLE(b_A, η) **and** ADMISSIBLE(b_B, η) **then** ▷ Si b_A et b_B sont admissibles.
4: **call** RKMAT_ADD($\widetilde{A}_{|b_A}, \widetilde{B}_{|b_B}, \varepsilon, \widetilde{C}_{|b_C}$) ▷ Somme arrondie de r_k-matrices (*cf.* partie 2.1.4).
5: **else**
6: $\widetilde{C}_{|b_C} := \widetilde{A}_{|b_A} + \widetilde{B}_{|b_B}$ ▷ Addition classique de deux matrices.
7: **end if**
8: **else** ▷ Si b_A ou b_B n'est pas une feuille.
9: **for** $b'_A \in$ sons(b_A), $b'_C \in$ sons(b_B), $b'_C \in$ sons(b_C) **do**
10: **if** $\widehat{b}'_A = \widehat{b}'_B = \widehat{b}'_C$ **then**
11: **call** HMAT_ADD($b'_A, b'_B, \varepsilon, b'_C$) ▷ Récursion.
12: **end if**
13: **end for**
14: **end if**
15: **end procedure**

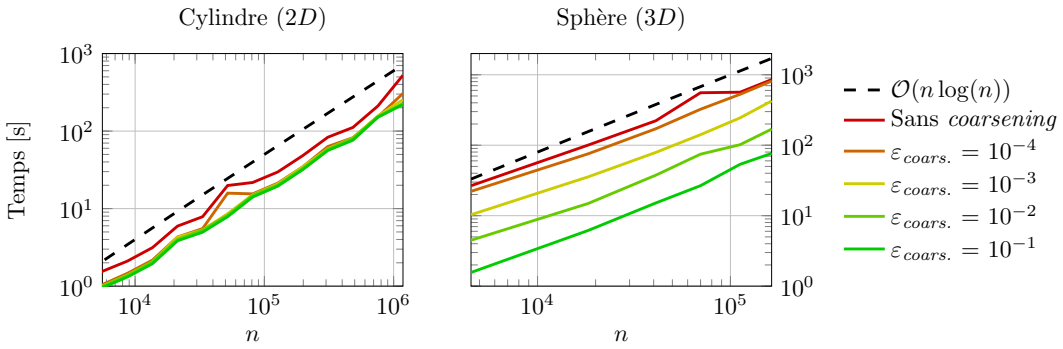


FIGURE 2.5.2 – Temps d'exécution de la somme arrondie de deux $\mathcal{H}$-matrices pour différentes valeurs de $\varepsilon_{coars.}$.

2.5.2 Produit $\mathcal{H}$-matrice-vecteur

Considérons une $\mathcal{H}$-matrice $\widetilde{A}$ constituée de quatre blocs (un niveau de hiérarchie) et un vecteur $\boldsymbol{x}$. Nous souhaitons réaliser l'opération $\widetilde{A}\boldsymbol{x} = \boldsymbol{y}$ comme schématisée en figure 2.5.3.

Cette opération est équivalente aux deux étapes

$$\boldsymbol{y}_1 = \widetilde{A}_{11}\boldsymbol{x}_1 + \widetilde{A}_{12}\boldsymbol{x}_2, \tag{2.5.6}$$

$$\boldsymbol{y}_2 = \widetilde{A}_{21}\boldsymbol{x}_1 + \widetilde{A}_{22}\boldsymbol{x}_2. \tag{2.5.7}$$

Contrairement aux autres opérations arithmétiques liées aux $\mathcal{H}$-matrices, le produit $\mathcal{H}$-matrice-vecteur n'induit pas d'approximation supplémentaire. En effet, le produit r_k-matrice-vecteur tel que décrit au paragraphe 2.1.4 est exact.

Pour réaliser cette opération à l'échelle d'une $\mathcal{H}$-matrice, il suffit donc de réaliser ces opérations au niveau de chaque feuille du *block cluster tree* et de sommer le vecteur obtenu aux coefficients correspondants du vecteur résultant. Ce processus est décrit à l'algorithme 2.5.2, RKMAT_VECT désignant le produit r_k-matrice-vecteur.

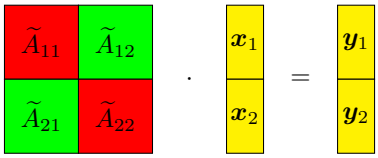


FIGURE 2.5.3 – Produit $\mathcal{H}$-matrice-vecteur pour une $\mathcal{H}$-matrice à un niveau de hiérarchie.

Algorithme 2.5.2 Produit d'une $\mathcal{H}$-matrice $\widetilde{A}_{|b}$ par un vecteur $\boldsymbol{x}$.

```
1:  procedure HMAT_VECT(Ã_|b, x, y)
2:      if Card(sons(b)) = 0 then                          ▷ Si b est une feuille.
3:          if cr(Ã_|b) < 1 then                  ▷ Si Ã_|b est représentée par une r_k-matrice.
4:              call RKMAT_VECT(Ã_|b, x, y)  ▷ Produit r_k-matrice-vecteur (cf. partie 2.1.4).
5:          else
6:              y ≔ Ã_|b x                              ▷ Produit matrice-vecteur classique.
7:          end if
8:      else                                                ▷ Si b n'est pas une feuille.
9:          y ≔ 0
10:         for b' ∈ sons(b) do
11:             call HMAT_VECT(Ã_|b', x_|b', y')                                ▷ Récursion.
12:             y_|b' ← y_|b' + y'                          ▷ Addition au vecteur résultant.
13:         end for
14:     end if
15: end procedure
```

La constante de rareté permet alors de majorer le nombre d'opérations du produit $\mathcal{H}$-matrice-vecteur selon le théorème suivant.

Théorème 2.5.2. *Soit n_{mv} le nombre d'opérations nécessaires à la réalisation du produit matrice-vecteur*

$$\widetilde{A}\boldsymbol{x}_\Gamma = \boldsymbol{b},$$

avec $A \in \mathbb{C}^{I\times J}$ incarnée par le block cluster tree *$T_{I\times J}$, et $\boldsymbol{x}_\Gamma \in \mathbb{C}^J$. On a alors la majoration*

$$n_{mv} \leqslant 2c_{sp}\max(k, n_{max})(\operatorname{depth}(T_I)\operatorname{Card}(I) + \operatorname{depth}(T_J)\operatorname{Card}(J)). \tag{2.5.8}$$

En particulier, si T_I et T_J sont équilibrés, alors

$$n_{mv} \sim \max(k, n_{max})(\operatorname{Card}(I)\log(\operatorname{Card}(I)) + \operatorname{Card}(J)\log(\operatorname{Card}(J))). \tag{2.5.9}$$

Ici, $I = J$, donc le temps de réalisation du produit $\mathcal{H}$-matrice-vecteur est proportionnel à $n\log(n)$ et il est plus faible si l'opération est réalisée sur une matrice ayant subit une opération de *coarsening* au préalable, et donc ayant une constante de rareté plus faible, comme nous pouvons le vérifier en figure 2.5.4.

En particulier, le solveur itératif GMRES (généralisation de la méthode de Minimisation du Résidu, de l'anglais *Generalized Minimal Residual method*) sera rapidement adaptable au format $\mathcal{H}$-matrice grâce à cette opération.

2.5.3 Multiplication formatée

Soient $\widetilde{M}$, $\widetilde{A}$ et $\widetilde{B}$ trois $\mathcal{H}$-matrices. La multiplication formatée, notée $\odot$, est une opération entre trois $\mathcal{H}$-matrices définie par

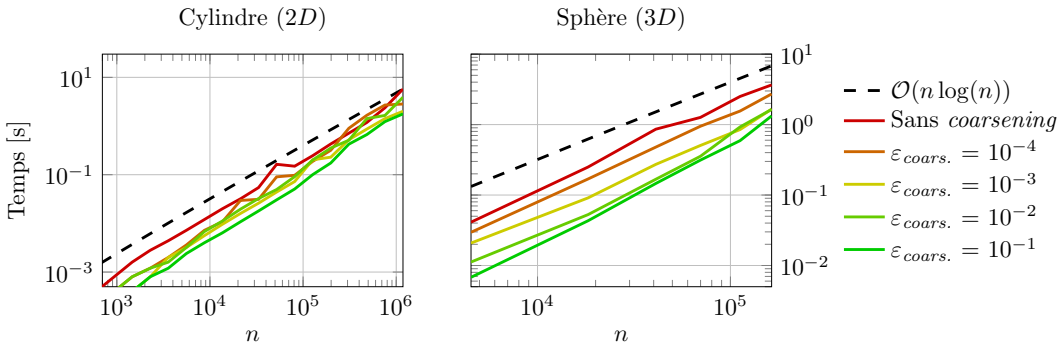
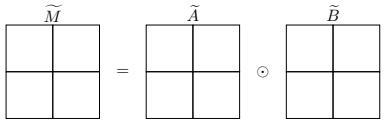


FIGURE 2.5.4 – Temps d'exécution du produit $\mathcal{H}$-matrice-vecteur pour différentes valeurs de $\varepsilon_{coars.}$.

$$\widetilde{M} \leftarrow \widetilde{M} \oplus \widetilde{A} \odot \widetilde{B}. \tag{2.5.10}$$

Cette opération correspond à la fois à une multiplication $\widetilde{C} \coloneqq \widetilde{A} \odot \widetilde{B}$ arrondie par troncature, puis une somme $\widetilde{M} \leftarrow \widetilde{M} \oplus \widetilde{C}$, elle aussi arrondie. Sa réalisation dépend de la structure des matrices impliquées. Il existe en particulier trois cas de figure que nous allons détailler.

Cas 1 : $\widetilde{M}$, $\widetilde{A}$ et $\widetilde{B}$ sont subdivisées.



Dans ce cas, la multiplication $\odot$ n'aura véritablement lieu qu'au niveau inférieur de la hiérarchie. En effet, la multiplication

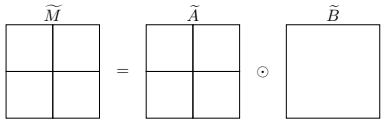
$$\begin{bmatrix} \widetilde{M}_{11} & \widetilde{M}_{12} \\ \widetilde{M}_{21} & \widetilde{M}_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix} \odot \begin{bmatrix} \widetilde{B}_{11} & \widetilde{B}_{12} \\ \widetilde{B}_{21} & \widetilde{B}_{22} \end{bmatrix} \tag{2.5.11}$$

répond à la formule de multiplication de deux matrices par blocs qui s'écrit pour $1 \leqslant i, j \leqslant 2$

$$\widetilde{M}_{ij} = \sum_{k=1}^{2} \widetilde{A}_{ik} \odot \widetilde{B}_{kj}. \tag{2.5.12}$$

Il s'agit alors de réaliser chacune des multiplications $\widetilde{A}_{ik} \odot \widetilde{B}_{kj}$ aux niveaux inférieurs de la hiérarchie, ce que permettront les deux cas suivants.

Cas 2 : $\widetilde{M}$ est subdivisée mais $\widetilde{A}$ et/ou $\widetilde{B}$ ne l'est pas.



Si ni $\widetilde{A}$ ni $\widetilde{B}$ n'est subdivisée, alors l'opération consiste en une multiplication classique si les blocs concernés sont denses, ou en une multiplication de r_k-matrices (*cf.* partie 2.1.4) si l'on a affaire à des blocs compressés. Le résultat est alors ajouté à $\widetilde{M}$ bloc par bloc par addition arrondie.

En revanche, si $\widetilde{A}$ est subdivisée et $\widetilde{B}$ ne l'est pas, la multiplication $\widetilde{C} \coloneqq \widetilde{A} \odot \widetilde{B}$ est réalisée par le biais d'une succession de produits $\mathcal{H}$-matrice-vecteur. En particulier :

- si $\widetilde{B}$ est un bloc non-admissible, alors $\widetilde{C}$ sera un bloc dense et

$$\widetilde{C}(:, i) \coloneqq \widetilde{A}\widetilde{M}(:, i). \tag{2.5.13}$$

  On ajoute alors $\widetilde{C}$ à $\widetilde{M}$ bloc par bloc par addition arrondie.

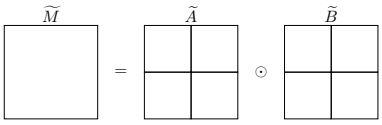
- si $\widetilde{B}$ est un bloc admissible, alors $\widetilde{C}$ sera une r_k-matrice et

$$U(:, i) \coloneqq \widetilde{A}U_B(:, i), \tag{2.5.14}$$

  puis en posant $\Sigma = \Sigma_B$ et $V = V_B$, et en effectuant une r_k-SVD sur $\widetilde{C} = U\Sigma V^H$, on obtient $\widetilde{C} \approx U_C S_C V_C^H$ avec les propriétés appropriées pour U_C, Σ_C et V_C. Comme dans les cas précédents, on ajoute alors $\widetilde{C}$ à $\widetilde{M}$ bloc par bloc par addition arrondie.

Dans le cas où $\widetilde{B}$ est subdivisée et $\widetilde{A}$ ne l'est pas, on applique les mêmes étapes en travaillant sur la transposée de chaque terme.

Cas 3 : $\widetilde{M}$ n'est pas subdivisée.



Il faut alors prendre en compte le format de $\widetilde{M}$, le travail n'étant pas le même si le bloc $\widetilde{M}$ est une matrice dense ou une r_k-matrice.

Dans ce dernier cas, nous devrons développer un algorithme qui effectuera l'opération de multiplication et de somme en accord avec le format de $\widetilde{M}$.

En particulier, si $\widetilde{M}$ est une r_k-matrice, nous utiliserons un nouvel algorithme qui prend en considération plusieurs cas de figure.

Cas 3.a Si $\widetilde{A}$ et $\widetilde{B}$ ne sont pas hiérarchiques, alors on effectue simplement la multiplication des matrices denses ou r_k-matrices concernées.

Cas 3.b Si $\widetilde{A}$ et/ou $\widetilde{B}$ n'est pas hiérarchique, on se ramène au **Cas 2** traité précédemment et on convertit le résultat obtenu en r_k-matrice.

Cas 3.c Si $\widetilde{A}$ et $\widetilde{B}$ sont hiérarchiques, alors on applique la récursion pour atteindre les feuilles de $\widetilde{A}$ ou celles de $\widetilde{B}$ et se ramener à l'un des cas précédemment traités. On obtient alors des blocs sous forme de r_k-matrices qu'il faudra ensuite agglomérer pour conserver la structure initiale de $\widetilde{M}$.

À l'échelle d'une $\mathcal{H}$-matrice complète, on peut résumer ce processus comme indiqué à l'algorithme 2.5.3, dans lequel $b_M \coloneqq (l_M, c_M)$ (resp. $b_A \coloneqq (l_A, c_A)$, $b_B \coloneqq (l_B, c_B)$) désigne le nœud du *block cluster tree* dont l'étiquette contient les indices des coefficients de la $\mathcal{H}$-matrice $\widetilde{M}$ (resp. $\widetilde{A}$, $\widetilde{B}$). La routine RKMAT_MUL_ADD désigne la méthode à utiliser dans le **Cas 3**, détaillée à l'algorithme 2.5.4.

Tout comme les deux opérations précédemment traitées, le temps de réalisation de la multiplication formatée est plus faible si l'opération est réalisée sur une matrice ayant subit une opération de *coarsening* au préalable. Pour le vérifier, nous avons assemblé la matrice $\widetilde{Z}$ que nous avons ensuite recompressée par *coarsening* à différentes valeurs de $\varepsilon_{coars.}$, puis nous avons réalisé l'opération $\widetilde{Z} \leftarrow \widetilde{Z} \oplus \widetilde{Z} \odot \widetilde{Z}$ (*cf.* figure 2.5.5).

Algorithme 2.5.3 Multiplication formatée de trois $\mathcal{H}$-matrices $\widetilde{M}_{|b_M}$, $\widetilde{A}_{|b_A}$ et $\widetilde{B}_{|b_B}$.

procedure HMAT_MUL_ADD($\widetilde{M}_{|b_M}$,$\widetilde{A}_{|b_A}$,$\widetilde{B}_{|b_B}$,ε)
  if Card(sons(b_M)) $\neq 0$ **and** Card(sons(b_A)) $\neq 0$ **and** Card(sons(b_B)) $\neq 0$ **then** ▷ **Cas 1**.
    for $l'_M \in \text{sons}(l_M)$, $c'_M \in \text{sons}(c_M)$, $c'_A \in \text{sons}(c_A)$ **do**
      call HMAT_MUL_ADD($\widetilde{M}_{|l'_M \times c'_M}$, $\widetilde{A}_{|l'_M \times c'_A}$, $\widetilde{B}_{|c'_A \times c'_M}$, ε) ▷ Récursion.
    end for
  else if Card(sons(b_M)) $\neq 0$ **then** ▷ **Cas 2**.
    $\widetilde{C} \coloneqq \widetilde{A}\widetilde{B}$ ▷ À adapter selon les cas.
    call HMAT_ADD($b_M, b_C, \varepsilon, b_M$)
  else ▷ **Cas 3**.
    call RKMAT_MUL_ADD($\widetilde{C}, \widetilde{A}, \widetilde{B}, \varepsilon$)
    call HMAT_ADD($b_M, b_C, \varepsilon, b_M$)
  end if
end procedure

Algorithme 2.5.4 Multiplication formatée de la r_k-matrice $\widetilde{M}$ et des $\mathcal{H}$-matrices $\widetilde{A}_{|b_A}$ et $\widetilde{B}_{|b_B}$.

procedure RKMAT_MUL_ADD($\widetilde{M}$,$\widetilde{A}_{|b_A}$,$\widetilde{B}_{|b_B}$,ε)
  if Card(sons(b_A)) $\neq 0$ **and** Card(sons(b_B)) $\neq 0$ **then** ▷ **Cas 3.c**.
    $\widetilde{C} \coloneqq 0$
    for $l'_A \in \text{sons}(l_A)$, $c'_B \in \text{sons}(c_B)$, $c'_A \in \text{sons}(c_A)$ **do**
      call RKMAT_MUL_ADD($\widetilde{C}_{|l'_A \times c'_B}$, $\widetilde{A}_{|l'_A \times c'_A}$, $\widetilde{B}_{|c'_A \times c'_B}$, ε) ▷ Récursion.
    end for
    call AGGLOMERATE($\widetilde{C}, \varepsilon$) ▷ Agglomération du résultat.
  else if Card(sons(b_A)) $\neq 0$ or Card(sons(b_B)) $\neq 0$ **then** ▷ **Cas 3.b**.
    $\widetilde{C} \coloneqq \widetilde{A}\widetilde{B}$ ▷ À adapter selon les cas.
  else ▷ **Cas 3.a**.
    $\widetilde{C} \coloneqq \widetilde{A}\widetilde{B}$ ▷ À adapter selon les cas.
  end if
  call RKMAT_ADD($\widetilde{M}, \widetilde{C}, \varepsilon, \widetilde{M}$) ▷ Somme arrondie de r_k-matrices (*cf.* partie 2.1.4).
end procedure

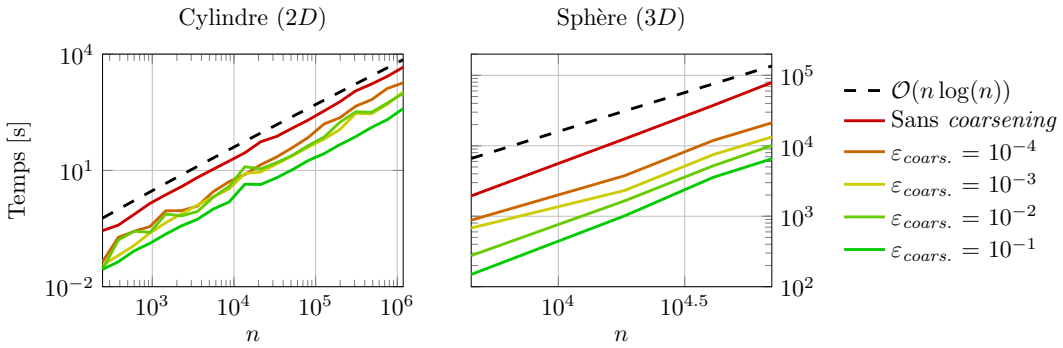


FIGURE 2.5.5 – Temps d'exécution de la multiplication formatée pour différentes valeurs de $\varepsilon_{coars.}$.

Remarque 2.5.1. *On peut également définir l'opération*

$$\widetilde{M} \leftarrow \widetilde{M} \ominus \widetilde{A} \odot \widetilde{B}, \tag{2.5.15}$$

appelée multiplication formatée par soustraction*, et qui correspond à la succession d'opérations*

$$\widetilde{C} \coloneqq \widetilde{A} \odot \widetilde{B}, \tag{2.5.16}$$

$$\widetilde{C} \leftarrow -\widetilde{C}, \tag{2.5.17}$$

$$\widetilde{M} \leftarrow \widetilde{M} \oplus \widetilde{C}. \tag{2.5.18}$$

Cette opération sera elle aussi appelée multiplication formatée*, et la fonction correspondante sera notée* HMAT_MUL_SUB.

2.5.4 Décomposition $\mathcal{H}$-LU

La décomposition $\mathcal{H}$-LU est une décomposition LU approchée d'une $\mathcal{H}$-matrice, suivant une précision ε_{LU} donnée [34, 35]. Elle permet d'adapter au format $\mathcal{H}$-matrice la décomposition LU décrite en définition A.1 (*cf.* annexe A).

Soit une $\mathcal{H}$-matrice $\widetilde{A}$ avec une structure donnée. La décomposition $\mathcal{H}$-LU de $\widetilde{A}$ consiste à construire les $\mathcal{H}$-matrices $\widetilde{L}$ et $\widetilde{U}$ ayant la même structure que $\widetilde{A}$ et vérifiant

- $\widetilde{A} \approx \widetilde{L}\widetilde{U}$,
- $\widetilde{L}$ est triangulaire inférieure,
- $\widetilde{U}$ est triangulaire supérieure.

On considère une $\mathcal{H}$-matrice $\widetilde{A}$ composée de quatre blocs, comme schématisée en figure 2.5.6.

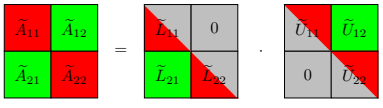


FIGURE 2.5.6 – Décomposition $\mathcal{H}$-LU d'une $\mathcal{H}$-matrice à un niveau de hiérarchie.

$\widetilde{L}$ et $\widetilde{U}$ sont alors également composées de quatre blocs, comme suit :

$$\begin{bmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{L}_{11} & 0 \\ \widetilde{L}_{21} & \widetilde{L}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{U}_{11} & \widetilde{U}_{12} \\ 0 & \widetilde{U}_{22} \end{bmatrix}, \tag{2.5.19}$$

avec $\widetilde{L}_{ii}$ triangulaire inférieure et $\widetilde{U}_{ii}$ triangulaire supérieure pour $i \in [\![1, 2]\!]$.

On a alors à résoudre les équations

$$\widetilde{A}_{11} = \widetilde{L}_{11}\widetilde{U}_{11}, \tag{2.5.20}$$
$$\widetilde{A}_{12} = \widetilde{L}_{11}\widetilde{U}_{12}, \tag{2.5.21}$$
$$\widetilde{A}_{21} = \widetilde{L}_{21}\widetilde{U}_{11}, \tag{2.5.22}$$
$$\widetilde{A}_{22} = \widetilde{L}_{21}\widetilde{U}_{12} + \widetilde{L}_{22}\widetilde{U}_{22}. \tag{2.5.23}$$

Ainsi :

- L'équation (2.5.20) consiste en une décomposition LU classique de $\widetilde{A}_{11}$ si celle-ci n'est pas hiérarchique, et permet de déterminer $\widetilde{L}_{11}$ et $\widetilde{U}_{11}$.
- Les équations (2.5.21) et (2.5.22) consistent à résoudre récursivement un système triangulaire grâce à une méthode de *descente récursive*, détaillée dans la suite de cette partie.
- Enfin, l'équation (2.5.23) nécessite d'utiliser la multiplication formatée par soustraction $\widetilde{A}_{22} \leftarrow \widetilde{A}_{22} \ominus \widetilde{L}_{22} \odot \widetilde{U}_{22}$ puis d'effectuer une décomposition $\widetilde{L}\widetilde{U}$ classique sur le résultat obtenu.

La décomposition $\mathcal{H}$-LU suit donc les étapes décrites dans l'algorithme 2.5.5, dans lequel HMAT_LT_SOLVE et HMAT_UT_SOLVE désignent les algorithmes de descente, HMAT_MUL_SUB la multiplication formatée par soustraction et LU la décomposition LU classique.

Remarque 2.5.2. *La routine* LU *utilise la routine* LAPACK *consacrée (*-GETRF*), qui réalise une décomposition PLU. La décomposition $\mathcal{H}$-LU est donc, en toute rigueur, une décomposition $\mathcal{H}$-PLU (*cf. *annexe A*).*

Algorithme 2.5.5 Décomposition $\mathcal{H}$-LU d'une $\mathcal{H}$-matrice $\widetilde{A}_{|b}$.

```
1:  procedure HMAT_LU(Ã_|b, L̃, Ũ, ε_LU)
2:      if Card(sons(b)) = 4 then                          ▷ Si b a quatre enfants.
3:          call HMAT_LU(Ã_11, L̃_11, Ũ_11, ε_LU)           ▷ Récursion.
4:          call HMAT_LT_SOLVE(Ã_12, L̃_11, Ũ_12, ε_LU)     ▷ Descente à droite.
5:          call HMAT_UT_SOLVE(Ã_21, L̃_21, Ũ_11, ε_LU)     ▷ Descente à gauche.
6:          call HMAT_MUL_SUB(Ã_22, L̃_21, Ũ_12, ε_LU)      ▷ Multiplication formatée
    (soustraction).
7:          call HMAT_LU(Ã_22, L̃_22, Ũ_22, ε_LU)           ▷ Récursion.
8:      else
9:          call LU(Ã, L̃, Ũ)
10:     end if
11: end procedure
```

On constate en lignes 3 et 7 un appel récursif de la routine HMAT_LU. La fonction HMAT_LT_SOLVE appelée en ligne 4 est l'algorithme de descente défini en algorithme 2.5.6.

Notons pour cette étape, $\widetilde{A}'$ (resp. $\widetilde{L}'$, $\widetilde{U}'$) le bloc $\widetilde{A}_{12}$ (resp. $\widetilde{L}_{11}$, $\widetilde{U}_{12}$) de notre matrice $\widetilde{A}$ (resp. $\widetilde{L}$, $\widetilde{U}$) globale. On a, dans le cas où ces trois blocs sont subdivisés :

$$\begin{bmatrix} \widetilde{A}'_{11} & \widetilde{A}'_{12} \\ \widetilde{A}'_{21} & \widetilde{A}'_{22} \end{bmatrix} = \begin{bmatrix} \widetilde{L}'_{11} & 0 \\ \widetilde{L}'_{21} & \widetilde{L}'_{22} \end{bmatrix} \begin{bmatrix} \widetilde{U}'_{11} & \widetilde{U}'_{12} \\ \widetilde{U}'_{21} & \widetilde{U}'_{22} \end{bmatrix}, \tag{2.5.24}$$

avec $\widetilde{L}_{11}$ triangulaire inférieure déterminée en ligne 3 de l'algorithme 2.5.5 et $\widetilde{U}_{12}$ matrice inconnue à déterminer.

Cette résolution se déroule suivant les étapes

$$\widetilde{A}'_{11} = \widetilde{L}'_{11}\widetilde{U}'_{11}, \tag{2.5.25}$$
$$\widetilde{A}'_{12} = \widetilde{L}'_{11}\widetilde{U}'_{12}, \tag{2.5.26}$$
$$\widetilde{A}'_{21} \leftarrow \widetilde{A}'_{21} \ominus \widetilde{L}'_{21} \odot \widetilde{U}'_{11} \text{ puis } \widetilde{A}'_{21} = \widetilde{L}'_{22}\widetilde{U}'_{21}, \tag{2.5.27}$$
$$\widetilde{A}'_{22} \leftarrow \widetilde{A}'_{22} \ominus \widetilde{L}'_{21} \odot \widetilde{U}'_{12} \text{ puis } \widetilde{A}'_{22} = \widetilde{L}'_{22}\widetilde{U}'_{22}. \tag{2.5.28}$$

Toutes ces équations consistent en la résolution classique d'un système triangulaire. Les équations (2.5.27) et (2.5.28) nécessitent cependant une multiplication formatée en supplément de cette résolution. Cette succession d'équations constituent l'algorithme 2.5.6, dans lequel HMAT_MUL_SUB désigne la multiplication formatée et LT_SOLVE la descente classique.

Algorithme 2.5.6 Descente à gauche dans la décomposition $\mathcal{H}$-LU.

1: **procedure** HMAT_LT_SOLVE($\widetilde{A}'_{|b}$,$\widetilde{L}'$,$\widetilde{U}'$,ε_{LU})
2: **if** Card(sons(b)) = 4 **then** ▷ Si b a quatre enfants.
3: **call** HMAT_LT_SOLVE($\widetilde{A}'_{11}, \widetilde{L}'_{11}, \widetilde{U}'_{11}, \varepsilon_{LU}$) ▷ Récursion (2.5.25).
4: **call** HMAT_LT_SOLVE($\widetilde{A}'_{12}, \widetilde{L}'_{11}, \widetilde{U}'_{12}, \varepsilon_{LU}$) ▷ Récursion (2.5.26).
5: **call** HMAT_MUL_SUB($\widetilde{A}'_{21}, \widetilde{L}'_{21}, \widetilde{U}'_{11}, \varepsilon_{LU}$) ▷ Multiplication formatée (2.5.27).
6: **call** HMAT_LT_SOLVE($\widetilde{A}'_{21}, \widetilde{L}'_{22}, \widetilde{U}'_{21}, \varepsilon_{LU}$) ▷ Récursion (2.5.27).
7: **call** HMAT_MUL_SUB($\widetilde{A}'_{22}, \widetilde{L}'_{21}, \widetilde{U}'_{12}, \varepsilon_{LU}$) ▷ Multiplication formatée (2.5.28).
8: **call** HMAT_LT_SOLVE($\widetilde{A}'_{22}, \widetilde{L}'_{22}, \widetilde{U}'_{22}, \varepsilon_{LU}$) ▷ Récursion (2.5.27)
9: **else** ▷ Si A est une feuille.
10: **call** LT_SOLVE($\widetilde{A}', \widetilde{L}', \widetilde{U}'$) ▷ Descente classique.
11: **end if**
12: **end procedure**

La routine HMAT_UT_SOLVE en ligne 5 de l'algorithme 2.5.5 suit des étapes similaires à la routine HMAT_LT_SOLVE.

Remarque 2.5.3. *En pratique, la routine* LU *(resp.* HMAT_LU*) ne prend pas d'arguments $\widetilde{L}$ et $\widetilde{U}$. La matrice (resp. $\mathcal{H}$-matrice) $\widetilde{A}$ est en fait écrasée par $\widetilde{L}$ sur sa partie triangulaire inférieure ($\widetilde{L}$ étant à diagonale unitaire, sa diagonale n'est pas stockée) et $\widetilde{U}$ sur sa partie triangulaire supérieure.*

Par la suite, nous noterons $\widetilde{LU}$ la $\mathcal{H}$-matrice stockant la décomposition $\mathcal{H}$-LU d'une $\mathcal{H}$-matrice $\widetilde{A}$ donnée.

Le temps de réalisation de la décomposition $\mathcal{H}$-LU est proportionnel à $n\log^3(n)$ [30] et dépend du choix de ε_{LU} (*cf.* figure 2.5.7). Ce temps est également plus faible si l'algorithme est réalisé sur une matrice ayant subit une opération de *coarsening* au préalable (*cf.* figure 2.5.8).

La décomposition $\mathcal{H}$-LU nous permet à présent d'adapter la résolution LU directe d'un système linéaire au cas des $\mathcal{H}$-matrices. Elle sera également utile afin de construire un préconditionneur de solveur itératif tel que GMRES (généralisation de la méthode de Minimisation du Résidu, de l'anglais *Generalized Minimal Residual method*) [20].

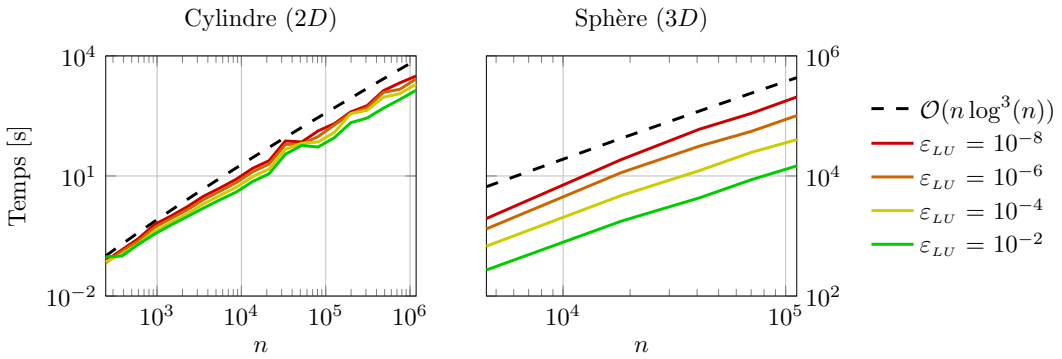


FIGURE 2.5.7 – Temps d'exécution de la décomposition $\mathcal{H}$-LU pour différentes valeurs de ε_{LU} ($\varepsilon_{coars.}$ =10^{-4}).

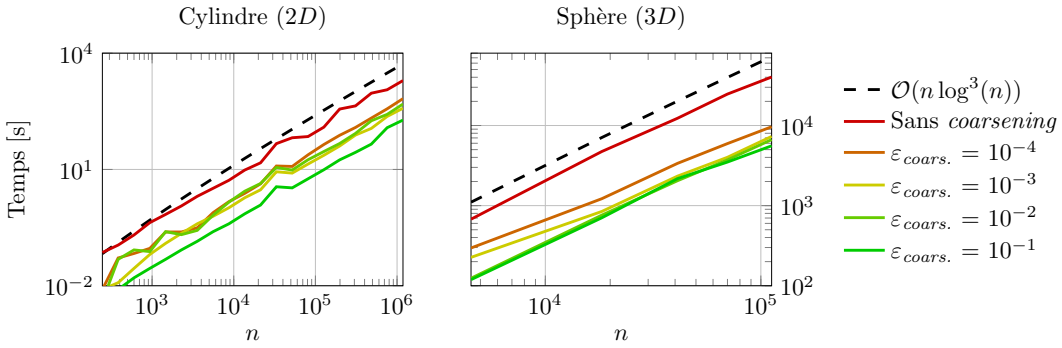


FIGURE 2.5.8 – Temps d'exécution de la décomposition $\mathcal{H}$-LU pour différentes valeurs de $\varepsilon_{coars.}$ (ε_{LU} =10^{-4}).

2.5.5 Résolution $\mathcal{H}$-LU

Pour adapter le solveur direct LU explicité en annexe A.1 au format $\mathcal{H}$-matrice, nous appliquons d'abord la décomposition $\mathcal{H}$-LU à la $\mathcal{H}$-matrice $\widetilde{Z}$. Nous obtenons alors $\widetilde{Z} = \widetilde{L}\widetilde{U}$, où $\widetilde{L}$ et $\widetilde{U}$ sont également des $\mathcal{H}$-matrices de même structure hiérarchique que $\widetilde{Z}$.

Reste donc à adapter les deux étapes de résolution suivantes :

- d'une part, le système linéaire triangulaire inférieur

$$\widetilde{L}\boldsymbol{j}' = \boldsymbol{e}, \tag{2.5.29}$$

- d'autre part, système linéaire triangulaire supérieur

$$\widetilde{U}\boldsymbol{j} = \boldsymbol{j}'. \tag{2.5.30}$$

Nous ne détaillerons que la première étape dans le cas d'une $\mathcal{H}$-matrice à un niveau de hiérarchie, la seconde lui étant très similaire. Nous verrons comment généraliser ces étapes à un niveau de hiérarchie par un algorithme récursif.

Nous souhaitons donc résoudre $\widetilde{L}\boldsymbol{j}' = \boldsymbol{e}$ où $\widetilde{L}$ est une matrice triangulaire inférieure de quatre blocs, et $\boldsymbol{j}'$ et $\boldsymbol{e}$ sont deux vecteurs que l'on peut décomposer en deux sous-vecteurs tels que

$$\begin{bmatrix} \widetilde{L}_{11} & 0 \\ \widetilde{L}_{21} & \widetilde{L}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{j}'_1 \\ \boldsymbol{j}'_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \end{bmatrix}, \tag{2.5.31}$$

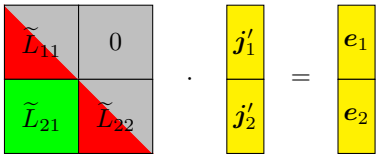


FIGURE 2.5.9 – Résolution du système linéaire triangulaire inférieur $\widetilde{L}\boldsymbol{j}' = \boldsymbol{e}$ à un niveau de hiérarchie.

où $\widetilde{L}_{ii}$ est triangulaire inférieure ($i \in [\![1, 2]\!]$).

Cette résolution se déroule suivant les étapes

$$\widetilde{L}_{11}\boldsymbol{j}'_1 = \boldsymbol{e}_1, \tag{2.5.32}$$

$$\boldsymbol{e}_2 \leftarrow \boldsymbol{e}_2 - \widetilde{L}_{21}\boldsymbol{j}'_1, \tag{2.5.33}$$

$$\widetilde{L}_{22}\boldsymbol{j}'_2 = \boldsymbol{e}_2. \tag{2.5.34}$$

Les étapes (2.5.32) et (2.5.34) sont des systèmes triangulaires inférieurs classiques et l'étape (2.5.33) se compose d'un produit $\mathcal{H}$-matrice-vecteur et d'une soustraction de vecteurs. Aucune de ces trois étapes n'implique d'approximation supplémentaire.

La généralisation de cette résolution à plusieurs niveaux de hiérarchie est décrite à l'algorithme 2.5.7. La routine L_SOLVE y désigne la résolution triangulaire inférieure classique, utilisant la routine LAPACK consacrée (-TRTRS). La routine HMAT_VECT fait référence au produit $\mathcal{H}$-matrice-vecteur explicité à l'algorithme 2.5.2.

Algorithme 2.5.7 Résolution triangulaire inférieure $\widetilde{L}_{|b}\boldsymbol{j}' = \boldsymbol{e}$.

```
1: procedure HMAT_L_SOLVE(L̃_|b, j', e)
2:     if Card(sons(b)) = 4 then                    ▷ Si b a quatre enfants.
3:         call HMAT_L_SOLVE(L̃_11, j'_1, e_1)       ▷ Récursion (2.5.32).
4:         call HMAT_VECT(L̃_21, j'_1, e'_2)
5:         e_2 ← e_2 − e'_2
6:         call HMAT_L_SOLVE(L̃_22, j'_2, e_2)       ▷ Récursion (2.5.34).
7:     else                                         ▷ Si b est une feuille.
8:         call L_SOLVE(L̃, j', e)                   ▷ Résolution classique.
9:     end if
10: end procedure
```

La résolution triangulaire supérieure $\widetilde{U}\boldsymbol{j} = \boldsymbol{j}'$ est réalisée de façon similaire à la résolution triangulaire inférieure que nous venons de définir. Il en résulte l'algorithme récursif HMAT_L_SOLVE que nous ne détaillerons donc pas ici.

L'algorithme 2.5.8 donne enfin la solution $\boldsymbol{j}$ du système (1.1.81).

Algorithme 2.5.8 Résolution $\mathcal{H}$-LU du système $\widetilde{Z}_{|b}\boldsymbol{j} = \boldsymbol{e}$.

```
1: procedure HMAT_LU_SOLVE(Z̃_|b, j, e, ε_LU)
2:     call HMAT_LU(Z̃_|b, L̃, Ũ, ε_LU) ▷ Décomposition H-LU de Z̃_|b (cf. algorithme 2.5.5).
3:     call HMAT_L_SOLVE(L̃, j', e)      ▷ Résolution triangulaire inférieure (2.5.29).
4:     call HMAT_U_SOLVE(Ũ, j, j')      ▷ Résolution triangulaire supérieure (2.5.30).
5: end procedure
```

On peut notamment remarquer que l'approximation, contrôlée par ε_{LU}, n'intervient qu'au moment de la décomposition $\mathcal{H}$-LU.

Conclusion du chapitre

Dans ce chapitre, nous nous sommes familiarisés avec les notions mathématiques fondatrices du format $\mathcal{H}$-matrice.

En particulier, nous avons pu comprendre l'origine et la validité de l'approximation d'un bloc matriciel dense en r_k-matrice. Ce format de stockage compressé offre des avantages en terme de coût mémoire, et est facilement utilisable grâce aux opérations arithmétiques existantes sur les r_k-matrices. Ces r_k-matrices remplaceront donc certains blocs dits admissibles de notre matrice globale. Nous avons ensuite établi les bases théoriques sur lesquelles s'appuie la notion de *block cluster tree*, structure qui incarne la hiérarchie d'une $\mathcal{H}$-matrice. Une fois l'assemblage d'une $\mathcal{H}$-matrice effectué, il est possible de réduire encore son coût mémoire en y appliquant un algorithme de *coarsening* qui en simplifie la hiérarchie. Tout comme pour les r_k-matrices, il est alors possible de définir une arithmétique propre aux $\mathcal{H}$-matrices, comprenant des opérations dont la complexité numérique est réduite par rapport à l'arithmétique des matrices denses.

À présent il convient de déterminer quelles sont les méthodes de compression disponibles pour obtenir des r_k-matrices, et d'utiliser l'arithmétique des $\mathcal{H}$-matrices pour résoudre divers problèmes.

Chapitre 3

Compression et résolution d'un système linéaire hiérarchique

Dans le chapitre précédent, nous avons défini le format $\mathcal{H}$-matrice et l'arithmétique associée, sans détailler les méthodes de compression qui peuvent lui être appliquées. En particulier, nous avons utilisé la méthode ACA, brièvement expliquée en partie 1.3.3, sans prendre le temps de valider cette méthode, ni de vérifier ses limitations. Pour s'affranchir des inconvénients de l'ACA, Steffen Börm et Lars Grasedyck [10] ont développé une nouvelle méthode de compression, nommée Approximation en Croix Hybride (en anglais Hybrid Cross Approximation*, notée HCA). Cette méthode hybride est encore peu utilisée à l'heure actuelle. Il nous a donc semblé intéressant d'en évaluer l'application aux équations de Maxwell, et en particulier à la formulation EFIE.*

Par ailleurs, il reste encore à utiliser l'arithmétique des $\mathcal{H}$-matrices que nous avons décrite au chapitre précédent afin de résoudre le système linéaire (1.1.81).

Nous commencerons donc par une étude plus approfondie des performances de l'ACA, puis nous présenterons la méthode de compression HCA et l'appliquerons à nos cas de référence. Nous adapterons enfin quelques méthodes de résolution de systèmes linéaires au format $\mathcal{H}$-matrice.

Mots clés

ACA, HCA, solveur direct $\mathcal{H}$-LU, préconditionnement de méthodes itératives

3.1 Performances de la compression par ACA

Nous avons décrit en partie 1.3.3 l'algorithme ACA (*cf.* algorithme 1.3.1). Nous avons ensuite utilisé cette méthode de compression tout au long du chapitre 2 dans le but de valider la complexité des opérations de l'arithmétique des $\mathcal{H}$-matrices. Toutefois nous n'avons pas encore déterminé de règle liant le choix de la précision ε_{ACA} et la précision de la matrice obtenue par compression ACA. C'est entre autres ce que nous allons faire dans cette section.

3.1.1 Évaluation de l'erreur provoquée par l'ACA

L'ACA est un algorithme itératif qui construit une approximation $\widetilde{Z}_{|b}$ de rang k d'un bloc matriciel $Z_{|b}$ sous la forme d'une r_k-matrice au format (2.1.4). L'erreur commise sur le bloc est estimée à chaque itération. Les itérations s'arrêtent à convergence, c'est-à-dire lorsque l'erreur évaluée est inférieure à l'erreur prescrite ε_{ACA}. Dans ce cas l'algorithme s'arrête lorsque toutes les colonnes de $Z_{|b}$ sont construites.

Pour évaluer l'influence de ε_{ACA} sur la précision de cette méthode de compression, nous pouvons l'appliquer à nos cas de référence, décrits en fin de section 1.1. Nous construisons d'une part le bloc $Z_{|b}$ plein, et d'autre part la $\mathcal{H}$-matrice $\widetilde{Z}_{|b}$ pour différentes valeurs de ε_{ACA}. Nous générons ensuite un vecteur aléatoire $\boldsymbol{x}$ et observons l'erreur relative $\|\widetilde{Z}_{|b}\boldsymbol{x} - Z_{|b}\boldsymbol{x}\|_2 \,/\, \|Z_{|b}\boldsymbol{x}\|_2$ commise par cette compression. Cette évaluation demande le calcul de la matrice pleine, ce qui nous contraint à ne la réaliser qu'avec un nombre limité de DDLs (*cf.* figure 3.1.1).

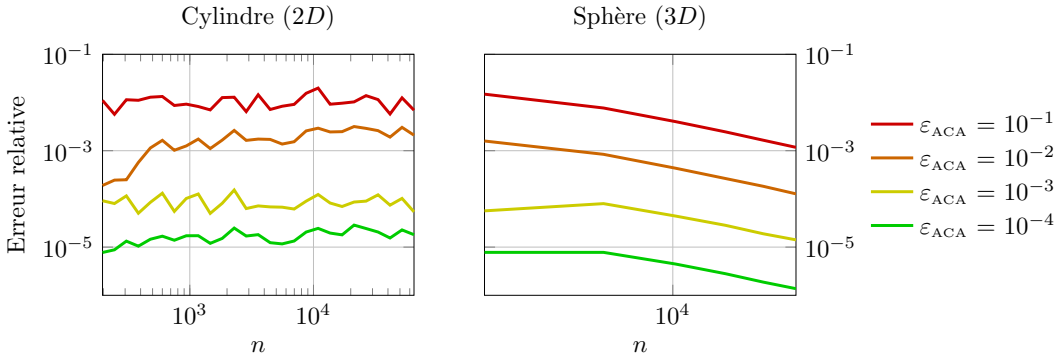


FIGURE 3.1.1 – Erreur relative induite par la compression ACA pour différentes valeurs de ε_{ACA} (sans *coarsening*).

Sur ces exemples, nous pouvons constater que l'erreur relative observée est bien inférieure à l'erreur prescrite ε_{ACA} quelle qu'en soit la valeur. Ceci nous permet de "valider" la précision de cette méthode de compression.

3.1.2 Adaptation des précisions de l'ACA et du *coarsening*

Nous avons détaillé au chapitre 2 l'algorithme de recompression par *coarsening* (*cf.* algorithme 2.4.2). Pour rappel, cet algorithme permet de réduire la constante de rareté $c_{\text{sp}}(\widetilde{Z})$ d'une $\mathcal{H}$-matrice $\widetilde{Z}$, ce qui se traduit par une simplification de sa hiérarchie. Nous avons également montré que le *coarsening* permet de réduire l'espace mémoire nécessaire à stocker $\widetilde{Z}$ ainsi que le temps de réalisation des opérations arithmétiques sur $\widetilde{Z}$.

Il convient à présent d'apprécier l'influence du *coarsening* sur la précision de la méthode. Pour cela, fixons $\varepsilon_{\text{ACA}} = 10^{-4}$ et observons l'erreur relative entre la matrice pleine

Z et la matrice creuse $\widetilde{Z}$ obtenue par compression ACA suivie d'un *coarsening* pour différentes valeurs de $\varepsilon_{coars.}$. Cette nouvelle erreur relative peut être observée sur la figure 3.1.2.

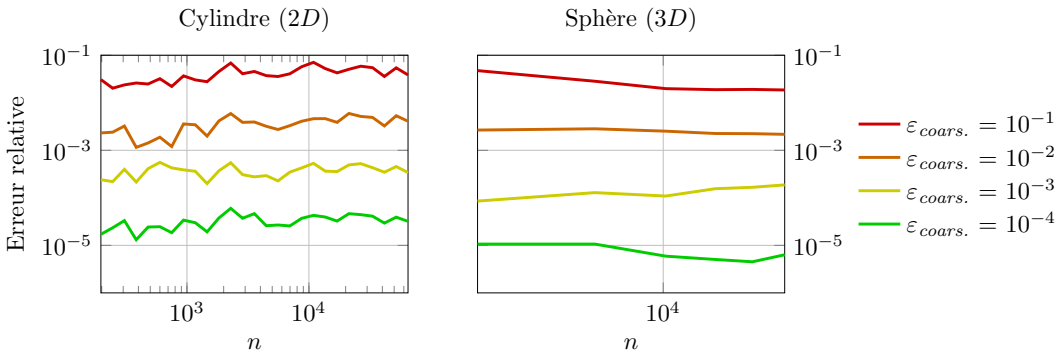


FIGURE 3.1.2 – Erreur relative induite par la compression ACA complétée du *coarsening* pour différentes valeurs de $\varepsilon_{coars.}$ ($\varepsilon_{\text{ACA}} = 10^{-4}$).

En comparaison avec la figure 3.1.1, nous vérifions que l'erreur observée est plus importante avec *coarsening* que sans recompression, quelle que soit la valeur de $\varepsilon_{coars.} \geqslant \varepsilon_{\text{ACA}}$. Le *coarsening* dégrade donc légèrement la précision de la méthode de compression, mais le critère de précision fixé par $\varepsilon_{coars.}$ est respecté, et reste en particulier en accord avec ε_{ACA} lorsque ces deux précisions sont égales.

Les figures précédentes nous permettent d'ores et déjà d'établir une règle d'accord entre $\varepsilon_{coars.}$ et ε_{ACA}. En effet, pour une précision de ε_{ACA} souhaitée, il semble convenable de fixer

$$\varepsilon_{coars.} \coloneqq \varepsilon_{\text{ACA}}. \tag{3.1.1}$$

Dès à présent, et sauf mention contraire, nous adopterons ce réglage.

3.1.3 Influence de l'intégration numérique

Si l'ACA permet un gain de temps substantiel en ne calculant qu'une partie des coefficients du bloc matriciel $Z_{|b}$ à approcher, cette économie ne concerne que la partie régulière de la matrice (*cf.* remarque 1.1.5). Chacun des coefficients concernés nécessite une double intégration. Ceci implique, comme dans le cas de l'assemblage de la matrice pleine (*cf.* partie 1.4.2), une dépendance quadratique en n_{Gauss} du temps d'assemblage.

Pour évaluer la part quadratique du temps d'assemblage dans le cas de l'ACA, nous comparons pour quelques exemples, dans le tableau 3.1.1, le rapport t/n_{Gauss}^2, où t désigne le temps d'assemblage pour $\varepsilon_{\text{ACA}} = 10^{-4}$ de la $\mathcal{H}$-matrice $\widetilde{Z}$.

Nous pouvons constater que cette dépendance quadratique se vérifie bien dès lors que n_{Gauss} est strictement supérieur à 1. En effet, le rapport t/n_{Gauss}^2 n'est pas tout à fait constant, et vaut environ deux fois plus pour $n_{\text{Gauss}} = 1$ que pour les autres valeurs de n_{Gauss} considérées. Ce phénomène se retrouve dans le cas du calcul de la matrice pleine (*cf.* figure 1.4.1). Ainsi, plus le nombre de DDLs est élevé et moins la partie singulière de la matrice $\mathcal{H}$-matrice $\widetilde{Z}$ n'aura d'influence sur le coût total de son assemblage.

3.1.4 Choix de la précision de compression selon la taille du problème

Dans le chapitre 2, nous avions pu voir que le choix de ε_{ACA} avait une influence sur le temps d'assemblage. En effet, celui-ci est d'autant plus long que la valeur de ε_{ACA} est petite. Ce temps reste toutefois proportionnel à $n \log(n)$ quelle que soit la valeur de ε_{ACA}.

Cylindre ($2D$)					Sphère ($3D$)				
		n_{Gauss}					n_{Gauss}		
n		1	2	3	n		1	3	7
1463	t (s)	0,544	1,268	2,476	1155	t (s)	5,420	20,732	125,004
	$\frac{t}{n_{\text{Gauss}}^2}$ (s)	0,544	0,317	0,275		$\frac{t}{n_{\text{Gauss}}^2}$ (s)	5,420	2,304	2,551
13 625	t (s)	4,708	12,744	25,752	10 302	t (s)	102,904	405,220	2056,788
	$\frac{t}{n_{\text{Gauss}}^2}$ (s)	4,708	3,186	2,861		$\frac{t}{n_{\text{Gauss}}^2}$ (s)	102,904	45,024	45,975
126 888	t (s)	57,620	138,888	273,732	112 206	t (s)	2293,422	7783,980	39 173,420
	$\frac{t}{n_{\text{Gauss}}^2}$ (s)	57,620	34,722	30,415		$\frac{t}{n_{\text{Gauss}}^2}$ (s)	2293,422	864,887	799,460

TABLEAU 3.1.1 – Temps d'assemblage par ACA en fonction de n_{Gauss}.

Vérifions maintenant l'influence de ε_{ACA} sur la précision par rapport à la solution analytique. Pour ceci, nous évaluerons l'erreur relative entre la solution numérique du système creux $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ et la solution analytique $\boldsymbol{j}_{\text{réf.}}$.

Pour résoudre le système creux $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$, nous utiliserons le solveur itératif GMRES (*cf.* algorithme A.1) adapté au format des $\mathcal{H}$-matrices. Pour cela, il suffit de remplacer les produits matrice-vecteur de l'algorithme A.1 par des produits $\mathcal{H}$-matrice-vecteur telles que décrites en partie 2.5.2 (*cf.* algorithme 2.5.2). Cette adaptation nous permet d'obtenir un solveur itératif rapidement.

Ici nous fixons $\varepsilon_{\text{GMRES}} = 10^{-12}$. Cette précision très fine nous assure que l'erreur mesurée correspond à l'erreur induite par la compression et non par la résolution. L'erreur relative alors obtenue est présentée dans le cas de la sphère en figure 3.1.3.

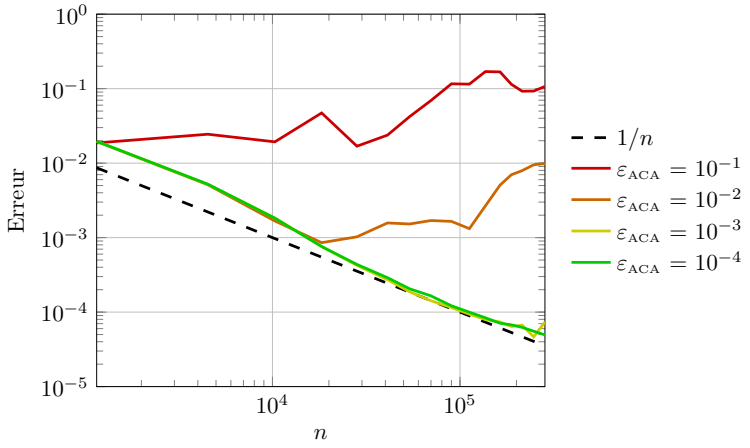


FIGURE 3.1.3 – Erreur relative par rapport à la solution analytique pour différentes valeurs de ε_{ACA}.

Cette figure nous permet de constater que la précision par rapport à la solution analytique décroît en fonction du nombre de DDLs, jusqu'à atteindre un seuil à la fois proche et inférieur à ε_{ACA}. Cette décroissance est attendue puisque plus le nombre de DDLs est élevé, plus la solution numérique est proche de la solution analytique. Elle a une pente proche de $1/n$. Le seuil correspond quant à lui à la limite de l'ACA, qui ne peut pas, par essence, être plus précise que ε_{ACA}.

Nous pouvons conclure de cette figure qu'il est inutile d'avoir une précision trop importante pour un nombre de DDLs donné. Par exemple, pour $n = 18294$, le pas de maillage

moyen est de $\lambda/10$; la précision obtenue par rapport à la solution analytique est sensiblement la même quelle que soit la valeur de ε_{ACA} (inférieure à 10^{-1}). Ainsi, bien que beaucoup de paramètres entrent en compte (*cf.* section 1.4), la précision de la solution numérique est surtout commandée par la discrétisation plutôt que par la précision de compression exigée. La précision sur la compression peut également s'ajuster en fonction de la précision sur la discrétisation demandée.

3.2 Approximation en Croix Hybride (HCA)

3.2.1 Motivations

Steffen Börm, Lars Grasedyck et Wolfgang Hackbusch montrent que l'estimation de l'erreur lors du déroulement de l'ACA n'est pas fiable notamment dans le cas des géométries contenant des singularités et lorsque le noyau est défini à l'aide d'opérateurs différentiels [11]. C'est dans le but de contourner ces limitations de l'ACA qu'a été développée une nouvelle méthode, l'Approximation en Croix Hybride (en anglais *Hybrid Cross Approximation*, notée HCA).

Introduite par Steffen Börm et Lars Grasedyck en 2005 [10], l'HCA est une méthode de compression qui exploite à la fois le système d'approximation en croix développé dans l'ACA et quelques résultats d'interpolation polynomiale, ce qui en fait une méthode hybride. Les travaux de thèse de Kieran Delamotte [36] la considèrent mais ne sont pas accessibles à ce jour. Cette méthode consiste en la construction d'une forme dégénérée du noyau de Green, que l'on utilise comme approximation du noyau réel sur des pivots choisis par méthode ACA. Elle est actuellement encore peu exploitée en électromagnétisme.

Dans cette section, nous expliciterons les principes de cette méthode de compression ainsi que ses développements pour l'EFIE dans le cas du mode TM en dimension 2, puis dans le cas général en dimension 3.

3.2.2 Principes de la méthode

On définit la méthode pour un noyau g de la forme

$$g(\boldsymbol{x}, \boldsymbol{y}) = D_{\boldsymbol{x}} D_{\boldsymbol{y}} \gamma(\boldsymbol{x}, \boldsymbol{y}), \tag{3.2.1}$$

où $D_{\boldsymbol{x}}$ et $D_{\boldsymbol{y}}$ sont des opérateurs différentiels respectivement suivant $\boldsymbol{x}$ et $\boldsymbol{y}$, et γ une fonction *asymptotiquement lisse* (*cf.* définition 1.3.1), ayant notamment les bonnes propriétés pour l'ACA (*cf.* partie 1.3.3).

Soit un bloc matriciel admissible $Z \in \mathbb{C}^{I \times J}$, I étant l'ensemble des coefficients de ses lignes et J l'ensemble des coefficients de ses colonnes. Pour $i \in I$ et $j \in J$, les coefficients de la matrice Z construite à partir de la méthode de Galerkine sont de la forme

$$z_{ij} = \int_{\Gamma \times \Gamma} \varphi_i(\boldsymbol{x}) g(\boldsymbol{x}, \boldsymbol{y}) \varphi_j(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y}. \tag{3.2.2}$$

On utilise alors un noyau dégénéré $\widetilde{g}$ pour définir une approximation $\widetilde{Z} = (\widetilde{z}_{ij})_{i \in I, j \in J}$ de Z par

$$\widetilde{z}_{ij} = \int_{\Gamma \times \Gamma} \varphi_i(\boldsymbol{x}) \widetilde{g}(\boldsymbol{x}, \boldsymbol{y}) \varphi_j(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{y} \tag{3.2.3}$$

Sélection de pivots

Nous reprendrons ici les notations introduites aux sections 2.2.2 et 2.3.3.

Supposons que Ω_I et Ω_J forment une paire admissible de domaines et considérons $\mathcal{B}_I$ (resp. $\mathcal{B}_J$), la boîte englobante de Ω_I (resp. Ω_J). La première étape de l'HCA consiste à sélectionner des points de calculs $(\boldsymbol{x}_i)_{i\in I}$ dans $\mathcal{B}_I$ et $(\boldsymbol{y}_j)_{j\in J}$ dans $\mathcal{B}_J$. Pour ce faire nous utiliserons l'algorithme ACA.

Jusqu'à maintenant, l'algorithme ACA nous a permis de déterminer une approximation locale de la matrice impédance Z. Dans le cadre de l'HCA, l'utilisation de l'ACA nous donne cette fois une approximation locale $\widetilde{g}$ de la fonction γ. Dans un souci de clarté, nous distinguerons ces deux algorithmes en notant "ACA$_\gamma$" l'ACA appliquée dans le cadre de l'HCA.

L'algorithme ACA$_\gamma$ est appliqué sur le couple de grilles de points $(\mathcal{G}_I, \mathcal{G}_J)$, la grille $\mathcal{N}_I$ (resp. $\mathcal{N}_J$) quadrillant la boîte $\mathcal{B}_I$ (resp. $\mathcal{B}_J$). On commence par le choix d'un premier pivot $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ tel que $\gamma(\boldsymbol{x}_1, \boldsymbol{y}_1) \neq 0$. On définit la fonction γ_1 qui à $\boldsymbol{x}$ et $\boldsymbol{y} \in \mathbb{R}^3$ associe

$$\gamma_1(\boldsymbol{x}, \boldsymbol{y}) = \frac{\gamma(\boldsymbol{x}, \boldsymbol{y}_1)\gamma(\boldsymbol{x}_1, \boldsymbol{y})}{\gamma(\boldsymbol{x}, \boldsymbol{y})}. \tag{3.2.4}$$

γ_1 fournit alors une première approximation de γ, et on note $r_1 = \gamma - \gamma_1$ le résidu de cette approximation.

L'étape suivante consiste à choisir un second pivot $(\boldsymbol{x}_2, \boldsymbol{y}_2)$ dans $(\mathcal{N}_I, \mathcal{N}_J)$, tel que $r_1(\boldsymbol{x}_2, \boldsymbol{y}_2)$ soit maximal (et non nul), puis à définir une nouvelle approximation γ_2 de γ définie par

$$\gamma_2(\boldsymbol{x}, \boldsymbol{y}) = \gamma_1(\boldsymbol{x}, \boldsymbol{y}) + \frac{r_1(\boldsymbol{x}, \boldsymbol{y}_2)r_1(\boldsymbol{x}_2, \boldsymbol{y})}{r_1(\boldsymbol{x}, \boldsymbol{y})}. \tag{3.2.5}$$

On définit alors $r_2 = \gamma - \gamma_2$ le résidu de cette deuxième approximation.

On réitère ce processus jusqu'à ce que le résidu évalué sur le domaine d'approximation soit inférieur à la précision ε_{HCA}.

Finalement, on obtient deux listes de pivots $(\boldsymbol{x}_i)_{i\in[\![1,k]\!]}$ et $(\boldsymbol{y}_j)_{j\in[\![1,k]\!]}$, avec k le rang issu de l'ACA$_\gamma$, et le noyau dégénéré $\widetilde{g}$ s'écrit sous la forme

$$\widetilde{g}(\boldsymbol{x}, \boldsymbol{y}) = D_{\boldsymbol{x}} D_{\boldsymbol{y}} \widetilde{\gamma}(\boldsymbol{x}, \boldsymbol{y}), \tag{3.2.6}$$

avec [10]

$$\widetilde{\gamma}(\boldsymbol{x}, \boldsymbol{y}) = \begin{pmatrix} \gamma(\boldsymbol{x}, \boldsymbol{y}_1) \\ \vdots \\ \gamma(\boldsymbol{x}, \boldsymbol{y}_k) \end{pmatrix}^T M^{-1} \begin{pmatrix} \gamma(\boldsymbol{x}_1, \boldsymbol{y}) \\ \vdots \\ \gamma(\boldsymbol{x}_k, \boldsymbol{y}) \end{pmatrix}, \tag{3.2.7}$$

où $M = (m_{ij})_{i,j\in[\![1,k]\!]}$ est la matrice définie par les coefficients

$$m_{ij} = \gamma(\boldsymbol{x}_i, \boldsymbol{y}_j). \tag{3.2.8}$$

Pour déterminer les listes de pivots, il est notamment possible de partir de grilles de points $(\mathcal{G}_I, \mathcal{G}_J)$ totalement aléatoires dans les boîtes $\mathcal{B}_I$ et $\mathcal{B}_J$. Nous avons choisi d'appliquer l'ACA sur les grilles de *points d'interpolation de Tchebychev* tensorisés et adaptés aux boîtes $\mathcal{B}_I$ et $\mathcal{B}_J$ [10].

Soit m_T un entier naturel non nul et soit d la dimension du domaine considéré ($d = 2$ ou 3). La boîte $\mathcal{B}_I$ étant délimitée par les points $\boldsymbol{p}_I^-$ et $\boldsymbol{p}_I^+$, on définit les points d'interpolation de Tchebychev tensorisés pour $i, j, k \in [\![1, m_T]\!]$ par

$$x_{I_{T(i)}} = \frac{x_I^+ + x_I^-}{2} + \frac{x_I^+ - x_I^-}{2} \cos\left(\frac{2(i+1)\pi}{2m_T}\right) \tag{3.2.9}$$

$$y_{I_{T(j)}} = \frac{y_I^+ + y_I^-}{2} + \frac{y_I^+ - y_I^-}{2} \cos\left(\frac{2(j+1)\pi}{2m_T}\right) \tag{3.2.10}$$

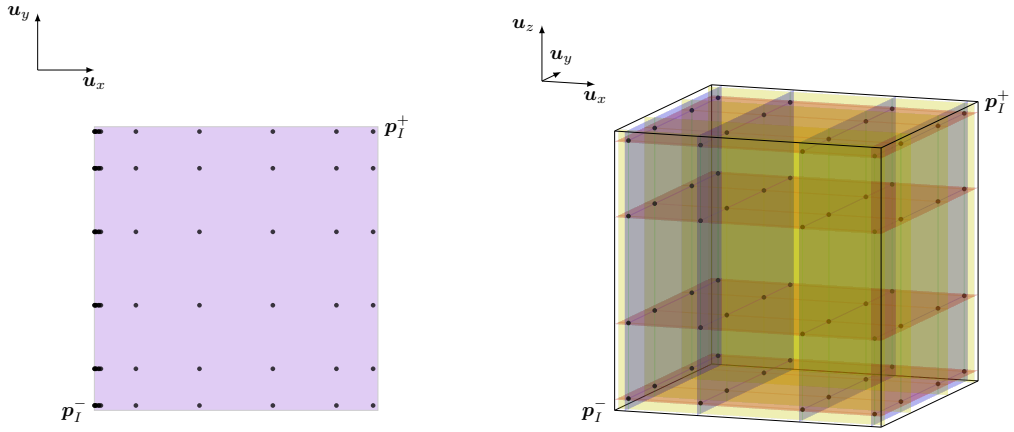
$$z_{I_{T(k)}} = \frac{z_I^+ + z_I^-}{2} + \frac{z_I^+ - z_I^-}{2} \cos\left(\frac{2(k+1)\pi}{2m_T}\right) \text{ si } d = 3, \tag{3.2.11}$$

et ces m_T^d points d'interpolation de $\mathcal{B}_I$ s'écrivent

$$\boldsymbol{p}_{I_{T(i,j)}} = \begin{pmatrix} x_{I_{T(i)}} & y_{I_{T(j)}} \end{pmatrix}^T \qquad \text{pour } d = 2, \tag{3.2.12}$$

$$\boldsymbol{p}_{I_{T(i,j,k)}} = \begin{pmatrix} x_{I_{T(i)}} & y_{I_{T(j)}} & z_{I_{T(k)}} \end{pmatrix}^T \qquad \text{pour } d = 3. \tag{3.2.13}$$

La figure 3.2.1 donne un exemple de tensorisation d'une boîte $\mathcal{B}_I$ des points de Tchebychev en dimensions 2 et 3.



(a) Cas $2D$: boîte carrée ($m_T = 6$) (b) Cas $3D$: boîte cubique ($m_T = 4$)

FIGURE 3.2.1 – Points de Tchebychev dans une boîte en dimensions 2 et 3.

Séparation des variables

Notons $N = M^{-1} = (n_{ij})_{i,j \in [\![1,k]\!]}$. Alors

$$\widetilde{\gamma}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{l=1}^{k} \sum_{m=1}^{k} n_{lm} \gamma(\boldsymbol{x}, \boldsymbol{y}_l) \gamma(\boldsymbol{x}_m, \boldsymbol{y}). \tag{3.2.14}$$

Ainsi

$$\widetilde{g}(\boldsymbol{x}, \boldsymbol{y}) = D_{\boldsymbol{x}} D_{\boldsymbol{y}} \sum_{l=1}^{k} \sum_{m=1}^{k} n_{lm} \gamma(\boldsymbol{x}, \boldsymbol{y}_l) \gamma(\boldsymbol{x}_m, \boldsymbol{y}), \tag{3.2.15}$$

soit

$$\widetilde{g}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{l=1}^{k} \sum_{m=1}^{k} n_{lm} \left(D_{\boldsymbol{x}} \gamma(\boldsymbol{x}, \boldsymbol{y}_l)\right) \left(D_{\boldsymbol{y}} \gamma(\boldsymbol{x}_m, \boldsymbol{y})\right). \tag{3.2.16}$$

On a donc

$$\widetilde{z}_{ij} = \int_{\Gamma\times\Gamma} \varphi_i(\boldsymbol{x}) \sum_{l=1}^{k}\sum_{m=1}^{k} n_{lm} \left(D_{\boldsymbol{x}}\gamma(\boldsymbol{x},\boldsymbol{y}_l)\right)\left(D_{\boldsymbol{y}}\gamma(\boldsymbol{x}_m,\boldsymbol{y})\right)\varphi_j(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}\,\mathrm{d}\boldsymbol{x}, \tag{3.2.17}$$

soit

$$\widetilde{z}_{ij} = \sum_{l=1}^{k}\sum_{m=1}^{k} n_{lm} \left(\int_{\Gamma} \varphi_i(\boldsymbol{x}) D_{\boldsymbol{x}}\gamma(\boldsymbol{x},\boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}\right)\left(\int_{\Gamma} \varphi_j(\boldsymbol{y}) D_{\boldsymbol{y}}\gamma(\boldsymbol{x}_m,\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}\right). \tag{3.2.18}$$

Expression possible de la matrice N

Il existe plusieurs stratégies pour exprimer $N = (n_{lm})_{l,m\in[\![1,k]\!]}$ de façon à séparer les indices l et m, et ainsi de pouvoir dissocier les sommes apparaissant dans (3.2.18). Steffen Börm et Lars Grasedyck proposent dans [10] une méthode récursive définissant $N \in \mathbb{R}^{k\times k}$ comme le produit de deux matrices C^T et D de $\mathbb{R}^{k\times k}$. Bien qu'il soit possible d'adapter cette méthode au cas complexe, nous avons opté pour l'utilisation de la décomposition LU de $M = N^{-1}$, que l'on peut implémenter directement dans l'algorithme ACA [37].

Ainsi nous avons

$$M = LU \implies N = M^{-1} = (LU)^{-1} = U^{-1}L^{-1}. \tag{3.2.19}$$

Posons alors

$$C \coloneqq U^{-T} = (c_{ij})_{i,j\in[\![1,k]\!]}, \tag{3.2.20}$$

et

$$D \coloneqq L^{-1} = (d_{ij})_{i,j\in[\![1,k]\!]}. \tag{3.2.21}$$

On a alors $N = C^T D = (n_{ij})_{ij\in[\![1,k]\!]}$ avec $n_{lm} = \sum_{p=1}^{k} c_{pl}d_{pm}$. Il en découle

$$\widetilde{z}_{ij} = \sum_{p=1}^{k}\left(\sum_{l=1}^{k} c_{pl}\int_{\Gamma} \varphi_i(\boldsymbol{x}) D_{\boldsymbol{x}}\gamma(\boldsymbol{x},\boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}\right)\left(\sum_{m=1}^{k} d_{pm}\int_{\Gamma} \varphi_j(\boldsymbol{y}) D_{\boldsymbol{y}}\gamma(\boldsymbol{x}_m,\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}\right). \tag{3.2.22}$$

Ainsi, en définissant les matrices $A = (a_{il})_{i\in I, l\in[\![1,k]\!]}$ et $B = (b_{jm})_{j\in J, m\in[\![1,k]\!]}$ telles que

$$a_{il} = \int_{\Gamma} \varphi_i(\boldsymbol{x}) D_{\boldsymbol{x}}\gamma(\boldsymbol{x},\boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}, \tag{3.2.23}$$

$$b_{jm} = \int_{\Gamma} \varphi_j(\boldsymbol{y}) D_{\boldsymbol{y}}\gamma(\boldsymbol{x}_m,\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}, \tag{3.2.24}$$

on obtient

$$\widetilde{z}_{ij} = \sum_{p=1}^{k}\left(\sum_{l=1}^{k} c_{pl}a_{il}\right)\left(\sum_{m=1}^{k} d_{pm}b_{jm}\right). \tag{3.2.25}$$

Autrement dit,

$$\widetilde{Z} = UV^H \text{ avec } U = AC^T \text{ et } V^H = BD^T. \tag{3.2.26}$$

Ceci constitue une approximation $\widetilde{Z}$ de Z ayant pour rang k.

Remarque 3.2.1. *L'écriture (3.2.26) est une r_k-matrice au format (2.1.4). Pour en déduire une r_k-matrice au format (2.1.23), il suffit d'y appliquer la r_k-SVD à une précision ε donnée. En général, on pose*

$$\varepsilon = \varepsilon_{HCA}. \tag{3.2.27}$$

Algorithme de la compression HCA

Pour résumer, l'HCA vise à construire une approximation $\widetilde{Z} = UV^H$ d'un bloc matriciel $Z \in \mathbb{C}^{I\times J}$ et se déroule selon les étapes suivantes :

1. détermination des pivots $(\boldsymbol{x}_i)_{i\in[\![1,k]\!]}$ et $(\boldsymbol{y}_j)_{j\in[\![1,k]\!]}$ par ACA à précision ε_{HCA} sur les m_T^d points de Tchebychev,
2. décomposition LU de la matrice $M = (\gamma(x_i, y_j))_{i,j\in[\![1,k]\!]}$,
3. calcul des potentiels de simple couche $a_{il} = \mathcal{S}_c(\boldsymbol{\varphi}_i)(\boldsymbol{y}_l)$ et $b_{jm} = \mathcal{S}_c(\boldsymbol{\varphi}_j)(\boldsymbol{x}_m)$ pour $i \in I$, $j \in J$ et $l, m \in [\![1, k]\!]$,
4. construction par multiplication de U et V.

On en déduit l'algorithme 3.2.1.

Algorithme 3.2.1 *Hybrid Cross Approximation* (HCA).

procedure HCA(Z,ε_{HCA},m_T,U,V)
  call TCHEBYCHEV($\mathcal{B}_I, m_T, (\boldsymbol{p}_i^{\mathcal{I}})_{i\in[\![1,m_T^d]\!]}$)
  call TCHEBYCHEV($\mathcal{B}_J, m_T, (\boldsymbol{p}_j^{\mathcal{J}})_{j\in[\![1,m_T^d]\!]}$)
  call ACA($(\boldsymbol{p}_i^{\mathcal{I}}), (\boldsymbol{p}_j^{\mathcal{J}}), \varepsilon_{\text{HCA}}, \mathcal{I}', \mathcal{J}', L_M, U_M$)
  $k = \text{size}(\mathcal{I}')$
  for $l = 1, k$ **do**
    $\boldsymbol{x}_l = \boldsymbol{p}^{\mathcal{I}}_{\mathcal{I}'(l)}$
    $\boldsymbol{y}_l = \boldsymbol{p}^{\mathcal{J}}_{\mathcal{J}'(l)}$
  end for
  $C = U_M^{-T}$
  $D = L_M^{-1}$
  call BUILD_A($(\boldsymbol{y}_i)_{i\in[\![1,k]\!]}, A$)
  call BUILD_B($(\boldsymbol{x}_j)_{j\in[\![1,k]\!]}, B$)
  $U \coloneqq AC^T$
  $V^H \coloneqq BD^T$
end procedure

La routine TCHEBYCHEV donne, en fonction de la boîte englobante et de l'ordre d'interpolation souhaité, les coordonnées des points d'interpolation correspondants. La routine ACA se charge, en plus de fournir les listes de pivots $\mathcal{I}'$ et $\mathcal{J}'$, d'effectuer la décomposition LU de $M = (\gamma(\boldsymbol{x}_i, \boldsymbol{y}_j))_{i,j\in[\![1,k]\!]}$. Enfin les routines BUILD_A et BUILD_B permettent de construire les matrices A et B selon leur définition. Nous verrons dans la suite comment les définir pour la formulation EFIE dans le cas d'un problème réduit en dimension 2 (mode TM), ainsi que dans le cas général en dimension 3.

3.2.3 Expression de l'HCA

HCA d'un potentiel de simple couche ($2D$)

Rappelons l'expression des coefficients de $Z = (z_{ij})_{i\in I, j\in J}$ comme explicités en partie 1.1.3

$$z_{ij} = \int_{\Gamma\times\Gamma} \varphi_i(\boldsymbol{x}) G(\boldsymbol{x}, \boldsymbol{y}) \varphi_j(\boldsymbol{y}) \, \mathrm{d}\boldsymbol{y} \, \mathrm{d}\boldsymbol{x}, \tag{3.2.28}$$

avec les fonctions de base $(\varphi_i)_{i\in I}$ définies par

$$\varphi_i(\boldsymbol{x}) = 1_{|s_i}(\boldsymbol{x}), \tag{3.2.29}$$

et G le noyau de Green défini par

$$G(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\mathrm{i}} H_0^{(1)}(k_0\,|\boldsymbol{x}-\boldsymbol{y}|), \tag{3.2.30}$$

pour $\boldsymbol{x} \in \Omega_I$ et $\boldsymbol{y} \in \Omega_J$, où (Ω_I, Ω_J) constitue une paire admissible de sous-domaines de la frontière Γ.

Dans ce cas, les boîtes englobantes $\mathcal{B}_I$ et $\mathcal{B}_J$ sont des rectangles que l'on peut quadriller de m_T^2 points d'interpolation de Tchebychev, afin d'y appliquer l'algorithme ACA sur le noyau de Green. L'ACA nous fournit le rang k ainsi que les deux listes de pivots $(\boldsymbol{x}_i)_{i\in[\![1,k]\!]}$ et $(\boldsymbol{y}_j)_{j\in[\![1,k]\!]}$. Par identification, nous pouvons immédiatement en déduire l'expression des matrices $M = (m_{ij})_{i,j\in[\![1,k]\!]}$, $A = (a_{il})_{i\in I, l\in[\![1,k]\!]}$ et $B = (b_{jm})_{j\in J, m\in[\![1,k]\!]}$, permettant d'appliquer l'HCA à ce cas particulier, soit

$$m_{ij} = G(\boldsymbol{x}_i, \boldsymbol{y}_j), \tag{3.2.31}$$

$$a_{il} = \int_\Gamma \varphi_i(\boldsymbol{x}) G(\boldsymbol{x}, \boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}, \tag{3.2.32}$$

$$b_{jm} = \int_\Gamma \varphi_j(\boldsymbol{y}) G(\boldsymbol{x}_m, \boldsymbol{y})\ \mathrm{d}\boldsymbol{y}. \tag{3.2.33}$$

HCA de la formulation EFIE ($3D$)

L'expression de $Z = (z_{ij})_{i\in I, j\in J}$ dans le cas d'une formulation EFIE $3D$ (donnée en (1.1.87)) comporte des fonctions de base particulières, constituées de vecteurs de l'espace $\mathbb{R}^3$. De ce fait, la séparation des variables offerte par l'HCA mène à la construction de matrices dont les coefficients sont dans $\mathbb{C}^3$.

Notations. *Soient deux entiers naturels non nuls m et n. Considérons l'ensemble*

$$\left(\mathbb{C}^3\right)^{m\times n} = \{(\boldsymbol{a}_{ij})_{i\in[\![1,m]\!], j\in[\![1,n]\!]} \,|\, \forall i \in [\![1,m]\!] \quad \forall j \in [\![1,n]\!] \quad \boldsymbol{a}_{ij} \in \mathbb{C}^3\}. \tag{3.2.34}$$

Cet ensemble rassemble les matrices de taille $m \times n$ à coefficients dans $\mathbb{C}^3$. Nous noterons alors en gras et en majuscule tout élément de $\left(\mathbb{C}^3\right)^{m\times n}$.

En particulier, considérons pour $m, n, p \in \mathbb{N}$, les matrices $\boldsymbol{a} = (\boldsymbol{a}_{ij})_{i\in[\![1,m]\!], j\in[\![1,n]\!]} \in \left(\mathbb{C}^3\right)^{m\times n}$, $B = (b_{ij})_{i\in[\![1,n]\!], j\in[\![1,p]\!]} \in \mathbb{C}^{n\times p}$ et $\boldsymbol{C} = (\boldsymbol{c}_{ij})_{i\in[\![1,n]\!], j\in[\![1,p]\!]} \in \left(\mathbb{C}^3\right)^{n\times p}$, ainsi que le vecteur $\boldsymbol{x} = (x_i)_{i\in[\![1,3]\!]} \in \mathbb{C}^3$. Nous noterons alors :

- $\boldsymbol{A}B$ *l'opération dont le résultat* $\boldsymbol{D} = (\boldsymbol{d}_{ij})_{i\in[\![1,m]\!], j\in[\![1,p]\!]} \in \left(\mathbb{C}^3\right)^{m\times p}$ *est défini par*

$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,p]\!] \quad \boldsymbol{d}_{ij} = \sum_{k=1}^{n} b_{kj}\boldsymbol{a}_{ik}\,; \tag{3.2.35}$$

- $\boldsymbol{A}\cdot\boldsymbol{C}$ *l'opération dont le résultat* $D = (d_{ij})_{i\in[\![1,m]\!], j\in[\![1,p]\!]} \in \mathbb{C}^{m\times p}$ *est défini par*

$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,p]\!] \quad d_{ij} = \sum_{k=1}^{n} \boldsymbol{a}_{ik}\cdot\boldsymbol{c}_{kj}\,; \tag{3.2.36}$$

- $\boldsymbol{x}\cdot\boldsymbol{A}$ *l'opération dont le résultat* $D = (d_{ij})_{i\in[\![1,m]\!], j\in[\![1,n]\!]} \in \mathbb{C}^{m\times n}$ *est défini par*

$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,n]\!] \quad d_{ij} = \boldsymbol{x}\cdot\boldsymbol{a}_{ij}. \tag{3.2.37}$$

Remarque 3.2.2. *Soient $\boldsymbol{A} \in \left(\mathbb{C}^3\right)^{m\times n}$ et $\boldsymbol{B} \in \left(\mathbb{C}^3\right)^{p\times n}$. Construisons les matrices $\widehat{A} \in \mathbb{C}^{m\times 3p}$ et $\widehat{B} \in \mathbb{C}^{n\times 3p}$ telles que*

$$\widehat{A} = \begin{bmatrix} \boldsymbol{u}_x\cdot\boldsymbol{A} & \boldsymbol{u}_y\cdot\boldsymbol{A} & \boldsymbol{u}_z\cdot\boldsymbol{A} \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix}, \tag{3.2.38}$$

$$\widehat{B} = \begin{bmatrix} \boldsymbol{u}_x \cdot \boldsymbol{B} & \boldsymbol{u}_y \cdot \boldsymbol{B} & \boldsymbol{u}_z \cdot \boldsymbol{B} \end{bmatrix} = \begin{bmatrix} B_x & B_y & B_z \end{bmatrix}. \tag{3.2.39}$$

Alors

$$\widehat{A}\widehat{B}^H = \begin{bmatrix} A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} B_x^H \\ B_y^H \\ B_z^H \end{bmatrix} \tag{3.2.40}$$

$$= \begin{bmatrix} A_x B_x^H + A_y B_y^H + A_z B_z^H \end{bmatrix} \tag{3.2.41}$$

$$\widehat{A}\widehat{B}^H = \boldsymbol{A} \cdot \boldsymbol{B}^H. \tag{3.2.42}$$

Rappelons à présent l'expression de $Z = (z_{ij})_{i\in I, j\in J}$ dans le cas d'une formulation EFIE donnée en (1.1.87), soit pour $i \in I$ et $j \in J$

$$z_{ij} = \int_{\Gamma\times\Gamma} G(\boldsymbol{x},\boldsymbol{y}) \left(\boldsymbol{\varphi}_i(\boldsymbol{x}) \cdot \boldsymbol{\varphi}_j(\boldsymbol{y}) - \frac{1}{k_0^2} \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{x}) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_j(\boldsymbol{y}) \right) \mathrm{d}\boldsymbol{y}\ \mathrm{d}\boldsymbol{x} \tag{3.2.43}$$

avec les fonctions de base $(\boldsymbol{\varphi}_i)_{i\in I}$ définies par les courants surfaciques locaux $(\boldsymbol{J}_i)_{i\in I}$ tels que

$$\boldsymbol{\varphi}_i(\boldsymbol{x}) = \boldsymbol{J}_i(\boldsymbol{x}), \tag{3.2.44}$$

et G le noyau de Green défini par

$$G(\boldsymbol{x},\boldsymbol{y}) = \frac{\mathrm{e}^{-\mathrm{i}k_0|\boldsymbol{x}-\boldsymbol{y}|}}{|\boldsymbol{x}-\boldsymbol{y}|} \tag{3.2.45}$$

pour $\boldsymbol{x} \in \Omega_I$ et $\boldsymbol{y} \in \Omega_J$, où (Ω_I, Ω_J) constitue une paire admissible de sous-domaines de la frontière Γ.

Cette fois-ci, les boîtes englobantes $\mathcal{B}_I$ et $\mathcal{B}_J$ sont des pavés droits que l'on peut quadriller de m_T^3 points d'interpolation de Tchebychev, afin d'y appliquer l'algorithme ACA sur le noyau de Green, qui nous fournira le rang k ainsi que les deux listes de pivots $(x_i)_{i\in[\![1,k]\!]}$ et $(y_j)_{j\in[\![1,k]\!]}$. Cependant, contrairement au cas du potentiel de simple couche (2D), l'identification est moins immédiate. Par ailleurs, les fonctions de base $(\boldsymbol{\varphi}_i)_{i\in I}$ sont des vecteurs de l'espace et non des scalaires, comme c'était le cas précédemment.

Il peut alors être judicieux de réécrire le coefficient z_{ij} sous la forme d'une différence de deux intégrales doubles, sur lesquelles il sera ensuite plus aisé d'appliquer l'HCA. On a donc pour $i \in I$ et $j \in J$,

$$z_{ij} = \int_{\Gamma\times\Gamma} G(\boldsymbol{x},\boldsymbol{y}) \boldsymbol{\varphi}_i(\boldsymbol{x}) \cdot \boldsymbol{\varphi}_j(\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}\ \mathrm{d}\boldsymbol{x} - \frac{1}{k_0^2} \int_{\Gamma\times\Gamma} G(\boldsymbol{x},\boldsymbol{y}) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{x}) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_j(\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}\ \mathrm{d}\boldsymbol{x}, \tag{3.2.46}$$

ou autrement dit

$$z_{ij} = z_{ij}^{(\mathrm{v})} - z_{ij}^{(\mathrm{s})}, \tag{3.2.47}$$

avec

$$z_{ij}^{(\mathrm{v})} = \int_{\Gamma\times\Gamma} G(\boldsymbol{x},\boldsymbol{y}) \boldsymbol{\varphi}_i(\boldsymbol{x}) \cdot \boldsymbol{\varphi}_j(\boldsymbol{y})\ \mathrm{d}\boldsymbol{y}\ \mathrm{d}\boldsymbol{x} \tag{3.2.48}$$

$$z_{ij}^{(\mathrm{s})} = \frac{1}{k_0^2} \int_{\Gamma\times\Gamma} G(\boldsymbol{x},\boldsymbol{y}) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_i(\boldsymbol{x}) \boldsymbol{\nabla}_\Gamma \cdot \boldsymbol{\varphi}_j(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}\ \mathrm{d}\boldsymbol{x}. \tag{3.2.49}$$

Nous pouvons donc voir Z comme la différence de deux matrices $Z^{(\mathrm{v})}$ et $Z^{(\mathrm{s})}$, dont les expressions sont faciles à approcher du cas générique sur lequel l'HCA est applicable. En particulier, on définit les matrices

$$M = (m_{ij})_{i\in I, j\in J} \text{ telle que} \qquad \forall i \in I \quad \forall j \in J \quad m_{ij} = G(\boldsymbol{x}_i, \boldsymbol{y}_j), \tag{3.2.50}$$

$$\boldsymbol{a} = (\boldsymbol{a}_{il})_{i\in I, l\in[\![1,k]\!]} \text{ telle que} \qquad \forall i \in I \quad \forall l \in [\![1,k]\!] \quad \boldsymbol{a}_{il} = \int_\Gamma \boldsymbol{\varphi}_i(\boldsymbol{x}) G(\boldsymbol{x}, \boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}, \tag{3.2.51}$$

$$\boldsymbol{B} = (\boldsymbol{b}_{jm})_{j\in J, m\in[\![1,k]\!]} \text{ telle que} \quad \forall j \in J \quad \forall m \in [\![1,k]\!] \quad \boldsymbol{b}_{jm} = \int_\Gamma \boldsymbol{\varphi}_j(\boldsymbol{y}) G(\boldsymbol{x}_m, \boldsymbol{y})\ \mathrm{d}\boldsymbol{y}, \tag{3.2.52}$$

$$\dot{a} = (\dot{a}_{il})_{i\in I, l\in[\![1,k]\!]} \text{ telle que} \qquad \forall i \in I \quad \forall l \in [\![1,k]\!] \quad \dot{a}_{il} = \frac{1}{k_0} \int_\Gamma \boldsymbol{\nabla}_\Gamma \boldsymbol{\varphi}_i(\boldsymbol{x}) G(\boldsymbol{x}, \boldsymbol{y}_l)\ \mathrm{d}\boldsymbol{x}, \tag{3.2.53}$$

$$\dot{B} = (\dot{b}_{jm})_{j\in J, m\in[\![1,k]\!]} \text{ telle que} \quad \forall j \in J \quad \forall m \in [\![1,k]\!] \quad \dot{b}_{jm} = \frac{1}{k_0} \int_\Gamma \boldsymbol{\nabla}_\Gamma \boldsymbol{\varphi}_j(\boldsymbol{y}) G(\boldsymbol{x}_m, \boldsymbol{y})\ \mathrm{d}\boldsymbol{y}. \tag{3.2.54}$$

Posons $n_I \coloneqq \mathrm{Card}(I)$ et $n_J \coloneqq \mathrm{Card}(J)$. Les matrices $M \in \mathbb{C}^{k\times k}$, $\dot{A} \in \mathbb{C}^{n_I\times k}$ et $\dot{B} \in \mathbb{C}^{n_J\times k}$ sont des matrices denses classiques, ayant leurs coefficients dans $\mathbb{C}$, tandis que les matrices $\boldsymbol{A}$ et $\boldsymbol{B}$ ont leurs coefficients dans $\mathbb{C}^3$.

Posons alors

$$\boldsymbol{U} = \boldsymbol{A}C^T \in \left(\mathbb{C}^3\right)^{n_I\times k}, \tag{3.2.55}$$

$$\boldsymbol{V} = (\boldsymbol{B}D^T)^H \in \left(\mathbb{C}^3\right)^{n_J\times k}, \tag{3.2.56}$$

$$\dot{U} = \dot{A}C^T \in \mathbb{C}^{n_I\times k}, \tag{3.2.57}$$

$$\dot{V} = (\dot{B}D^T)^H \in \mathbb{C}^{n_J\times k}, \tag{3.2.58}$$

avec C et D les matrices définies en (3.2.20) et (3.2.21). On a donc les approximations

$$\widetilde{Z}^{(\mathrm{v})} = \boldsymbol{U}\cdot\boldsymbol{V}^H \text{ et } \widetilde{Z}^{(\mathrm{s})} = \dot{U}\dot{V}^H. \tag{3.2.59}$$

Suivant la remarque 3.2.2, construisons les matrices, $\widehat{U} \in \mathbb{C}^{n_I\times 3k}$ et $\widehat{V} \in \mathbb{C}^{n_J\times 3k}$ telles que

$$\widehat{U} = \begin{bmatrix} \boldsymbol{u}_x\cdot\boldsymbol{U} & \boldsymbol{u}_y\cdot\boldsymbol{U} & \boldsymbol{u}_z\cdot\boldsymbol{U} \end{bmatrix} = \begin{bmatrix} U_x & U_y & U_z \end{bmatrix}, \tag{3.2.60}$$

$$\widehat{V} = \begin{bmatrix} \boldsymbol{u}_x\cdot\boldsymbol{V} & \boldsymbol{u}_y\cdot\boldsymbol{V} & \boldsymbol{u}_z\cdot\boldsymbol{V} \end{bmatrix} = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}. \tag{3.2.61}$$

Alors

$$\boldsymbol{U}\cdot\boldsymbol{U}^H = \widehat{U}\widehat{V}^H, \tag{3.2.62}$$

ce qui implique

$$\widetilde{Z} = \widetilde{Z}^{(\mathrm{v})} - \widetilde{Z}^{(\mathrm{s})} = \widehat{U}\widehat{V}^H - \dot{U}\dot{V}^H. \tag{3.2.63}$$

Ainsi, d'après la remarque 2.1.7 (somme de deux r_k-matrices de représentation (2.1.4)), nous pouvons construire les matrices $U \in \mathbb{C}^{n_I\times 4k}$ et $V \in \mathbb{C}^{n_J\times 4k}$ définies par

$$U = \begin{bmatrix} \widehat{U} & -\dot{U} \end{bmatrix}, \tag{3.2.64}$$

$$V = \begin{bmatrix} \widehat{V} & \dot{V} \end{bmatrix}. \tag{3.2.65}$$

Nous obtenons enfin

$$\widetilde{Z} = UV^H. \tag{3.2.66}$$

La r_k-matrice $\widetilde{Z}$ ainsi définie est donc au format (2.1.4) et de rang $4k$, ce qui peut être gênant pour une compression efficace. L'application d'une r_k-SVD semble donc indispensable ici.

3.2.4 Réglage des paramètres

Comme exposé à l'algorithme 3.2.1, nous pouvons considérer que l'HCA se déroule en trois parties : une ACA_γ accompagnée d'une décomposition LU, des intégrations simples et des multiplications de matrices, le tout étant suivi d'une r_k-SVD. En particulier, l'ACA_γ induit par nature une erreur d'approximation qui doit être inférieure à $\varepsilon_{\mathrm{HCA}}$. En terme de temps, le déroulement de l'ACA_γ dépend potentiellement de $\varepsilon_{\mathrm{HCA}}$, mais également de m_T. Ce dernier paramètre peut par ailleurs agir sur le rang k issu de l'ACA_γ. Enfin, l'étape comportant les intégrations numériques dépend temporellement du nombre n_{Gauss} de points de quadrature de Gauss-Legendre utilisés.

Il serait alors utile de dégager des indications de réglage de ces divers paramètres afin de s'assurer d'un assemblage à la fois précis et efficace.

Réglage de m_T

Influence sur la précision. L'ordre d'interpolation de Tchebychev m_T intervient uniquement lors de l'ACA_γ, qui fournit des pivots $(\boldsymbol{x}_i)_{i\in[\![1,k]\!]}$ et $(\boldsymbol{y}_j)_{j\in[\![1,k]\!]}$, k étant alors le rang résultant de l'ACA_γ. Ces pivots permettent alors de définir une approximation $\widetilde{\gamma}$ de γ définie en (3.2.6).

Pour évaluer l'erreur obtenue en fonction du choix de m_T, nous parcourons les blocs admissibles $b \coloneqq l \times c$ de $I \times J$ (c'est-à-dire les blocs matriciels $\widetilde{Z}_{|b} \in \mathbb{C}^{I\times J}$ de $\widetilde{Z}$). Sur chacun de ces blocs, nous réalisons l'ACA_γ à précision $\varepsilon_{\mathrm{HCA}}$, puis nous en déduisons pour tout $\boldsymbol{x} \in \mathcal{G}_l$ et $\boldsymbol{y} \in \mathcal{G}_c$ la valeur $\gamma(\boldsymbol{x},\boldsymbol{y})$ et sa version dégénérée $\widetilde{\gamma}(\boldsymbol{x},\boldsymbol{y})$. Nous évaluons l'erreur absolue entre $Z_{|b}$ et $\widetilde{Z}_{|b}$ par

$$\delta^{(\mathrm{a})}(Z_{|b},\widetilde{Z}_{|b}) \coloneqq \left(\sum_{\boldsymbol{x}\in\mathcal{B}_l,\boldsymbol{y}\in\mathcal{B}_c} |\gamma(\boldsymbol{x},\boldsymbol{y}) - \widetilde{\gamma}(\boldsymbol{x},\boldsymbol{y})|\right)^{\frac{1}{2}}. \tag{3.2.67}$$

À l'échelle de la $\mathcal{H}$-matrice $\widetilde{Z}$ globale, nous évaluons alors l'erreur totale par la formule

$$\delta^{(\mathrm{a})}(Z,\widetilde{Z}) \coloneqq \left(\sum_{b \text{ admissible}} \frac{n_b m_b}{n^2}\delta^{(\mathrm{a})}(Z_{|b},\widetilde{Z}_{|b})^2\right)^{\frac{1}{2}}, \tag{3.2.68}$$

où n_b (resp. m_b) désigne le nombre de lignes (resp. de colonnes) du bloc b. Cette erreur est représentée en figure 3.2.2 pour différentes valeurs de m_T.

Cette figure nous permet de constater que l'erreur ainsi définie s'améliore à mesure que la valeur de m_T augmente. Cependant cette amélioration n'existe plus dès lors que $m_T \geqslant 4$. Pour $\varepsilon_{\mathrm{HCA}} = 10^{-4}$, il semble donc inutile de prendre $m_T > 4$ car la précision maximale est atteinte pour ce nombre de points d'interpolation.

Influence sur le rang à l'issue de l'ACA_γ. L'ordre d'interpolation m_T a également une influence sur le rang k obtenu à l'issue de l'ACA_γ, puisque ce rang est majoré par m_T^d.

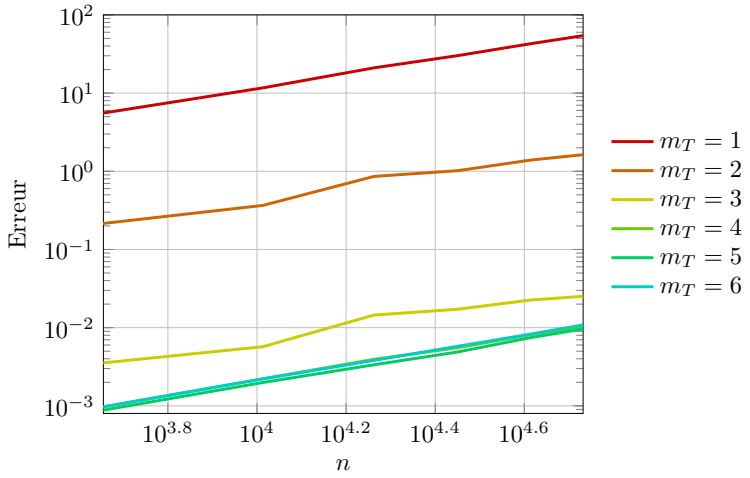


FIGURE 3.2.2 – Erreur relative induite sur le noyau par ACA pour différentes valeurs de m_T ($\varepsilon_{\text{HCA}} = 10^{-4}$).

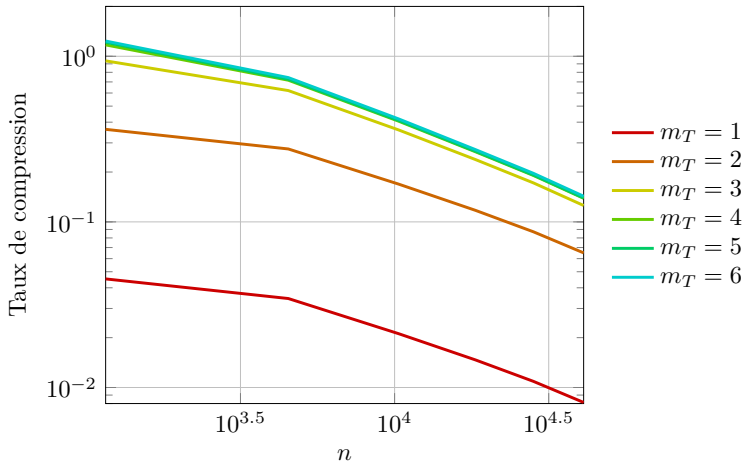


FIGURE 3.2.3 – Taux de compression sur les blocs admissibles avant recompression par r_k-SVD pour différentes valeurs de m_T ($\varepsilon_{\text{HCA}} = 10^{-4}$).

Aussi est-il intéressant d'observer l'évolution du taux de compression moyen sur les blocs admissibles pour différentes valeurs de m_T, comme représenté en figure 3.2.3.

Pour $\varepsilon_{\text{HCA}} = 10^{-4}$, nous pouvons constater que le taux de compression converge vers un maximum atteint dès que $m_T \geqslant 4$. Cette valeur seuil de m_T est la même que celle obtenue par observation de l'erreur relative sur le noyau en figure 3.2.2. Ceci semble indiquer que malgré une quantité importante de points de Tchebychev dans les cas $m_T = 5$ et $m_T = 6$, l'ACA$_\gamma$ arrive à convergence pour un rang similaire à celui obtenu pour $m_T = 4$.

Règle quant au choix de m_T. Dans [9], Steffen Börm règle les valeurs ε_{HCA} et m_T conjointement, sans toutefois préciser de règle d'accord entre ces deux paramètres.

L'étude menée ici semble montrer que pour $\varepsilon_{\text{HCA}} = 10^{-4}$, il est suffisant de poser $m_T = 4$ et une valeur supérieure est inutile.

Dès à présent, de façon plus générale et sauf mention contraire, nous définirons conjointement ε_{HCA} et m_T sur le modèle de [9], c'est à dire tels que

$$\varepsilon_{\text{HCA}} = 10^{-m_T}. \tag{3.2.69}$$

Réglage de ε_{HCA}

La précision souhaitée ε_{HCA} intervient en premier lieu lors de l'étape de compression du noyau par ACA_γ.

Pour évaluer l'erreur générée sur le noyau en fonction du choix de ε_{HCA}, nous parcourons les blocs admissibles $\widetilde{Z}_{|b} \in \mathbb{C}^{I\times J}$ de $\widetilde{Z}$, sur lesquels nous réalisons l'ACA_γ à précision ε_{HCA} (en appliquant la règle (3.2.69)), puis nous en déduisons pour tout $\boldsymbol{x} \in I$ et $\boldsymbol{y} \in J$ la valeur $\gamma(\boldsymbol{x}, \boldsymbol{y})$ et sa version dégénérée $\widetilde{\gamma}(\boldsymbol{x}, \boldsymbol{y})$. Nous évaluons alors l'erreur entre ces deux valeurs pondérée sur la $\mathcal{H}$-matrice $\widetilde{Z}$ globale. Cette erreur relative est représentée en figure 3.2.4 pour différentes valeurs de ε_{HCA}.

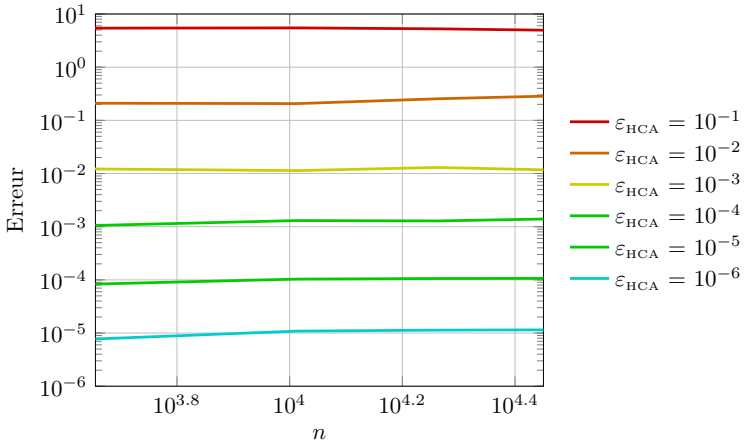


FIGURE 3.2.4 – Erreur relative induite sur le noyau par ACA pour différentes valeurs de ε_{HCA}.

Cette figure nous permet d'une part de valider la précision de l'ACA_γ sur le noyau, et d'autre part de vérifier que la règle d'accord (3.2.69) est compatible avec l'obtention de l'erreur désirée quelle que soit la valeur de ε_{HCA}.

Le paramètre ε_{HCA} intervient par ailleurs lors de la recompression par r_k-SVD, ainsi que durant le *coarsening* s'il y a lieu. Comme nous l'avons montré au chapitre 2 (parties 2.1.2 et 2.4.2), l'erreur induite par ses opérations purement algébriques respecte alors la précision prescrite. De ce fait, nous pouvons en déduire qu'en cas de *coarsening*, il semble judicieux de poser

$$\varepsilon_{coars.} \coloneqq \varepsilon_{\text{HCA}}. \tag{3.2.70}$$

Choix de la précision de compression selon la taille du problème

Vérifions maintenant l'influence de ε_{HCA} sur la précision par rapport à la solution analytique. Pour ceci, nous évaluerons l'erreur relative entre la solution numérique du système creux $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ assemblé par HCA, et la solution analytique $\boldsymbol{j}_{\text{réf.}}$. Nous utilisons les règles d'accord définies ci-avant en posant pour une valeur de m_T donnée, $\varepsilon_{\text{HCA}} = 10^{-m_T}$ et $\varepsilon_{coars.} = \varepsilon_{\text{HCA}}$.

Pour résoudre le système creux $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$, nous utiliserons le solveur itératif GMRES (*cf.* algorithme A.1) adapté au format des $\mathcal{H}$-matrices, en fixant $\varepsilon_{\text{GMRES}} = 10^{-12}$. L'erreur relative alors obtenue est présentée dans le cas de la sphère et comparée au cas de la compression ACA en figure 3.2.5.

Cette figure nous permet de constater que la précision par rapport à la solution analytique décroît en fonction du nombre de DDLs, jusqu'à atteindre un seuil à la fois proche et inférieur à ε_{HCA}. L'allure obtenue ici est en fait la même que celle obtenue dans le cas de

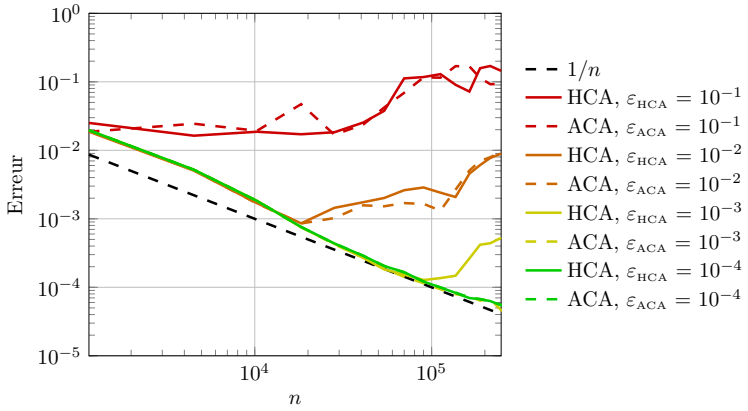


FIGURE 3.2.5 – Erreur relative par rapport à la solution analytique pour différentes valeurs de ε_{HCA} et comparaison avec l'ACA.

l'ACA pour des valeurs grossières de ε_{HCA} ; toutefois, on remarque que pour $\varepsilon_{\text{HCA}} = 10^{-3}$, la précision de l'ACA décroît plus que celle de l'HCA.

3.2.5 Étude des temps de calcul de l'HCA

Comme nous l'avons vu dans la partie précédente, chacun des paramètres de réglage de l'HCA influe sur la précision de la méthode à des étapes différentes de l'algorithme. De même, le choix des paramètres a un impact visible sur le temps de réalisation des étapes de la méthode. C'est ce que nous allons analyser dans cette partie.

Influences de m_T et ε_{HCA}

Nous avons déjà pu constater au chapitre 2 que le temps de réalisation de l'ACA était d'autant plus long que le paramètre ε_{ACA} est petit (*cf.* 2.4.1). On peut assez logiquement penser qu'il en sera de même pour des valeurs de m_T d'autant plus élevées. Pour le vérifier, fixons $\varepsilon_{\text{HCA}} = 10^{-4}$ et observons les différents temps moyens de réalisation de l'étape de compression par ACA_γ du noyau en fonction de m_T (*cf.* figure 3.2.6).

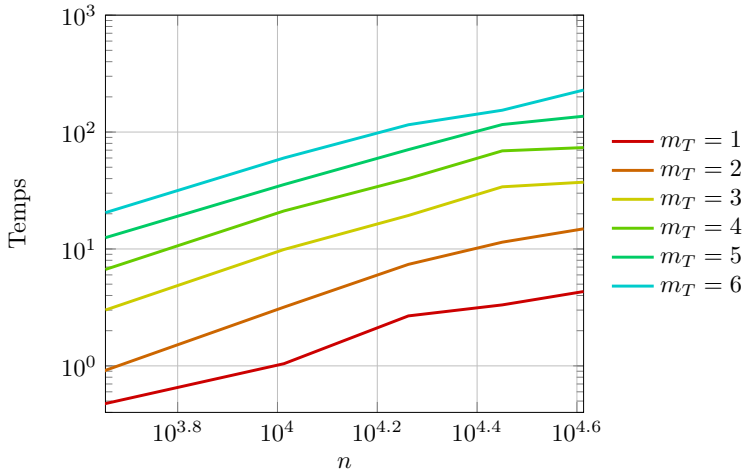


FIGURE 3.2.6 – Temps moyen de réalisation de l'étape de compression par ACA du noyau en fonction de m_T ($\varepsilon_{\text{HCA}} = 10^{-4}$).

Cette figure confirme l'intérêt, à précision obtenue égale, de choisir la valeur de m_T la plus faible possible, et finit donc de justifier la règle d'accord entre ε_{HCA} et m_T définie en (3.2.69). Le temps consacré à l'ACA dans la compression HCA correspond toutefois à une faible proportion du temps total de compression, comme nous le verrons dans la suite de cette section.

Influence de l'intégration numérique

Le choix du nombre de points de quadrature de Gauss-Legendre utilisés pour l'intégration numérique n'influe temporellement que sur l'étape de l'HCA durant laquelle sont réalisées les différentes intégrations. Si l'ACA avait *a priori* une dépendance quadratique en n_{Gauss}, ce n'est pas le cas pour l'HCA. En effet, les intégrales doubles de la BEM sont ici séparées en intégrales simples, ce qui laisse présager une dépendance linéaire en n_{Gauss} du temps de réalisation de cette étape. Nous pouvons le vérifier pour quelques exemples dans le tableau 3.2.1.

		n_{Gauss}		
n		1	3	7
4521	t (s)	12,628	41,468	93,992
	t/n_{Gauss} (s)	12,628	13,823	13,427
18 294	t (s)	98,508	300,116	650,776
	t/n_{Gauss} (s)	98,508	100,039	92,968
69 930	t (s)	434,092	1223,888	2670,956
	t/n_{Gauss} (s)	434,092	407,963	381,565

TABLEAU 3.2.1 – Durée des intégrations dans l'assemblage par HCA en fonction de n_{Gauss}.

Ce tableau nous permet de vérifier le coût linéaire en n_{Gauss} de la partie intégration de l'HCA. Cette dépendance est à comparer avec celle, quasi quadratique, observée dans le cas de l'ACA (*cf.* tableau 3.1.1).

Répartition des temps sur les différentes étapes de l'HCA

Nous savons à présent que le temps d'assemblage par HCA dépend à la fois de m_T, de ε_{HCA} et de n_{Gauss}, et les parties précédentes nous ont permis de déterminer dans quelles étapes de l'algorithme HCA ces paramètres avaient le plus de poids.

Nous n'avons toutefois pas parlé de l'étape de multiplication matricielle. Si le bloc $\widetilde{Z}_{|b}$ est de taille $m \times n$, les multiplications effectuées imposent un nombre d'opérations total proportionnel à $(m+n)k^2$, k désignant le rang obtenu suite à la compression du noyau de Green par ACA_γ. Ainsi la durée de l'étape comprenant les multiplications est indirectement dépendante de m_T et de ε_{HCA}.

Finalement, nous pouvons évaluer dans quelques cas la part de chacune de ces étapes (*cf.* 3.2.2).

TABLEAU 3.2.2 – Proportion des durées des différentes étapes de l'HCA ($\varepsilon_{\text{HCA}} = 10^{-4}$).

Ce tableau nous permet tout d'abord de constater que la durée des intégrations occupe la plus grande part du temps de compression par HCA, tandis que l'étape constituée de l'ACA$_\gamma$ et de la décomposition LU occupe une faible part du temps de compression total, de même que le temps des multiplications matricielles. En revanche nous pouvons également constater que la dernière étape, consistant en une recompression par r_k-SVD, occupe une part importante de la compression totale. Ceci s'explique par le fait que si l'ACA$_\gamma$ nous donne une approximation de rang k du noyau, l'EFIE étant la somme de deux intégrales dont une est vectorielle et l'autre scalaire, le rang en sortie de l'HCA est de $4k$. La part de la r_k-SVD est d'ailleurs bien plus faible dans le cas de l'HCA appliquée au cylindre infini (dimension 2).

Quoi qu'il en soit, la part importante de l'étape comportant les intégrations numériques nous permet d'affirmer que l'HCA devient avantageuse sur l'ACA dès lors que n_{Gauss} est supérieur à 1. En effet, considérant le réglage $n_{\text{Gauss}} = 3$ justifié en section 1.4.2, l'ACA est réalisée dans un temps proportionnel à $n_{\text{Gauss}}^2 = 9$, contre $2n_{\text{Gauss}} = 6$ pour l'étape d'intégration de l'HCA. Dans la partie 4.2.3, nous comparerons les performances temporelles de ces deux méthodes sur des cas de surfaces rugueuses.

Cette étude de l'HCA nous a donc permis dans un premier temps de développer cette méthode afin de l'adapter au cas de l'EFIE. Son usage est avantageux sur celui de l'ACA car son temps total de réalisation est moindre pour $n_{\text{Gauss}} = 3$, tout en ayant une précision comparable à celle de l'ACA pour les cas canoniques. Par ailleurs, le fait que l'HCA ne repose pas sur une estimation de l'erreur en fait une méthode de compression *a priori* plus robuste que l'ACA.

3.3 Résolution d'un système linéaire hiérarchique

Dans cette partie, nous verrons comment adapter plusieurs méthodes de résolution au format $\mathcal{H}$-matrice. Il s'agira de résoudre l'équation

$$\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}, \tag{3.3.1}$$

dans laquelle $\widetilde{Z}$ désigne l'écriture hiérarchique de la matrice impédance Z de l'objet diffractant, $\boldsymbol{e}$ le second membre dépendant du champ incident $\boldsymbol{E}^{\text{inc}}$ et $\boldsymbol{j}$ le vecteur des courants à déterminer.

Comme utilisé en partie 3.1.4, le GMRES s'adapte aisément au format hiérarchique en remplaçant les produits matrice-vecteur par leur pendant hiérarchique. Nous ne développerons donc pas d'avantage ce solveur. Toutefois l'algorithme du GMRES classique est disponible en annexe A.2.

3.3.1 Solveur $\mathcal{H}$-LU

Influence de ε_{LU}

Erreur et résidu. Pour juger de l'influence de la précision ε_{LU} sur l'erreur commise lors de la résolution $\mathcal{H}$-LU, il convient de résoudre avec cette méthode un système creux $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$, dans lequel $\boldsymbol{e}$ est connu et généré par multiplication de $\widetilde{Z}$ par un vecteur $\boldsymbol{j}_{\text{réf.}}$ aléatoire. Nous pouvons alors évaluer l'erreur relative entre $\boldsymbol{j}$ et $\boldsymbol{j}_{\text{réf.}}$ et la tracer pour différentes valeurs de ε_{LU} (*cf.* figure 3.3.1).

Cette figure nous permet de constater que l'erreur commise par décomposition $\mathcal{H}$-LU augmente à mesure que le nombre de DDLs augmente. Ce phénomène est d'autant plus flagrant que la valeur de ε_{LU} est grossière. Ceci est probablement une traduction du

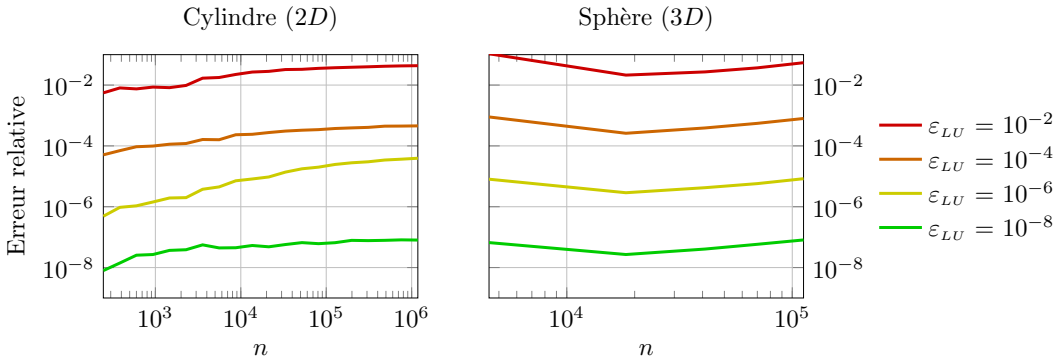


FIGURE 3.3.1 – Erreur relative induite par la résolution $\mathcal{H}$-LU pour différentes valeurs de ε_{LU}.

conditionnement de la matrice traitée, qui se dégrade avec la taille de la matrice. Aussi l'erreur est légèrement supérieure à la précision souhaitée

Nous pouvons également évaluer le résidu issu du solveur $\mathcal{H}$-LU. Pour cela nous évaluons cette fois l'erreur relative entre $\boldsymbol{e} = \widetilde{Z}\boldsymbol{j}_{\text{réf.}}$ et le produit $\widetilde{Z}\boldsymbol{j}$, ce qui nous permet d'obtenir la figure 3.3.2.

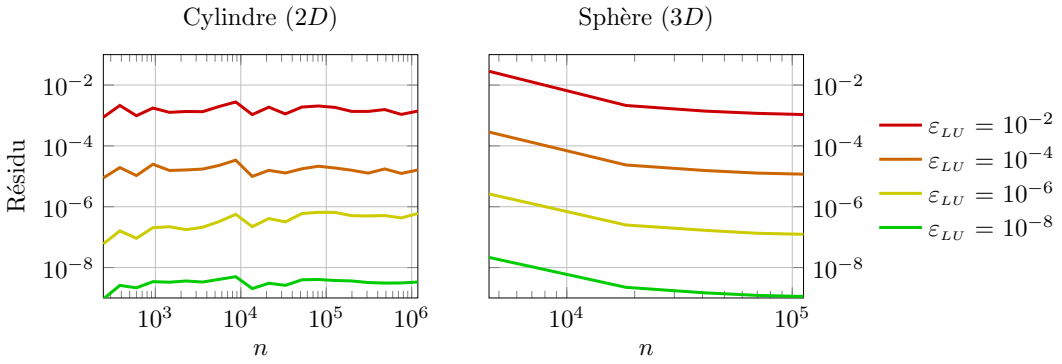


FIGURE 3.3.2 – Résidu induit par la résolution $\mathcal{H}$-LU pour différentes valeurs de ε_{LU}.

Nous pouvons constater que cette erreur reste de l'ordre de ε_{LU} quelle que soit sa valeur, sauf pour un faible nombre de DDLs dans le cas $3D$, où l'erreur est légèrement supérieure à la valeur prescrite de ε_{LU}.

Cette figure nous permet de valider la méthode de résolution $\mathcal{H}$-LU, même si la figure 3.3.1 met en évidence une perte de précision due au conditionnement de la matrice traitée.

Espace mémoire. À moins qu'il ne soit possible d'écraser la $\mathcal{H}$-matrice $\widetilde{Z}$, il est important de prendre en compte l'espace mémoire qu'occupe la $\mathcal{H}$-matrice contenant la décomposition $\mathcal{H}$-LU de $\widetilde{Z}$, ce que nous pouvons apprécier en figure 3.3.3.

Nous pouvons constater que le taux de compression de $\widetilde{LU}$ dépend de ε_{LU}. De ce fait nous pouvons nous attendre à ce que le temps de résolution dépende également de ε_{LU}, ce que nous allons vérifier dès à présent.

Temps de résolution. Le temps de résolution est avant tout fonction du taux de compression de la $\mathcal{H}$-matrice une fois décomposée. Elle est par ailleurs à mettre en comparaison avec le temps de décomposition $\mathcal{H}$-LU qui précède la résolution. Nous avons pu voir en

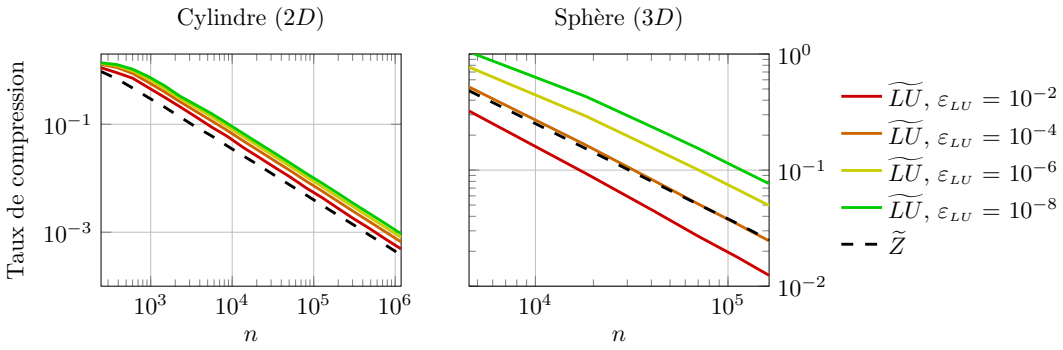


FIGURE 3.3.3 – Taux de compression de la $\mathcal{H}$-matrice contenant la décomposition $\mathcal{H}$-LU pour différentes valeurs de ε_{LU}.

figure 2.5.7 quelle était l'influence de ε_{LU} sur la durée de décomposition, ce que nous pourrons regarder plus en détails dans les cas décrits au tableau 3.3.1.

		ε_{LU}			
n		10^{-2}	10^{-4}	10^{-6}	10^{-8}
	Temps d'assemblage de $\widetilde{Z}$ (s)	85,144			
	Taux de compression de $\widetilde{Z}$ (s)	$4{,}9 \times 10^{-3}$			
Cylindre ($2D$)	Taux de compression de $\widetilde{LU}$	$6{,}6 \times 10^{-3}$	$8{,}8 \times 10^{-3}$	$1{,}1 \times 10^{-2}$	$1{,}2 \times 10^{-2}$
$n = 81209$	Temps de décomposition (s)	52,804	70,548	96,260	133,400
	Temps de résolution (s)	0,272	0,284	0,308	0,548
	Temps total (s)	**53,076**	**70,832**	**96,568**	**133,948**
	Temps d'assemblage de $\widetilde{Z}$ (s)	3114,332			
	Taux de compression de $\widetilde{Z}$ (s)	$8{,}0 \times 10^{-2}$			
Sphère ($3D$)	Taux de compression de $\widetilde{LU}$	$2{,}7 \times 10^{-2}$	$5{,}1 \times 10^{-2}$	$1{,}0 \times 10^{-1}$	$1{,}5 \times 10^{-1}$
$n = 69930$	Temps de décomposition (s)	8624,348	24 647,056	55 168,496	109 941,724
	Temps de résolution (s)	2,848	4,644	5,824	7,496
	Temps total (s)	**8627,196**	**24 651,700**	**55 174,320**	**109 948,220**

TABLEAU 3.3.1 – Temps total d'exécution du solveur $\mathcal{H}$-LU selon la précision ε_{LU} de la décomposition. Les $\mathcal{H}$-matrices sont assemblées par HCA avec $\varepsilon_{\text{HCA}} = 10^{-4}$

Ce tableau nous permet de vérifier que le temps de résolution $\mathcal{H}$-LU dépend du taux de compression de la $\mathcal{H}$-matrice $\widetilde{LU}$, mais aussi que le temps de résolution est négligeable devant le temps de décomposition. C'est donc ce dernier critère qu'il convient de prendre en compte.

La décomposition $\mathcal{H}$-LU permet donc de résoudre notre système hiérarchique de façon d'autant plus précise que la $\mathcal{H}$-matrice $\widetilde{Z}$ est bien conditionnée. Cette méthode de résolution est cependant coûteuse en temps, notamment à cause de la durée de la décomposition $\mathcal{H}$-LU.

Toutefois, lors d'une résolution par solveur direct $\mathcal{H}$-LU, la $\mathcal{H}$-matrice $\widetilde{Z}$ est écrasée par $\widetilde{LU}$. L'espace mémoire occupé par cette nouvelle $\mathcal{H}$-matrice est alors légèrement supérieur à celui occupé par $\widetilde{Z}$ avant qu'elle ne soit écrasée.

La décomposition $\mathcal{H}$-LU peut alors devenir intéressante dans le cadre d'un préconditionnement de solveur itératif, tel que GMRES. C'est ce que nous allons voir dans la

section suivante.

3.3.2 Préconditionnement des méthodes itératives

Il arrive qu'un solveur itératif ne parvienne pas à converger en un nombre raisonnable d'itérations, rendant la résolution très longue. Ce défaut de convergence est en général dû au conditionnement du système. Lorsque l'on considère un problème aux valeurs propres, il peut même arriver que le solveur itératif ne converge pas du tout s'il est proche d'une valeur de résonance. Dans les deux cas, il est possible d'améliorer le comportement de la convergence de la méthode itérative en y appliquant un *préconditionneur*. Le principe de tels préconditionneurs est rappelé en annexe A.2.

Adaptation au format $\mathcal{H}$-matrice

Soit une matrice $P \in \mathbb{C}^{n\times n}$. P est un préconditionneur à droite du système $Z\boldsymbol{j} = \boldsymbol{e}$ si P^{-1} est relativement proche de Z^{-1}, de sorte que le nouveau système

$$P^{-1}Z\boldsymbol{j} = P^{-1}\boldsymbol{e}, \tag{3.3.2}$$

soit proche d'une simple multiplication matrice-vecteur, c'est-à-dire

$$\boldsymbol{j} \approx P^{-1}\boldsymbol{e}. \tag{3.3.3}$$

Dans le cas des $\mathcal{H}$-matrices, le préconditionneur doit lui aussi être une $\mathcal{H}$-matrice $\widetilde{P}$, telle que

$$\boldsymbol{j} \approx \widetilde{P}^{-1}\boldsymbol{e}. \tag{3.3.4}$$

En pratique, nous ne construisons pas l'inverse de P, mais nous résolvons le système

$$\widetilde{P}\boldsymbol{j} \approx \boldsymbol{e}. \tag{3.3.5}$$

Nous savons résoudre ce système par une méthode directe : la résolution $\mathcal{H}$-LU. C'est de cette manière que nous allons donc introduire le préconditionnement dans le solveur itératif GMRES que nous avons précédemment adapté au format $\mathcal{H}$-matrice.

Une version recompressée par *coarsening* de $\widetilde{Z}$ s'impose alors comme candidat idéal dans le rôle du préconditionneur, puisqu'elle constitue une bonne approximation de $\widetilde{Z}$, de taux de compression plus faible que celui de $\widetilde{Z}$. Nous pouvons en effet rappeler que l'erreur relative résultant d'un *coarsening* à précision $\varepsilon_{coars.}$ est de l'ordre de cette précision, comme l'indiquait la figure 2.4.5 dans le chapitre précédent.

Dans la suite nous choisirons donc cette solution, en posant $\varepsilon_{\text{préc.}} = \varepsilon_{LU} = \varepsilon_{coars.}$.

GMRES préconditionné

L'ajout d'un préconditionneur $\widetilde{P}$ est intégré directement dans l'algorithme du solveur GMRES et ne conduira qu'à une résolution $\mathcal{H}$-LU supplémentaire à chaque itération. La décomposition $\mathcal{H}$-LU s'effectuant à une précision ε_{LU} près, ce solveur optimisé prend donc deux arguments supplémentaires que sont le préconditionneur $\widetilde{P}$ et la précision de décomposition $\mathcal{H}$-LU de ce dernier, notée $\varepsilon_{\text{préc.}}$.

Comme nous avons pu le voir précédemment, le temps d'une décomposition $\mathcal{H}$-LU est très supérieur à celui d'une résolution $\mathcal{H}$-LU. On a donc, dès le départ,

$$\widetilde{P} = \widetilde{LU} \tag{3.3.6}$$

à la précision $\varepsilon_{\text{préc.}}$ près.

Une fois la décomposition $\mathcal{H}$-LU faite, il suffit de faire précéder, au sein de chaque itération, la multiplication $\mathcal{H}$-matrice-vecteur $\widetilde{Z}\boldsymbol{j} = \boldsymbol{j}'$ en ligne 10 de l'algorithme A.1 d'une étape de résolution $\mathcal{H}$-LU

$$\widetilde{LU}\boldsymbol{j} = \boldsymbol{w}_j. \tag{3.3.7}$$

Nous pouvons prévoir que le temps de chaque itération sera plus long dans la version préconditionnée du GMRES, puisqu'on ajoute une étape de résolution $\mathcal{H}$-LU. Cette durée supplémentaire dépendra du taux de compression de $\widetilde{P}$. De plus, les figures 2.5.7 et 2.5.8 nous rappellent que le temps de décomposition $\mathcal{H}$-LU, qui sera déterminant dans cette version préconditionnée du solveur itératif, dépend à la fois du taux de compression de la $\mathcal{H}$-matrice et du choix de ε_{LU}.

Il est également important de prendre en considération l'espace mémoire qu'occupe le préconditionneur $\widetilde{P}$. En effet, la taille de cette $\mathcal{H}$-matrice étant la même que celle de $\widetilde{Z}$, plus le taux de compression de $\widetilde{P}$ est avantageux, moins la méthode itérative n'aura d'influence sur le coût mémoire total de la résolution. En particulier, la figure 2.4.4 nous rappelle dans quelle mesure l'espace mémoire occupé par une $\mathcal{H}$-matrice dépend de la précision $\varepsilon_{coars.}$ de sa recompression.

Ceci nous donne des indications sur les choix possibles de préconditionneurs et de précision $\varepsilon_{\text{préc.}}$.

Validation de l'outil de préconditionnement

Le préconditionnement $\mathcal{H}$-LU précédemment décrit dépend donc d'un nouveau paramètre $\varepsilon_{\text{préc.}}$. Étudions le comportement du préconditionneur $\mathcal{H}$-LU sur le solveur GMRES. Il peut être intéressant de comparer les performances de ce GMRES préconditionné sur un exemple précis. Prenons le cas d'une sphère ($3D$) discrétisée avec $n = 112206$ DDLs, assemblée par ACA avec $\varepsilon_{\text{ACA}} = 10^{-4}$, et recompressée par *coarsening* avec $\varepsilon_{coars.} = 10^{-4}$. Nous pouvons alors évaluer les performances de la méthode GMRES préconditionnée, comme consignées dans le tableau 3.3.2.

		$\varepsilon_{\text{préc.}}$		
	Sans prec.	10^{-2}	10^{-3}	10^{-4}
Temps d'assemblage (s)	11 959,340			
Temps de recompression (s)	1903,452			
Temps de *coarsening* à $\varepsilon_{\text{préc.}}$ (s)	$\emptyset$	419,375	546,244	$\emptyset$
Temps de décomposition $\mathcal{H}$-LU (s)	$\emptyset$	3071,464	7229,172	16 643,892
Temps de GMRES (s)	4544,476	39,576	23,656	23,804
Temps total (s)	18 414,560	17 400,500	21 669,156	30 996,404
Nombre d'itérations	976	4	2	1
Temps par itération (s)	4,656	9,894	11,828	23,804
Stockage de $\widetilde{Z}$ (Go)	6,211			
Stockage de $\widetilde{P}$ (Go)	$\emptyset$	2,885	4,519	6,561
Coût mémoire total (Go)	6,211	9,096	10,730	12,772
Erreur relative	$2{,}00 \times 10^{-2}$	$2{,}21 \times 10^{-3}$	$4{,}90 \times 10^{-4}$	$9{,}83 \times 10^{-4}$

TABLEAU 3.3.2 – Influence de $\varepsilon_{\text{préc.}}$ sur la résolution GMRES ($n = 112\,206$).

Ce tableau, dense en informations, mérite un certain nombre de commentaires.

En premier lieu, l'erreur relative calculée ne répond pas toujours aux exigences fixées par la tolérance $\varepsilon_{\text{GMRES}}$. En particulier, dans le cas $\varepsilon_{\text{préc.}} = 10^{-1}$ (non consigné dans le tableau), le préconditionnement ne permet pas de converger vers une solution acceptable, le résidu restant alors très supérieur à la valeur visée quelle que soit le nombre d'itérations effectuées.

Nous pouvons également constater que le temps de résolution dépend en immense majorité du temps nécessaire à la décomposition $\mathcal{H}$-LU. Le temps de réalisation des itérations est en effet négligeable devant la durée de décomposition $\mathcal{H}$-LU. Nous pouvons également noter que plus $\varepsilon_{\text{préc.}}$ est précis, plus la durée de décomposition $\mathcal{H}$-LU est longue. Il en est de même pour la durée d'une itération de GMRES. Ceci s'explique par le taux de compression de $\widetilde{P}$, qui est d'autant plus grand que $\varepsilon_{\text{préc.}}$ est fin. En revanche, plus $\varepsilon_{\text{préc.}}$ est faible, plus le préconditionneur est algébriquement proche de l'inverse de la $\mathcal{H}$-matrice $\widetilde{Z}$ et plus le nombre d'itérations est réduit. En particulier, une seule itération est nécessaire à atteindre la convergence pour $\varepsilon_{\text{préc.}} = 1 \times 10^{-4}$. Ce cas correspond en fait à une résolution directe par solveur $\mathcal{H}$-LU et n'est pas la solution la plus raisonnable du fait du temps nécessaire à effectuer la décomposition $\mathcal{H}$-LU à une telle précision.

Ainsi, bien que le temps individuel d'une itération soit plus important pour des valeurs de $\varepsilon_{\text{préc.}}$ plus faibles, ce phénomène est compensé par le nombre d'itérations, très réduit. Notons par ailleurs que le temps d'une itération de GMRES sans préconditionnement est plus faible que dans le cas préconditionné, quelle que soit la valeur de $\varepsilon_{\text{préc.}}$, mais que dans le cas du GMRES classique le nombre d'itérations est beaucoup plus important, compensant cet effet.

En ce qui concerne les coûts mémoires, nous pouvons voir que la version préconditionnée de la méthode itérative est d'autant moins intéressante que la précision $\varepsilon_{\text{préc.}}$ est grossière. En particulier, le préconditionnement peut devenir rédhibitoire dans le cas où l'espace mémoire disponible est réduit.

Pour conclure, en terme de temps, le solveur GMRES sans préconditionneur semble la solution la plus efficace. Cependant, un préconditionnement à $\varepsilon_{\text{préc.}} = 10^{-2}$ permet d'atteindre une solution plus précise en un temps similaire et en un nombre d'itérations beaucoup plus faible.

3.3.3 Cas résolution avec plusieurs seconds membres

Plaçons nous à présent dans le cas de la résolution d'un système comportant un nombre p de seconds membres. Pour rappel, ce calcul nécessite de résoudre le système

$$ZI = V, \tag{3.3.8}$$

où $Z \in \mathbb{C}^{n\times n}$ est la matrice impédance dont nous avons l'habitude, $V \in \mathbb{C}^{n\times p}$ la matrice dont les colonnes représentent les seconds membres $\boldsymbol{e}_i$ avec $i \in [\![1, p]\!]$ et $I \in \mathbb{C}^{n\times p}$ la matrice des p vecteurs solutions $\boldsymbol{j}_i$ avec $i \in [\![1, p]\!]$ à déterminer. Nous verrons dans cette partie que selon la méthode de résolution utilisée, cela peut impliquer un temps de résolution multiplié par le nombre de seconds membres traités.

Adaptation des méthodes de résolution

GMRES. Dans le cas de cette méthode itérative, nous considérons chaque second membre séparément. Ainsi, pour p seconds membres nous effectuerons p résolutions itératives différentes.

Notons $t_{\text{ité.}}$ le temps moyen d'une itération de GMRES. Ce temps dépendant en grande partie du temps de multiplication matrice-vecteur qui dépend lui-même de la matrice Z (ou de la $\mathcal{H}$-matrice $\widetilde{Z}$ et de son taux de compression), chacune des itérations prendra un temps t proche de $t_{\text{ité.}}$. Notons également $n^-_{\text{ité.}}$ (resp. $n^+_{\text{ité.}}$) le plus petit (resp. le plus grand) nombre d'itérations nécessaires à la convergence pour l'un des seconds membres. Le temps total de la méthode de résolution T_{GMRES} vérifie

$$pn^-_{\text{ité.}} t_{\text{ité.}} \leqslant T_{\text{GMRES}} \leqslant pn^+_{\text{ité.}} t_{\text{ité.}}. \tag{3.3.9}$$

Si de plus $n^-_{\text{ité.}}$ et $n^+_{\text{ité.}}$ sont proches et élevés, la minoration de ce temps par $pn^-_{\text{ité.}} t_{\text{ité.}}$ peut très vite devenir problématique.

Comme dans le cas de la résolution d'un système à un second membre, l'ajout d'un préconditionneur réduira le nombre d'itérations tout en augmentant la durée d'une itération unique. Le temps total T_{GMRESp} de cette méthode vérifie le même encadrement que le GMRES.

Solveur direct $\mathcal{H}$-LU. Dans le cas de cette méthode directe, le temps de décomposition $\mathcal{H}$-LU est le temps le plus important à prendre en compte. Or le fait de devoir traiter plusieurs seconds membres ne nécessite qu'une seule décomposition $\mathcal{H}$-LU. De ce fait seule l'étape de résolution sera multipliée par p, si bien que le temps total de la méthode $T_{LU\text{ sol.}}$ vérifie

$$T_{LU\text{ sol.}} = t_{\mathcal{H}\text{-}LU} + pt_{LU\text{ sol.}}. \tag{3.3.10}$$

Cette durée de résolution ne semble pas, *a priori*, trop élevée. Toutefois cette méthode reste sujette à des problèmes de précision dûs à un éventuel mauvais conditionnement de la matrice Z.

Comparaison des trois méthodes

Plaçons nous à présent dans le cas particulier du calcul de la SER monostatique dans le cas de la sphère. Pour rappel, ce calcul nécessite de résoudre le système $\widetilde{Z}I = V$, où $\widetilde{Z}$ est la matrice impédance dont nous avons l'habitude, V la matrice dont les colonnes représentent les vecteurs $e(\phi)$ avec $\phi \in [0, 360[$ désignant l'angle d'incidences de $\boldsymbol{E}^{\text{inc}}$, et I la matrice de 360 colonnes à déterminer.

Nous assemblons la $\mathcal{H}$-matrice $\widetilde{Z}$ correspondant à une sphère parfaitement conductrice maillée avec 18 294 DDLs, en utilisant la méthode HCA avec $\varepsilon_{\text{HCA}} = 10^{-4}$ et $m_T = 4$. Nous appliquons également à la $\mathcal{H}$-matrice $\widetilde{Z}$ une recompression par *coarsening*, avec $\varepsilon_{coars.} = 10^{-4}$. Nous résolvons alors le système de trois manières différentes :

- par GMRES, avec $\varepsilon_{\text{GMRES}} = 10^{-4}$,
- par solveur direct $\mathcal{H}$-LU avec $\varepsilon_{LU} = 10^{-4}$,
- par GMRES préconditionné, avec $\varepsilon_{\text{GMRES}} = 10^{-4}$ et $\varepsilon_{\text{préc.}} \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.

Cela nous permet d'obtenir le tableau 3.3.3.

Ce tableau souligne les avantages et inconvénients de chacune des méthodes dans le cas d'une résolution multi second membre.

Tout d'abord en terme de temps, on constate que le GMRES est la méthode de résolution la plus longue, tandis que le solveur $\mathcal{H}$-LU direct est la technique la plus rapide. Le préconditionnement accélère effectivement le GMRES en fonction de la précision de $\varepsilon_{\text{préc.}}$ choisie. Plus $\varepsilon_{\text{préc.}}$ est fin, plus le temps par itération est élevé mais plus le nombre d'itérations est faible. Le temps de décomposition $\mathcal{H}$-LU intervient également dans le temps global de résolution, mais comme cette décomposition n'a lieu qu'une fois, il semble

	GMRES	$\mathcal{H}$-LU	GMRES précond.		
			10^{-1}	10^{-2}	10^{-3}
Temps d'assemblage (s)			854,912		
Temps de recompression (s)			221,264		
Temps de *coarsening* à $\varepsilon_{\text{préc.}}$ (s)	∅	∅	36,152	62,280	71,884
Temps de décomposition $\mathcal{H}$-LU (s)	∅	1622,080	124,800	353,768	852,980
Temps de GMRES (s)	109 164,62	∅	16 370,384	1684,504	1394,260
Temps de résolution $\mathcal{H}$-LU (s)	∅	226,808	∅	∅	∅
Temps total (s)	**109 164,62**	**2048,888**	**16 532,192**	**2101,556**	**2320,428**
Nombre d'itérations minimal	664	∅	48	3	2
Nombre d'itérations maximal	678	∅	53	4	2
Nombre d'itérations total	241 630	∅	17 993	1281	720
Nombre d'itérations moyen	671,19	∅	49,981	3,56	2
Temps moyen par itération (s)	0,452	∅	0,910	1,315	1,937
Stockage de $\widetilde{Z}$ (Mo)			819,51		
Stockage de $\widetilde{LU}$ (Mo)	∅	879,25	217,16	384,82	602,94
Coût mémoire total (Mo)	854,912	879,25	1036,7	1204,3	1422,5

Tableau 3.3.3 – Différentes méthodes de résolution pour le calcul de la SER monostatique.

plus avantageux de faire une décomposition $\mathcal{H}$-LU avec ε_{LU} fin dans un cas multi second membre.

En terme de stockage, la solution la plus avantageuse reste le GMRES, qui ne nécessite que le stockage de la $\mathcal{H}$-matrice $\widetilde{Z}$. Le solveur direct $\mathcal{H}$-LU est la méthode la moins avantageuse de ce point de vue. Dans le cas du GMRES préconditionné, plus $\varepsilon_{\text{préc.}}$ est fin, plus la $\mathcal{H}$-matrice $\widetilde{LU}$ est volumineuse.

Nous avons donc à notre disposition plusieurs méthodes de résolution, chacune ayant ses points forts et ses points faibles. Selon que notre objectif est d'occuper peu d'espace mémoire, d'effectuer une résolution rapide ou une résolution qui soit affranchie des éventuels problèmes de conditionnement de la matrice, il est possible de trouver un compromis parmi les méthodes que nous avons présentées dans cette partie.

Conclusion du chapitre

Dans ce chapitre, nous avons dans un premier temps validé la méthode de compression ACA, puis nous avons présenté et validé la méthode HCA, avant de nous attacher à adapter des méthodes de résolution connues au format $\mathcal{H}$-matrice.

Si l'ACA a fait ses preuves, tant en terme de précision que de complexité algorithmique, cette méthode de compression présente l'inconvénient de dépendre d'une estimation de l'erreur qui peut s'avérer erronée dans certains cas. L'HCA pallie à ces problèmes, d'une part en appliquant l'approximation en croix uniquement sur le noyau, et d'autre part en séparant les intégrales doubles en intégrales simples. Cette méthode de compression est donc à la fois plus robuste et plus rapide que l'ACA, tout en offrant une précision similaire à cette dernière dans les cas où l'ACA est robuste.

Nous pouvons résoudre le système $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ grâce au solveur $\mathcal{H}$-LU direct, dont le temps

de réalisation n'est pas soumis à l'inconnue du nombre d'itérations, mais dépend surtout de la décomposition $\mathcal{H}$-LU (son temps de réalisation ainsi que le taux de compression de la matrice résultante). Cependant ce solveur est gourmand en mémoire et très sensible au conditionnement du système à résoudre. Le GMRES peut également s'adapter au format $\mathcal{H}$-matrice mais est également sensible au conditionnement du système, ce qui peut engendrer l'augmentation du nombre d'itérations nécessaires pour converger vers la solution. Le préconditionnement $\mathcal{H}$-LU permet de palier à ce problème, en aidant le GMRES à converger en un nombre réduit d'itérations.

Les outils étudiés sont à présent validés, leur comportement étant conforme à ce que l'on pouvait en attendre sur les exemples canoniques que constituent le cylindre infini et la sphère. Reste maintenant à déterminer si ces résultats sont reproductibles sur des exemples d'application plus concrets et plus complexes. C'est ce que nous testerons dans le chapitre 4.

Chapitre 4

Application des $\mathcal{H}$-matrices à des exemples non-canoniques

Dans les chapitres précédents, nous avons pu voir que le format $\mathcal{H}$-matrice permet d'assembler une approximation de la matrice impédance de façon rapide et compressée, et à précision contrôlée. Nous avons pu mettre à l'épreuve les méthodes développées sur des cas de références qui nous ont été utiles pour en valider la précision. Cependant, ces cas ne sont pas à l'image des applications concrètes que l'on peut rencontrer dans l'industrie.

En particulier, dans le domaine aéronautique, il est fréquent de devoir effectuer des calculs de SER sur des objets électriquement grands. Nous traiterons donc le cas d'un avion maillé par la des éléments finis de frontière de type RWG [38], ce qui sera l'occasion pour nous d'appliquer quelques résultats de calcul parallèle.

De même, les surfaces rugueuses constituent un domaine d'étude foisonnant et des cas d'application intéressants. En effet, leur géométrie complexe permet de tester les limites des méthodes utilisées. Ces surfaces nécessitent un maillage fin pour être fidèlement représentées, et les matrices impédances résultantes, en plus d'être denses, sont généralement mal conditionnées. Ceci rend le système linéaire $Z\boldsymbol{j} = \boldsymbol{e}$ difficile à résoudre à précision raisonnable. Nous appliquerons donc le format $\mathcal{H}$-matrice à des surfaces de Weierstrass, cas particuliers de surfaces rugueuses, que nous étudierons d'abord en dimension 2 avant de traiter quelques exemples en dimension 3.

Pour finir, nous verrons que les matrices hiérarchiques et les méthodes de compression précédemment étudiées peuvent s'appliquer à des problèmes autres que la diffraction d'une onde. C'est le cas des PCFs, dont la théorie repose sur les équations de Maxwell et les équations intégrales, et sur lesquelles nous appliquerons certaines des méthodes précédemment étudiées.

Mots clés

Structures électriquement grandes, calcul parallèle, surfaces de Weierstrass, PCF

4.1 Calcul de la diffraction par un avion sur architecture à mémoire partagée

L'aéronautique est un domaine d'application courant en modélisation électromagnétique. En particulier, il est fréquent de calculer la SER bistatique d'un avion dans le cadre de la télédétection radar. Dans cette section, nous utiliserons les $\mathcal{H}$-matrices pour effectuer un calcul de SER bistatique d'un avion comportant un grand nombre de DDLs. Nous en profiterons pour appliquer au solveur itératif (GMRES), et en particulier à la multiplication $\mathcal{H}$-matrice-vecteur, quelques principes de calcul parallèle sur architecture à mémoire partagée (*cf.* annexe C).

4.1.1 $\mathcal{H}$-matrices et calcul parallèle

Dans le format $\mathcal{H}$-matrice, chaque bloc matriciel est considéré indépendamment des autres, qu'il soit ou non compressé. Nous rappelons par ailleurs que la grande majorité des algorithmes traitant les $\mathcal{H}$-matrices sont récursifs ; chaque bloc est traité s'il est une feuille de l'arbre quaternaire contenant la structure hiérarchique représentant la $\mathcal{H}$-matrice. En cela, certaines des opérations liées aux $\mathcal{H}$-matrices semblent pouvoir être parallélisées de façon directe et portable. Ce devrait être plus particulièrement le cas pour l'algorithme d'assemblage d'une $\mathcal{H}$-matrice, dans lequel chaque bloc est construit indépendamment des autres. De tels travaux ont d'ailleurs déjà été menés dans le cadre de la thèse de Benoît Lizé [39]. Cet aspect de l'optimisation logicielle n'étant pas au cœur de nos travaux, les tâches de parallélisation effectuées ici en resteront au stade de l'ébauche.

Considérons une routine OPERATION quelconque traitant chacun des blocs matriciels indépendamment des autres dans une $\mathcal{H}$-matrice. En première approche, la mise en parallèle de cette routine consiste à créer une liste des feuilles $\mathcal{L}(b)$ de la $\mathcal{H}$-matrice représentée par son *block cluster tree* b. Cette liste sera parcourue de façon plus simple par les différents processeurs que si les blocs étaient maintenus dans une structure arborescente. Une fois cette liste définie, il suffit alors de faire une boucle sur ses éléments.

Cette boucle parallèle répartit les blocs à traiter de façon à ce que les processeurs soient continuellement occupés. Dès qu'un processeur a terminé de traiter un bloc, il peut commencer à gérer le prochain bloc non encore traité par les autres processeurs, et ainsi de suite, jusqu'à ce que tous les blocs soient traités.

Nous pouvons généraliser cette approche comme indiqué dans l'algorithme 4.1.1.

Algorithme 4.1.1 Parallélisation d'une opération séquentielle sur une $\mathcal{H}$-matrice.

```
1: procedure PARALLEL_OPERATION(L(b),ε)
2:     for b' ∈ L(b) do                ▷ Boucle parallèle répartie sur tous les processeurs.
3:         call OPERATION(b', ε)
4:     end for
5: end procedure
```

Cependant cette simplicité de mise en œuvre ne se reflète pas nécessairement sur les résultats obtenus. En particulier, il est difficile d'évaluer le temps de communication entre les processeurs selon l'opération traitée. Ce temps sera pourtant le premier responsable d'une perte en efficacité d'un programme parallélisé sur architecture partagée.

Dans un premier temps nous appliquerons l'algorithme 4.1.1 au cas de l'assemblage, et nous discuterons de possibles voies d'amélioration des résultats obtenus. Nous nous pencherons ensuite sur le cas, *a priori* plus complexe, de la multiplication $\mathcal{H}$-matrice-vecteur, pour lequel certaines données de sorties dépendent les unes des autres.

Parallélisation de l'assemblage

Dans une $\mathcal{H}$-matrice, chaque bloc matriciel est absolument indépendant des autres blocs de la $\mathcal{H}$-matrice. L'algorithme 4.1.1 semble donc tout indiqué pour la mise en parallèle de cette opération. Toutefois, en appliquant à l'assemblage le principe de l'algorithme 4.1.1, les résultats obtenus ont été décevants.

Pour l'illustrer, prenons le cas d'une sphère maillée avec $n = 69930$ DDLs et assemblons la $\mathcal{H}$-matrice $\widetilde{Z}$ d'une part récursivement (comme décrit dans l'algorithme 2.4.1) et d'autre part en parallélisant le processus selon l'algorithme 4.1.1.

On observe alors les temps d'assemblages et les efficacités correspondantes dans le tableau 4.1.1.

	Récursif	Parallèle				
		1	2	4	8	16
Temps (s)	2876,894	2314,717	2552,149	2326,132	2207,233	2615,876
Accélération ($a_{n_{\text{proc.}}} = \frac{t_1}{t_{n_{\text{proc.}}}}$)	$\emptyset$	1	1,3034	1,2429	1,1272	1,2368
Efficacité ($e_{n_{\text{proc.}}} = \frac{a_{n_{\text{proc.}}}}{n_{\text{proc.}}}$)	$\emptyset$	1	0,651 698	0,310 718	0,140 906	0,077 298

TABLEAU 4.1.1 – Durée et efficacité de la parallélisation de l'assemblage d'une $\mathcal{H}$-matrice en fonction du nombre de processeurs utilisés.

On rappelle que l'efficacité d'un code parallèle effectué sur un nombre $n_{\text{proc.}}$ de processeur se mesure par comparaison avec le même code effectué sur un seul processeur (ou *algorithme séquentiel*). Par définition de l'accélération et de l'efficacité d'une tâche parallèle (*cf.* annexe C), on considère que la parallélisation d'un code est optimale si $a = n_{\text{proc.}}$ et $e = 1$.

La donnée du temps d'assemblage récursif n'est précisée qu'à titre indicatif, et permet de vérifier, par comparaison avec le cas séquentiel ($n_{\text{proc.}} = 1$), que le parcours d'une liste de blocs est moins complexe et plus rapide que celui d'un *block cluster tree*.

Les résultats obtenus au tableau 4.1.1 sont donc peu satisfaisants. Ils sont probablement dus à des accès concurrents des différents processeurs aux données du maillage. En effet, le maillage est généré en amont de l'assemblage par un unique processeur, et son emplacement doit ensuite être atteint par tous les processeurs au moment de l'assemblage. Nous pouvons supposer que ceci provoque des concurrences dans les communications entre les divers processeurs, ce qui a pour conséquence d'allonger considérablement le temps d'assemblage d'un bloc donné à mesure que le nombre de processeurs augmente.

Pour remédier à ce problème, il conviendrait d'ordonnancer les opérations, et en particulier les accès mémoire, afin d'améliorer les performances de cet assemblage parallèle. Une autre possibilité serait de construire une copie du maillage locale à chaque processeur. Ces deux pistes d'améliorations sont en cours d'investigation.

Application au produit $\mathcal{H}$-matrice-vecteur

Le produit $\mathcal{H}$-matrice-vecteur est une opération légèrement plus complexe que l'assemblage et la somme arrondie. En effet, la donnée de sortie (le vecteur résultant) est modifiée à plusieurs reprises. Pour l'illustrer, reprenons la schématisation de cette opération dans le cas d'une $\mathcal{H}$-matrice à deux niveaux de hiérarchie (*cf.* figure 4.1.1). Nous utiliserons un symbole "prime" pour les objets correspondants au niveau 2 de la hiérarchie.

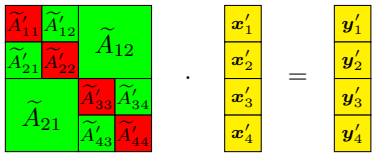


FIGURE 4.1.1 – Produit $\mathcal{H}$-matrice-vecteur pour une $\mathcal{H}$-matrice à deux niveaux de hiérarchie.

Cette opération est alors équivalente, au niveau 1 de hiérarchie, aux deux étapes

$$\boldsymbol{y}_1 \leftarrow \boldsymbol{y}_1 + \widetilde{A}_{12}\boldsymbol{x}_2, \tag{4.1.1}$$
$$\boldsymbol{y}_2 \leftarrow \boldsymbol{y}_2 + \widetilde{A}_{21}\boldsymbol{x}_1, \tag{4.1.2}$$

et au niveau 2,

$$\boldsymbol{y}'_1 \leftarrow \boldsymbol{y}'_1 + \widetilde{A}'_{11}\boldsymbol{x}'_1 + \widetilde{A}'_{12}\boldsymbol{x}'_2, \tag{4.1.3}$$
$$\boldsymbol{y}'_2 \leftarrow \boldsymbol{y}'_2 + \widetilde{A}'_{21}\boldsymbol{x}'_1 + \widetilde{A}'_{22}\boldsymbol{x}'_2, \tag{4.1.4}$$
$$\boldsymbol{y}'_3 \leftarrow \boldsymbol{y}'_3 + \widetilde{A}'_{33}\boldsymbol{x}'_3 + \widetilde{A}'_{34}\boldsymbol{x}'_4, \tag{4.1.5}$$
$$\boldsymbol{y}'_4 \leftarrow \boldsymbol{y}'_4 + \widetilde{A}'_{43}\boldsymbol{x}'_3 + \widetilde{A}'_{44}\boldsymbol{x}'_4. \tag{4.1.6}$$

On compte donc un total de six étapes, correspondant à dix produits $\mathcal{H}$-matrice-vecteur, chacun suivi d'une somme du vecteur résultant à une partie du vecteur de sortie. Ainsi le vecteur $\boldsymbol{y}_1$ par exemple sera modifié à plusieurs reprises, que ce soit au niveau 1 ou au niveau de ses sous-vecteurs $\boldsymbol{y}'_1$ et $\boldsymbol{y}'_2$, même si chacun des blocs de la $\mathcal{H}$-matrice ne sera considéré qu'une fois.

Cette situation peut alors créer des conflits si la somme sur une même partie du vecteur de sortie est effectuée simultanément par deux processeurs différents.

Il existe alors deux stratégies. La première consiste à utiliser une directive qui empêche deux sommes de s'effectuer simultanément (CRITICAL). Cette méthode a l'avantage d'être simple à mettre en place, conjointement avec l'algorithme 4.1.1. Cependant cette approche n'est pas la plus efficace, puisqu'à aucun moment deux sommes ne pourront se faire au même moment, et ce même si elles n'affectent pas la même partie du vecteur de sortie.

Il convient alors de traiter le problème sous un autre angle, en ordonnant les opérations de telle manière que toutes les opérations ayant un impact sur une partie donnée du vecteur de sortie soient traitées par le même processeur.

En pratique, cela implique de hiérarchiser la liste $\mathcal{L}(b)$ en sous-listes prenant en compte à la fois le niveau de hiérarchie auquel appartient chaque feuille, et la ligne du vecteur résultant affectée par cette feuille. Ainsi, on peut définir

- pour un niveau l donné, la liste de blocs

$$\mathcal{L}_l(b) = \{b' \in \mathcal{L}(b) \mid \operatorname{depth}(b') = l\}, \tag{4.1.7}$$

- pour un niveau l et un indice de ligne r donnés, la liste de blocs

$$\mathcal{L}_{l,r}(b) = \{b' \in \mathcal{L}_l(b) \mid r \text{ est la première ligne de } b'\}. \tag{4.1.8}$$

Ainsi $\mathcal{L}_l(b)$ correspond à la liste de tous les blocs situés au niveau l de la $\mathcal{H}$-matrice, et $\mathcal{L}_{l,r}(b) \subset \mathcal{L}_l(b)$ constitue la liste de tous les blocs du niveau l dont la première ligne est d'indice r.

La parallélisation du produit $\mathcal{H}$-matrice-vecteur a alors lieu au niveau de chaque ligne r de chaque niveau l, comme indiqué dans l'algorithme 4.1.2.

Algorithme 4.1.2 Parallélisation du produit $\mathcal{H}$-matrice-vecteur.

```
1: procedure PARALLEL_HMAT_VECT(Ã,ℒ(b),x,y)
2:     for l do                          ▷ Pour tous les niveaux de profondeur de hiérarchie.
3:         for r do      ▷ Pour toutes les lignes de départ des feuilles du niveau l (boucle parallèle).
4:             for b' ∈ ℒ_{l,r}(b) do
5:                 call HMAT_VECT(Ã_{|b'}, x_{|b'}, y')
6:                 y_{|b'} ← y_{|b'} + y'                ▷ Addition au vecteur résultant.
7:             end for
8:         end for
9:     end for
10: end procedure
```

Par cette organisation, nous nous assurons que chacune des parties du vecteur résultant ne peut être modifiée simultanément par deux processeurs. Néanmoins cette méthode présente le risque d'une perte d'efficacité due au phénomène de *latence*, puisqu'il est probable que tous les processeurs ne soient pas occupés au même moment, en particulier pour les niveaux de hiérarchisation les plus bas.

Pour évaluer les performances de cette parallélisation, nous avons appliqué la multiplication $\mathcal{H}$-matrice-vecteur parallélisée à une $\mathcal{H}$-matrice correspondant au problème de la sphère et appliquée pour différents nombres de processeurs. La figure 4.1.2 donne d'une part le temps de réalisation de 100 produits $\mathcal{H}$-matrice-vecteurs pour divers nombres de processeurs, et d'autre part l'efficacité associée.

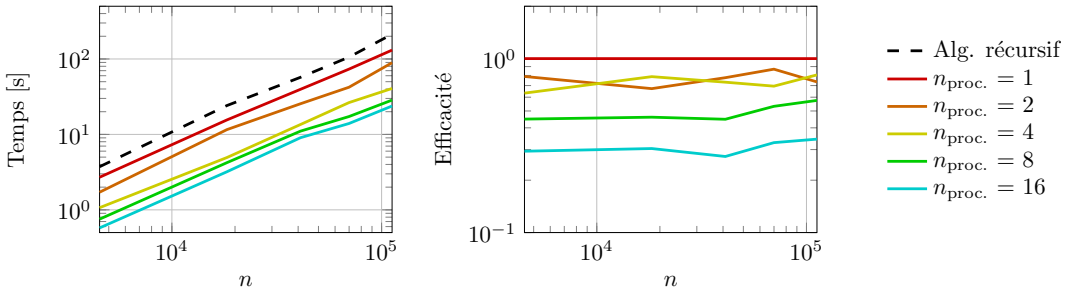


FIGURE 4.1.2 – Temps et efficacité de 100 produits $\mathcal{H}$-matrice-vecteur en fonction du nombre de processeurs utilisés.

Cette figure nous permet de constater une extensibilité correcte de la parallélisation pour le produit $\mathcal{H}$-matrice-vecteur. En effet, on remarque que plus il y a de processeurs utilisés, plus le temps de réalisation de cette opération est rapide. Par ailleurs, l'efficacité est relativement haute (entre 0,7 et 0,8) pour 2 et 4 processeurs. Elle est certes moins bonne pour 8 et 16 processeurs, mais demeure au dessus de 0,5 pour 8 processeurs. Enfin, nous pouvons noter que pour une géométrie donnée (ici la sphère de référence), le nombre de DDLs considérés ne modifie pas l'efficacité de l'algorithme parallèle.

Le produit $\mathcal{H}$-matrice-vecteur ainsi ordonnancé est donc un bon candidat à la parallélisation, ce qui nous permettra par la suite d'adapter le solveur itératif GMRES au calcul parallèle.

Pour conclure, si la mise en parallèle des opérations sur les $\mathcal{H}$-matrices peut paraître relativement aisée, il ne faut pas négliger les limitations du calcul parallèle en architecture

partagée. En particulier, le temps utilisé par la communication entre processeurs peut nuire grandement à l'efficacité de la parallélisation. Il est également important de se méfier des possibles modifications d'une donnée de sortie par différents processeurs en simultané.

En attendant de pouvoir effectuer les améliorations nécessaires à l'optimisation de l'assemblage parallèle, nous limiterons la parallélisation de nos opérations au produit $\mathcal{H}$-matrice-vecteur.

4.1.2 Résolution d'un problème électriquement grand

Afin d'éprouver le produit $\mathcal{H}$-matrice-vecteur parallèle, nous considérons le cas d'un avion, supposé parfaitement métallique, avec 77 190 DDLs. Cet objet est représenté en figure 4.1.3.

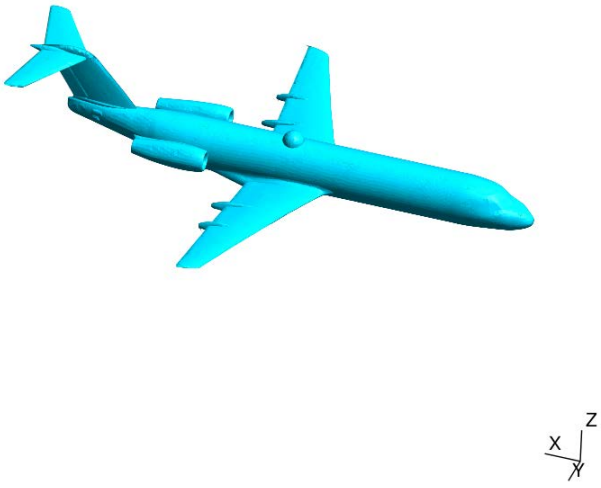


FIGURE 4.1.3 – Avion discrétisé en surface avec 77 190 DDLs.

Nous allons effectuer le calcul de la SER bistatique de cet avion à la fréquence de travail $f = 0{,}6\,\text{GHz}$, soit une longueur d'onde $\lambda \approx 0{,}5\,\text{m}$. Les statistiques du maillage sont alors les suivantes :

- longueur de l'arête la plus longue : $l_{\text{max.}} = 4{,}29 \times 10^{-2}\,\text{m} < \frac{\lambda}{10}$;
- longueur de l'arête la plus courte : $l_{\text{min.}} = 1{,}32 \times 10^{-4}\,\text{m}$;
- longueur moyenne des arêtes du maillage : $l_{\text{moy.}} = 9{,}85 \times 10^{-3}\,\text{m}$.

Pour ce faire, nous assemblons la matrice $\widetilde{Z}$ par HCA, avec $\varepsilon_{\text{HCA}} = 10^{-4}$. Nous résolvons ensuite le système $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ par GMRES ($\varepsilon_{\text{GMRES}} = 10^{-2}$) en remplaçant les produits $\mathcal{H}$-matrice-vecteur classiques par leur pendant parallèle (*cf.* algorithme 4.1.2). La figure 4.1.4 représente alors la SER bistatique obtenue dans le plan d'incidence $\phi = 0°$.

Le tableau 4.1.2 résume quant à lui les performances de la résolution, avec ou sans parallélisation.

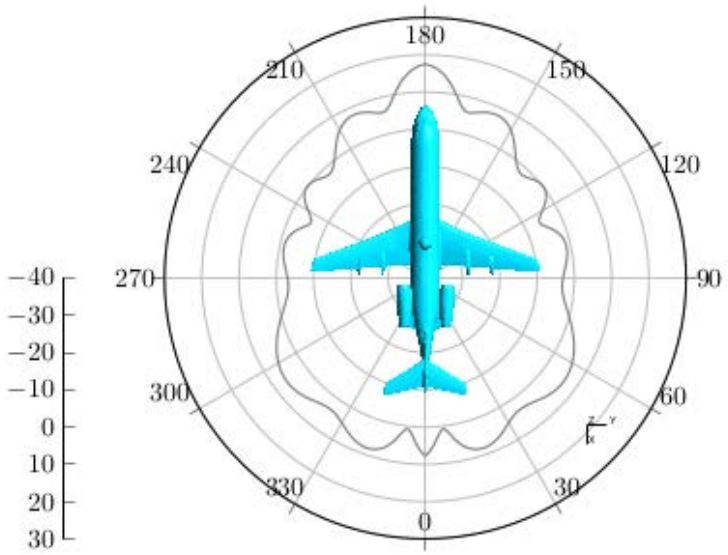


FIGURE 4.1.4 – SER bistatique de l'avion discrétisé en surface avec 77 190 DDLs ($\phi = 0°$).

	Récursif	Parallèle				
		1	2	4	8	16
Temps (s)	16 861,539	12 353,780	9146,622	6779,494	5724,984	5004,706
Accélération ($a_{n_{\text{proc.}}} = \frac{t_1}{t_{n_{\text{proc.}}}}$)	∅	1	1,3539	1,8267	2,1631	2,4744
Efficacité ($e_{n_{\text{proc.}}} = \frac{a_{n_{\text{proc.}}}}{n_{\text{proc.}}}$)	∅	1	0,670 96	0,456 66	0,270 39	0,154 65

TABLEAU 4.1.2 – Performances de la résolution itérative parallèle (cas de l'avion à 77 190 DDLs).

Les performances des résolutions parallèles sont moins bonnes que celles d'une multiplication $\mathcal{H}$-matrice-vecteur seule, ce qui est un résultat prévisible. En effet, bien que le solveur GMRES repose essentiellement sur des produits $\mathcal{H}$-matrice-vecteurs, il reste une partie non négligeable de cet algorithme qui n'est pas parallélisée. Notons toutefois que le temps de résolution le plus court est celui correspondant à une parallélisation sur 16 processeurs, même si la valeur de son efficacité est faible.

La parallélisation de certaines opérations de l'arithmétique des $\mathcal{H}$-matrices peut donc sembler intéressante et à première vue aisée à implémenter, notamment grâce à des directives OpenMP ; toutefois, le format $\mathcal{H}$-matrice n'est pas épargné par les limitations intrinsèques du calcul parallèle sur mémoire partagée. En particulier, il faut bien veiller à gérer les accès mémoire des processeurs mis en jeu afin qu'il ne se fassent pas concurrence, diminuant alors l'extensibilité de l'algorithme parallèle. L'ordonnancement des tâches doit donc être effectué, comme nous avons pu le tester sur le produit $\mathcal{H}$-matrice-vecteur qui nous a ensuite permis d'accélérer la résolution itérative d'un problème concret comportant un grand nombre de DDLs.

Pour des nombres de DDLs plus élevés, une résolution itérative n'est pas envisageable sans préconditionnement. Or la création du préconditionneur demeure la partie la plus coûteuse de la résolution. Il deviendra alors primordial, de parvenir à une parallélisation de la décomposition $\mathcal{H}$-LU afin de pouvoir réduire le temps global de la résolution itérative.

4.2 Diffraction par une surface rugueuse

La modélisation de la diffraction d'une onde électromagnétique par une surface rugueuse est un domaine de recherche très étudié. Elle permet notamment de considérer la diffraction d'ondes sur des surfaces au profil aléatoire, tel que la surface d'une étendue d'eau (mer, océan) ou encore d'une forêt. La résolution d'un problème de diffraction par une telle surface est numériquement difficile à résoudre de par le caractère multi-échelle (fractal) des géométries à traiter. Ce type de surfaces est donc intéressant car il offre une grande variété de configurations, dont certaines permettent de jauger les limites d'une formulation.

En particulier, il est possible de distinguer d'une part les surfaces rugueuses *périodiques*, et d'autre part les surfaces rugueuses à *profil aléatoire* [40]. Si les premières peuvent être définies par des fonctions périodiques à diverses échelles, seules certaines caractéristiques statistiques sont connues dans le cas des surfaces rugueuses aléatoires. Ces dernières permettent toutefois d'approcher plus fidèlement les scènes naturelles (sols, végétation, surface de la mer, *etc.*).

Il est également possible de définir des surfaces rugueuses de façon déterministe, sans qu'elles soient nécessairement périodiques. Les *surfaces de Weierstrass* répondent notamment à cette description, et permettent d'obtenir des profils géométriquement complexes, réglés selon divers paramètres.

Dans un premier temps, nous étudierons quelques surfaces de Weierstrass supposées invariantes par translation selon l'axe $\boldsymbol{u}_z$, donc en dimension 2. Nous verrons notamment comment les paramètres qui définissent ces surfaces agissent sur leur géométrie, ainsi que sur la nécessité de compresser la matrice impédance Z d'une telle surface. Nous pourrons ainsi résoudre le système $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ grâce aux diverses méthodes développées précédemment. Nous généraliserons ensuite cette étude au cas de surfaces de Weierstrass en dimension 3.

4.2.1 Cas des surfaces de Weierstrass $2D$

Définition en dimension 2

Nous considérons ici le cas d'une surface (linéique) de Weierstrass ($2D$) [41, 42], supposée invariante par translation selon $\boldsymbol{u}_z$ et dont les points $(x, y) \in \mathbb{R}^2$ sont définis pour $x \in [0, l]$ par

$$y = \mathrm{W}_{2D}(x) = h \sum_{n=n_1}^{n_2} \left(b^{-Hn} cos(2\pi b^n x) \right), \tag{4.2.1}$$

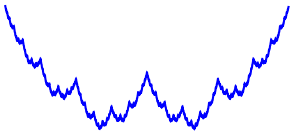
avec

- l : longueur de la surface ;
- h : hauteur de la surface ;
- n_1 : échelle minimale (en général, $n_1 = 0$ ou 1) ;
- n_2 : échelle maximale ;
- b : lacunarité de la surface fractale ;
- D_f : dimension fractale telle que $H = 2 - D_f$.

Chacun de ces paramètres agit différemment sur l'allure de la surface de Weierstrass.

Influence des divers paramètres

Plaçons nous à la fréquence de travail $f = 3\,\mathrm{GHz}$ ($\lambda \simeq 0{,}1\,\mathrm{m}$), et posons $n_1 = 1$, $b = 2$, $l = 0{,}5\,\mathrm{m} \simeq 5\lambda$ et $h = 0{,}1\,\mathrm{m} \simeq \lambda$. On peut alors observer l'influence des paramètres n_2 et D_f sur l'allure de la surface de Weierstrass en figure 4.2.1.

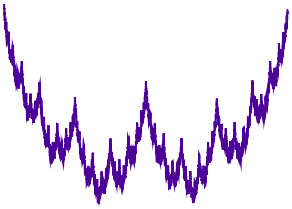


(a) Cas n° 1 : $n_2 = 10$, $D_f = 1.25$

(b) Cas n° 2 : $n_2 = 5$, $D_f = 1.5$

(c) Cas n° 3 : $n_2 = 15$, $D_f = 1.5$

(d) Cas n° 4 : $n_2 = 5$, $D_f = 1.75$



(e) Cas n° 5 : $n_2 = 15$, $D_f = 1.75$

FIGURE 4.2.1 – Influence des paramètres n_2 et D_f sur l'allure d'une surface de Weierstrass.

Nous remarquons en particulier que plus le nombre d'échelles $n_e = n_2 - n_1 + 1$ est élevé, plus la surface rugueuse comporte de détails fins. Par ailleurs, plus la dimension fractale D_f est proche de 2, plus l'amplitude des diverses échelles est élevée, augmentant alors la longueur effective $l_{\text{eff.}}$ de la surface.

Ainsi, plus H est petit (en valeur absolue), plus la surface nécessite une discrétisation fine. Il en est de même pour le paramètre n_2 à n_1 fixé.

Choix du pas de discrétisation

Remarquons que,

- si $D_f = 2$ et $n_1 = n_2 = 1$, alors $\mathrm{W}_{2D}(x) = h\cos(4\pi x)$, le profil de la surface est alors purement *sinusoïdal* ;
- si de plus $b = 0$, alors $\mathrm{W}_{2D}(x) = h$ et la surface de Weierstrass devient une *plaque lisse*.

Dans le cas de la plaque, la surface est de longueur effective $l_{\text{eff.}} = l = 5\lambda$. La règle d'un pas de maillage $p = \lambda/10$ est alors suffisante pour décrire la géométrie de cette surface parfaitement lisse, puisque deux points suffisent à modéliser un segment.

En revanche, dans le cas d'un profil sinusoïdal, il peut être nécessaire d'avoir un pas de discrétisation plus fin pour décrire précisément la géométrie, en fonction de la tolérance ε acceptée sur l'erreur relative.

L'étude du profil sinusoïdal est intéressante car elle permet de déterminer le pas de discrétisation nécessaire pour obtenir des résultats à une précision ε donnée. Le nombre de DDLs n_ε permettant d'avoir la précision ε désirée peut s'écrire en fonction du nombre d'échelles $n_e = n_2 - n_1 + 1$ et de la lacunarité b (qui définit la périodicité de la surface), selon la formule empirique

$$n_\varepsilon = b^{n_e} l_\varepsilon, \tag{4.2.2}$$

où l_ε représente le nombre de DDLs nécessaires pour approcher une surface sinusoïdale avec une erreur inférieure à ε.

Pour déterminer la valeur de l_ε selon la précision souhaitée, nous avons considéré la surface sinusoïdale que nous avons maillée avec différents nombres de segments et nous avons résolu le système plein $Z\boldsymbol{j} = \boldsymbol{e}$. Nous en avons déduit la SER bistatique. En prenant pour référence une discrétisation de 20 000 DDLs ($l_\varepsilon = 10\,000$), nous avons alors calculé l'erreur relative sur la SER bistatique en fonction de la discrétisation. La figure 4.2.2 représente cette SER bistatique pour plusieurs valeurs de l_ε tandis que la figure 4.2.3 présente l'erreur relative sur la SER bistatique en fonction du nombre de DDLs considéré, définie par

$$\delta^{(\mathrm{r})}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_{\text{réf.}}) = \frac{1}{n_\theta} \sum_{k=1}^{n_\theta} \frac{\|\sigma_k - \sigma_{\text{réf.}\,k}\|_2}{\|\sigma_{\text{réf.}\,k}\|_2}, \tag{4.2.3}$$

où $\boldsymbol{\sigma}$ désigne le vecteur de $\mathbb{R}^{n_\theta}$ contenant l'évaluation de la SER bistatique pour n valeurs différentes de θ, $\boldsymbol{\sigma}_{\text{réf.}}$ étant la solution de référence (ici, $n_\theta = 360$).

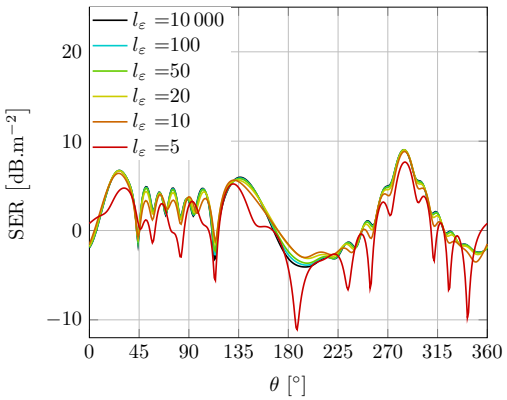


FIGURE 4.2.2 – SER bistatique pour différentes valeurs de l_ε.

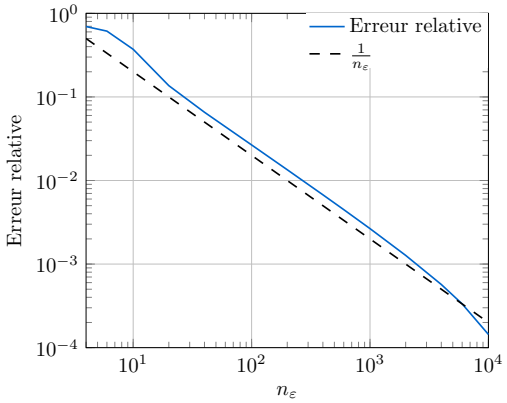


FIGURE 4.2.3 – Erreur sur la SER bistatique en fonction de n_ε.

La figure 4.2.2 permet de déterminer le nombre de segments nécessaires pour mailler une échelle de sinus avec un pas suffisamment fin pour que la SER bistatique soit fidèle à la physique. On constate notamment que cette SER converge vers la SER calculée avec $l_\varepsilon = 10\,000$ (soit 20 000 DDLs). En particulier, les cas $l_\varepsilon = 50$ et $l_\varepsilon = 100$ semblent permettre une représentation de la SER suffisamment précise pour que les courbes soient quasiment confondues.

La figure 4.2.3 indique plus précisément l'erreur relative entre le cas de référence et les différents cas traités. Il s'agit d'une erreur proprotionnelle à $\frac{1}{n_\varepsilon}$. En particulier, on a une erreur relative sur la SER d'environ $2{,}66 \times 10^{-2}$ pour $l_\varepsilon = 50$, et de $1{,}34 \times 10^{-2}$ pour $l_\varepsilon = 100$.

Dans la suite, nous considérerons qu'une erreur ε de l'ordre de 10^{-2} est acceptable, et nous prendrons $l_\varepsilon = 100$. Nous appliquerons la formule (4.2.2) pour déterminer le nombre de DDLs nécessaires à mailler la surface de Weierstrass en nous assurant une erreur de l'ordre de ε par rapport à la physique.

Considérons les surfaces de Weierstrass, définies dans le tableau 4.2.1 par leurs paramètres et leur nombre de DDLs (nous aurons dans tous les cas $n_1 = 1, b = 2, l = 0{,}5\,\text{m} \simeq 5\lambda$ et $h = 0{,}1\,\text{m} \simeq \lambda$).

Cas	n_2	D_f	Nombre de DDLs
Cas n$^\text{o}$ 1	10	1,25	102 400
Cas n$^\text{o}$ 2	5	1,5	3200
Cas n$^\text{o}$ 3	15	1,5	3 276 800
Cas n$^\text{o}$ 4	5	1,75	3200
Cas n$^\text{o}$ 5	15	1,75	3 276 800

TABLEAU 4.2.1 – Définition des exemples de surfaces de Weierstrass traités.

Les cas n$^\text{os}$ 1 à 5 sont représentés à la figure 4.2.1.

4.2.2 Application du format $\mathcal{H}$-matrice au cas d'une surface de Weierstrass $2D$

Pour rappel, nous disposons d'une RAM de 192 Go. Les cas n$^\text{os}$ 3 et 5 nécessitent un nombre de DDLs trop élevé pour que nous puissions construire la matrice impédance Z pleine. Il est donc nécessaire d'utiliser une méthode de compression. Il nous est toutefois possible de construire la matrice impédance pleine dans le cas des surfaces de Weierstrass n$^\text{os}$ 1, 2 et 4.

Performances des méthodes de compression

Considérons d'abord le cas n$^\text{o}$ 1 et construisons la matrice Z pleine et la $\mathcal{H}$-matrice $\widetilde{Z}$ correspondante. Nous pouvons alors résoudre les systèmes $Z\boldsymbol{j}_{\text{réf.}} = \boldsymbol{e}$ et $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$, puis comparer les vecteurs solutions $\boldsymbol{j}$ et $\boldsymbol{j}_{\text{réf.}}$ pour différentes valeurs de $\varepsilon_{\text{rés.}}$.

Nous assemblons les $\mathcal{H}$-matrices par HCA avec $\varepsilon_{\text{HCA}} = 10^{-m_T}$ pour $m_T \in [\![2, 4]\!]$. Pour la résolution, nous utilisons l'algorithme GMRES avec $\varepsilon_{\text{GMRES}} = 10^{-4}$. Les performances de l'HCA et de la résolution itérative sont alors disponibles dans le tableau 4.2.2.

Dans ce tableau, la donnée du résidu nous permet de vérifier la convergence de l'algorithme GMRES, qui est atteinte dans tous les cas puisque ce résidu est toujours inférieur à $\varepsilon_{\text{GMRES}} = 10^{-4}$.

Les temps d'assemblage des $\mathcal{H}$-matrices $\widetilde{Z}$ sont, quelle que soit la valeur de ε_{HCA}, drastiquement inférieurs au temps d'assemblage de la matrice Z. Par exemple, le temps d'assemblage de $\widetilde{Z}$ pour $\varepsilon_{\text{HCA}} = 10^{-4}$ représente 0.42% du temps d'assemblage de la matrice Z pleine. Nous pouvons faire un constat similaire pour le temps de résolution qui, pour $\varepsilon_{\text{HCA}} = 10^{-4}$, représente 1.33% du temps de résolution obtenu dans le cas du système plein. Ces temps sont évidemment d'autant plus courts que la précision ε_{HCA} est grossière.

Les temps de résolution du système dépendent doublement de ε_{HCA}, d'une part parce qu'une précision fine augmente le temps d'une itération de GMRES, et d'autre part parce qu'une précision grossière dégrade le système à résoudre et nécessite donc plus d'itérations pour parvenir à convergence.

	Sans	HCA		
	compression	$\varepsilon_{\text{HCA}} = 10^{-2}$	$\varepsilon_{\text{HCA}} = 10^{-3}$	$\varepsilon_{\text{HCA}} = 10^{-4}$
Temps d'assemblage (s)	15 012,252	63,224	82,608	101,072
Temps de résolution (s)	5422,228	64,168	62,044	71,928
Temps total (s)	**20 434,48**	**127,392**	**144,652**	**173,0**
Nombre d'itérations	199	218	200	199
Temps par itération (s)	27,247	0,294	0,310	0,361
Stockage de Z / $\widetilde{Z}$ (Go)	167,772	0,287	0,393	0,521
Erreur sur la SER	$\emptyset$	$4{,}18 \times 10^{-3}$	$1{,}20 \times 10^{-4}$	$1{,}48 \times 10^{-5}$
Résidu	$9{,}99 \times 10^{-5}$	$9{,}98 \times 10^{-5}$	$9{,}87 \times 10^{-5}$	$9{,}99 \times 10^{-5}$

TABLEAU 4.2.2 – Performances de la compression HCA dans le cas n° 1 pour différentes valeurs de ε_{HCA} ($\varepsilon_{\text{GMRES}} = 10^{-4}$).

La précision ε_{HCA} agit également sur l'erreur relative constatée sur la SER bistatique, qui est d'autant plus faible que la précision ε_{HCA} est fine, tout en étant toujours strictement inférieure à ε_{HCA}. Nous pouvons d'ailleurs visualiser l'allure de cette SER bistatique en figure 4.2.4, sur laquelle seule la précision $\varepsilon_{\text{HCA}} = 10^{-2}$ semble très légèrement différente de la SER de la matrice pleine.

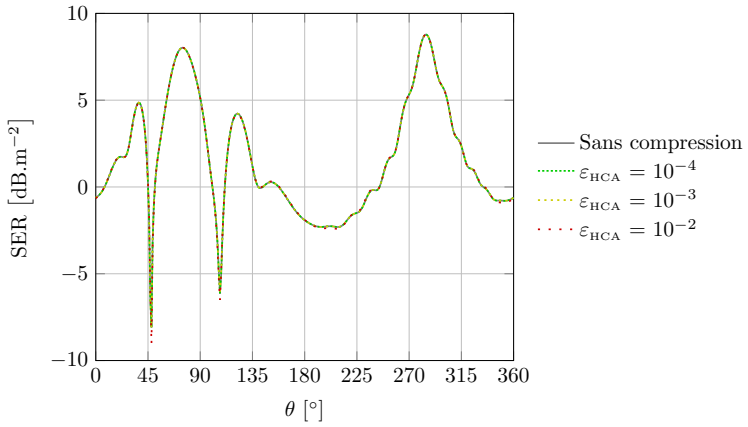


FIGURE 4.2.4 – SER bistatique de la surface de Weierstrass n° 3 pour plusieurs valeurs de ε_{HCA}.

Enfin, la précision ε_{HCA} a une influence sur la mémoire vive (en anglais *Random Access Memory*, notée RAM) utilisée pour le stockage de la $\mathcal{H}$-matrice $\widetilde{Z}$, qui reste négligeable face au stockage de la matrice Z pleine puisqu'elle est d'au plus 0.31% de ce dernier.

Cette étude nous permet de valider l'utilisation de l'HCA en tant que méthode de compression dans le cas des surfaces de Weierstrass. Dans la suite nous utiliserons cette méthode et nous fixerons $\varepsilon_{\text{HCA}} = 10^{-4}$.

Influence de la complexité de la surface de Weierstrass

Traitons par compression HCA les cas n$^{\text{os}}$ 2 à 5. Nous rappelons que l'étude s'effectue à une fréquence $f = 3\,\text{GHz}$, soit une longueur d'onde $\lambda \approx 0{,}1\,\text{m}$. Par ailleurs, pour les cas

n^{os} 3 et 5, nous avons les paramètres $l \approx 5\lambda$, $h \approx \lambda$, $b = 2$, $n_1 = 1$ et $n_2 = 15$. Dans les quatre cas, nous utiliserons $n_\varepsilon = 3\,276\,800$ DDLs, correspondant au nombre de DDLs indiqué pour les cas n^{os} 3 et 5, afin de comparer équitablement les temps d'assemblage et la RAM utilisée (*cf.* tableau 4.2.3).

	Cas n^o 2	Cas n^o 3	Cas n^o 4	Cas n^o 5
n_2	5	15	5	15
D_f	1,5	1,5	1,75	1,75
Temps d'assemblage (s)	3299,58	8021,10	4231,56	14 585,3
Stockage de $\widetilde{Z}$ (Go)	18,23	36,34	21,64	55,28

TABLEAU 4.2.3 – Performances de la compression par HCA pour les surfaces de Weierstrass n^{os} 2 à 5 ($\varepsilon_{\text{HCA}} = 10^{-4}$)

Nous constatons que l'allure plus ou moins géométriquement complexe des quatre surfaces étudiées se répercute à la fois sur le temps d'assemblage et sur la RAM de la $\mathcal{H}$-matrice $\widetilde{Z}$. En effet, plus n_2 ou D_f est élevé, plus la $\mathcal{H}$-matrice $\widetilde{Z}$ est coûteuse, que ce soit en terme de temps d'assemblage ou de RAM nécessaire à son stockage.

Comparaison des différentes méthodes de résolution

Une fois l'assemblage de la $\mathcal{H}$-matrice $\widetilde{Z}$ effectué, il est possible de comparer les différents solveurs étudiés au chapitre 3. Nous effectuerons cette comparaison en particulier pour les cas n^{os} 3 et 5 qui nécessitent un grand nombre de DDLs ($n_\varepsilon = 3\,276\,800$).

GMRES. Dans un premier temps, nous résolvons le système par GMRES avec $\varepsilon_{\text{GMRES}} = 10^{-4}$. Les performances de cette résolution sont résumées dans le tableau 4.2.4.

	Cas n^o 3 ($D_f = 1.5$)	Cas n^o 5 ($D_f = 1.75$)
Temps d'assemblage (s)	8021,10	14 585,3
Temps de GMRES (s)	49 135,98	2 892 889
Temps total (s)	**57 157,08**	**2 907 474**
Nombre d'itérations	1808	54 349
Temps par itération (s)	27,177	56,338
Stockage de $\widetilde{Z}$ (Go)	36,34	55,28

TABLEAU 4.2.4 – Performances du solveur itératif GMRES pour les surfaces de Weierstrass n^{os} 3 et 5 ($\varepsilon_{\text{HCA}} = 10^{-4}$ et $\varepsilon_{\text{GMRES}} = 10^{-4}$).

La résolution par GMRES a l'avantage de ne solliciter que la RAM nécessaire au stockage de la $\mathcal{H}$-matrice $\widetilde{Z}$. Cependant le temps de résolution dépend fortement du nombre d'itérations nécessaires pour atteindre la convergence, qui peut être élevé, comme c'est le cas pour les surfaces n^{os} 3 et 5. Par ailleurs, avec un nombre de DDLs aussi important ($n_\varepsilon = 3\,276\,800$), le temps d'une itération unique est également élevé, ce qui incite à tenter de réduire ce nombre d'itérations autant que possible.

La comparaison des cas n^{os} 3 et 5 nous permet de constater que l'augmentation du paramètre D_f implique également une dégradation de la résolution itérative, puisque le cas n^o 5 nécessite beaucoup plus d'itérations que le n^o 3 pour parvenir à convergence. Par ailleurs le temps d'une itération est logiquement plus long pour le cas n^o 5, à nombre de DDLs égal, comme le laissait présager le tableau 4.2.3, et en particulier la RAM utilisée pour le stockage de $\widetilde{Z}$, plus important dans le cas n^o 5 que dans le cas n^o 3. La résolution par GMRES est donc d'autant plus problématique que la surface est géométriquement complexe.

Dans la suite, nous évaluerons la précision des méthodes étudiées en prenant pour référence la SER bistatique obtenue par résolution GMRES.

Solveur direct $\mathcal{H}$-LU. Une autre stratégie consiste en résoudre notre système en utilisant le solveur direct $\mathcal{H}$-LU développé en partie 2.5.4.

Nous pouvons observer les performances de cette méthode dans le tableau 4.2.5.

	Cas n^o 3 ($D_f = 1.5$)	Cas n^o 5 ($D_f = 1.75$)
Temps d'assemblage (s)	8021,10	14 585,3
Temps de décomposition $\mathcal{H}$-LU (s)	11 957,50	28 781,6
Temps de résolution $\mathcal{H}$-LU (s)	18,872	42,284
Temps total (s)	**19 997,472**	**43 409,3**
Stockage de $\widetilde{LU}$ (Go)	39,040	59,186
Résidu	$7{,}26 \times 10^{-5}$	$1{,}94 \times 10^{-4}$
Erreur par rapport au GMRES	$1{,}14 \times 10^{-4}$	$3{,}71 \times 10^{-5}$

TABLEAU 4.2.5 – Performances du solveur direct $\mathcal{H}$-LU pour les surfaces de Weierstrass n^{os} 3 et 5 ($\varepsilon_{\text{HCA}} = 10^{-4}$ et $\varepsilon_{LU} = 10^{-4}$).

Ici, le temps de résolution dépend surtout du temps de décomposition $\mathcal{H}$-LU. Ce solveur est plus rapide que le GMRES, que ce soit dans le cas n^o 3 ou le cas n^o 5. La $\mathcal{H}$-matrice $\widetilde{Z}$ est écrasée et laisse place à la $\mathcal{H}$-matrice $\widetilde{LU}$ contenant la décomposition $\mathcal{H}$-LU de $\widetilde{Z}$. La RAM utilisée est alors légèrement supérieure à la RAM nécessaire au stockage de $\widetilde{Z}$.

La comparaison des cas n^{os} 3 et 5 nous permet de constater que le coût du solveur direct $\mathcal{H}$-LU est moins sensible à la complexité de la géométrie traitée que ne l'est le solveur itératif GMRES. Le rapport entre le temps d'assemblage et le temps de décomposition $\mathcal{H}$-LU est toutefois plus important dans le cas n^o 3 (67%) que dans le cas n^o 5 (50%). Le temps de décomposition semble donc être sensible à la complexité de la géométrie traitée, tout en étant limité, notamment par sa proportionnalité en $\mathcal{O}(n \log^3(n))$, contrairement au solveur GMRES dont le temps de réalisation dépend du nombre difficilement prédictible d'itérations nécessaires à parvenir à convergence.

On constate également que l'erreur obtenue par rapport au GMRES est plus importante dans le cas n^o 5 que dans le cas n^o 3. Ceci semble confirmer l'hypothèse selon laquelle plus la géométrie traitée est complexe, plus le conditionnement de $\widetilde{Z}$ est dégradé.

Préconditionnement $\mathcal{H}$-LU du GMRES. Les problèmes de diffraction par une surface rugueuse sont d'autant moins bien conditionnés que la géométrie traitée est complexe.

Ceci peut mener à une solution itérative longue car nécessitant beaucoup d'itérations pour converger, comme nous avons pu le voir dans le tableau 4.2.4. En pratique, il peut également arriver que le problème soit si mal conditionné qu'on ne puisse pas atteindre la convergence souhaitée. Dans une telle situation, l'utilisation d'un préconditionneur semble tout indiquée.

Comme nous avons pu le voir dans la partie 3.3.2, l'utilisation d'un préconditionneur $\widetilde{P}$ peut réduire considérablement le nombre d'itérations nécessaires à la résolution d'un système linéaire hiérarchique avec une méthode itérative telle que GMRES.

Nous utilisons le préconditionneur $\widetilde{P} = \widetilde{LU}$ où $\widetilde{LU}$ est la décomposition $\mathcal{H}$-LU à précision $\varepsilon_{\text{préc.}}$ de la $\mathcal{H}$-matrice $\widetilde{Z}$, préalablement recompressée par *coarsening* à précision $\varepsilon_{coars.} = \varepsilon_{\text{préc.}}$. Les performances de ce solveur sur la surface de Weierstrass n° 3 sont visibles au tableau 4.2.6.

$\varepsilon_{\text{préc.}}$	10^{-2}	10^{-3}
Temps d'assemblage (s)	8021,10	
Temps de *coarsening* à $\varepsilon_{\text{préc.}}$ (s)	359,184	554,912
Temps de décomposition $\mathcal{H}$-LU (s)	4952,43	10 129,6
Temps de GMRES (s)	39 401,1	247,188
Temps total (s)	**52 733,8**	**18 952,8**
Nombre d'itérations de GMRES	699	2
Temps par itération de GMRES (s)	56,367	123,594
Stockage de $\widetilde{Z}$ (Go)	36,34	
Stockage de $\widetilde{P}$ (Go)	18,75	27,42
Stockage total (Go)	**55,09**	**63,76**
Résidu	$9{,}99 \times 10^{-5}$	$9{,}01 \times 10^{-5}$
Erreur par rapport au GMRES	$1{,}19 \times 10^{-4}$	$1{,}07 \times 10^{-4}$

TABLEAU 4.2.6 – Performances du solveur itératif GMRES préconditionné pour la surface de Weierstrass n° 3 ($\varepsilon_{\text{HCA}} = 10^{-4}$ et $\varepsilon_{\text{GMRES}} = 10^{-4}$).

Comme nous avons pu le voir en partie 3.3.2, l'application d'un préconditionneur $\widetilde{P}$ change le temps de résolution, puisque le temps de construction de ce préconditionneur doit être pris en compte. S'agissant ici d'un préconditionneur de type $\mathcal{H}$-LU, le temps de décomposition $\mathcal{H}$-LU peut occuper une part plus ou moins importante du temps total de résolution, en fonction de la précision $\varepsilon_{\text{préc.}}$ choisie. En particulier, dans le cas n° 3, nous pouvons constater que le temps de décomposition $\mathcal{H}$-LU est inférieur au temps de résolution itérative pour $\varepsilon_{\text{préc.}} = 10^{-2}$, mais que cette hiérarchie est inversée pour $\varepsilon_{\text{préc.}} = 10^{-3}$. Dans ce dernier cas, le stockage de $\widetilde{P}$ est plus important et le temps d'une itération est d'autant plus élevé, mais le nombre d'itérations est drastiquement réduit, contrairement au cas $\varepsilon_{\text{préc.}} = 10^{-2}$ qui dépend encore beaucoup du nombre d'itérations.

Le temps de résolution total favorise le préconditionnement à précision plus forte ($\varepsilon_{\text{préc.}} = 10^{-3}$). Cette conclusion serait la même dans le cas d'un problème à plusieurs seconds membres.

Enfin il est important de souligner que le choix de ce solveur implique d'avoir une RAM disponible suffisamment importante pour stocker à la fois $\widetilde{Z}$ et $\widetilde{P}$, les deux $\mathcal{H}$-matrices étant

nécessaires à la résolution itérative.

La comparaison de ces méthodes de résolution nous a permis de constater à quel point la complexification d'une géométrie peut influer sur le temps de résolution, à nombre de DDLs égal. D'une part, une géométrie plus complexe implique une RAM plus importante pour le stockage de $\widetilde{Z}$, ce qui ralentit inévitablement les opérations qui s'y appliquent. D'autre part, le conditionnement dégrade la résolution, soit temporellement en augmentant le nombre d'itérations dans le cas d'une méthode itérative, soit en terme de précision dans le cas du solveur direct $\mathcal{H}$-LU. Le préconditionnement du GMRES semble alors un bon compromis pour s'affranchir partiellement de ces dégradations.

4.2.3 Généralisation aux surfaces de Weierstrass $3D$

L'étude précédente a permis de déterminer l'influence des divers paramètres définissant une surface de Weierstrass $2D$. Il s'agit à présent d'appréhender la façon dont ces conclusions se traduisent dans le cas d'une surface de Weierstrass $3D$.

Expression d'une surface de Weierstrass en dimension 3

Nous considérons le cas d'une surface de Weierstrass $3D$ [41, 42], dont les points $(x, y, z) \in \mathbb{R}^3$ peuvent être définis pour $x \in [0, l]$ et $y \in [0, L]$ par

$$z = W(x, y) = h \sum_{n=n_1}^{n_2} \left(b^{-Hn} \sin\Big(b^n \big(x\cos(\psi_n) + y\sin(\psi_n)\big)\Big)\right). \tag{4.2.4}$$

Aux paramètres déjà définis dans le cas $2D$ s'ajoutent

- L : largeur de la surface ;
- ψ_n : phase en fonction de n.

Pour le cas $3D$, on pose cette fois-ci

$$H = 3 - D_f. \tag{4.2.5}$$

La surface sera d'autant plus dégradée que H sera petit (en valeur absolue) ou D_f proche de 3.

Le paramètre ψ_n permet de caractériser le type de surface obtenu. Il est possible de lui donner une valeur aléatoire différente pour tout $n \in [\![n_1, n_2]\!]$. Si tous les ψ_n sont égaux, alors la surface n'est rugueuse que dans la direction désignée par ce paramètre. En revanche, si ce paramètre est linéairement réparti sur l'intervalle $[-\pi; \pi]$, c'est-à-dire

$$\psi_n = -\pi + 2\pi \frac{n - n_1}{n_2 - n_1}, \tag{4.2.6}$$

alors la surface est isotrope [43].

Remarque 4.2.1. *En particulier :*

- *si on pose $n_1 = 0$, l'expression de ψ_n est simplifiée, avec pour $n \in [\![0, n_2]\!]$ et $0 < n_2$*

$$\psi_n = -\pi + \frac{2\pi n}{n_2}; \tag{4.2.7}$$

- *dans le cas où $n_1 = n_2$, on se ramène au cas d'un paramètre ψ_n constant en fonction de n, et donc à une surface de Weierstrass $2D$.*

Dans la suite, nous poserons $n_1 = 0$.

Choix du pas de maillage

Nous pouvons généraliser l'équation (4.2.2) en considérant que le passage à une surface $3D$ implique la mise au carré du nombre l_ε de DDLs nécessaires à mailler une échelle sinusoïdale à précision ε. Ainsi nous suivront la règle

$$n_\varepsilon = b^{n_e} l_\varepsilon^2, \tag{4.2.8}$$

avec l_ε à choisir en fonction de la précision désirée.

En réalité, la valeur l_ε sera surtout limitée par la RAM disponible, ou impliquera un choix restreint du nombre d'échelles $n_e = n_2 - n_1 + 1$. Le tableau 4.2.7 donne à titre indicatifs le nombre de DDLs pour plusieurs valeurs de n_2 et de l_ε.

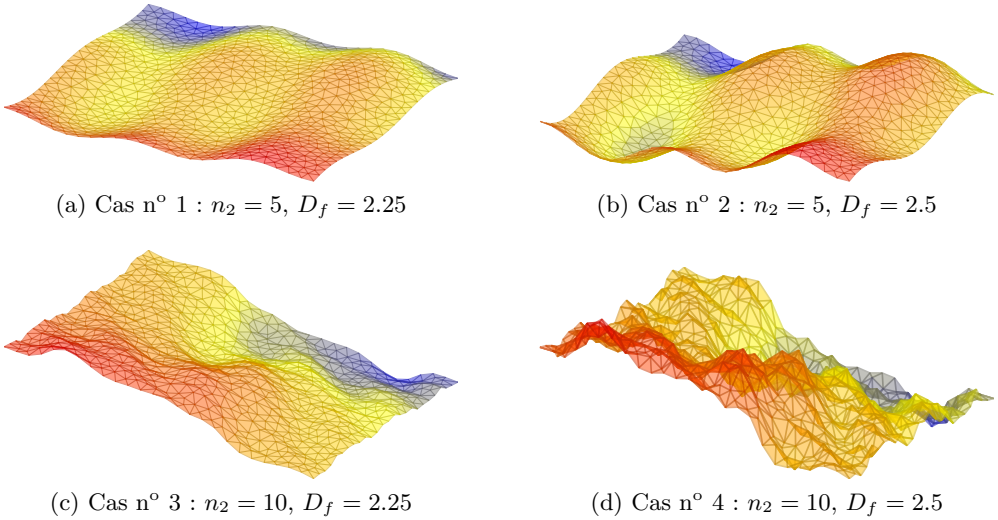
	$n_2 = 5$	$n_2 = 10$	$n_2 = 15$
$l_\varepsilon = 5$	1600	51 200	1 638 400
$l_\varepsilon = 10$	6400	204 800	6 553 600
$l_\varepsilon = 20$	25 600	819 200	26 214 400
$l_\varepsilon = 50$	160 000	5 120 000	163 840 000

TABLEAU 4.2.7 – Nombre minimal de DDLs nécessaires au maillage d'une surface de Weierstrass $3D$ en fonction de l_ε et de n_2 ($n_1 = 0$, $b = 2$).

Ce tableau est *a priori* valable pour toute valeur de D_f, bien que nous sachions que ce paramètre dégrade beaucoup l'allure de la surface, et donc le conditionnement du système à résoudre.

Fixons par exemple $l_\varepsilon = 20$. Il est alors possible de résoudre le système $\widetilde{Z}\boldsymbol{j} =$ avec un nombre raisonnable de DDLs pour $n_2 = 5$ et $n_2 = 10$. C'est ce que nous allons faire pour plusieurs valeurs de D_f.

Nous allons résoudre le système $\widetilde{Z}\boldsymbol{j} = \boldsymbol{e}$ en fixant les paramètres $n_1 = 0$, $b = 2$, $h = 0{,}1\,\text{m}$ et $L = l = 0{,}5\,\text{m}$. Nous ferons donc uniquement varier la valeur des paramètres n_2 et D_f. La figure 4.2.5 présente les surfaces de Weierstrass correspondant à $D_f = 2.25$ et $D_f = 2.5$.



(a) Cas n° 1 : $n_2 = 5$, $D_f = 2.25$

(b) Cas n° 2 : $n_2 = 5$, $D_f = 2.5$

(c) Cas n° 3 : $n_2 = 10$, $D_f = 2.25$

(d) Cas n° 4 : $n_2 = 10$, $D_f = 2.5$

FIGURE 4.2.5 – Surfaces de Weierstrass $3D$ pour différentes valeurs de D_f et de n_2.

Nous utiliserons les cas n^{os} 1 et 2 définis en figure 4.2.5 pour comparer les performances de l'ACA et de l'HCA sur ce type de surfaces. Il est en effet possible de mailler ces surfaces avec un nombre réduit de DDLs sans effet dommageable sur la précision.

Comparaison des méthodes de compression ACA et HCA

Pour ces deux cas, nous considérons un maillage de 25 860 DDLs et nous assemblons la matrice Z pleine, ainsi que la $\mathcal{H}$-matrice $\widetilde{Z}$, d'une part par ACA, d'autre part par HCA. Nous fixons la précision de ces méthodes à 10^{-4} et nous résolvons alors le système habituel afin de comparer les performances de ces deux méthodes de compression avec celles de l'assemblage plein.

L'assemblage se fera à précision $\varepsilon_{\text{ACA}} = \varepsilon_{\text{HCA}} = 10^{-4}$. Pour la résolution, nous utiliserons la méthode GMRES à tolérance $\varepsilon_{\text{GMRES}} = 10^{-4}$. Les résultats de cette étude sont disponibles au tableau 4.2.8.

	Cas n° 1			Cas n° 2		
	Dense	ACA	HCA	Dense	ACA	HCA
Temps d'assemblage (s)	6211,756	1133,772	716,816	6017,104	1242,172	844,584
Temps de résolution (s)	2385,736	160,760	218,172	2620,372	300,308	350,760
Temps total (s)	**8597,492**	**1294,532**	**934,988**	**8637,476**	**1542,480**	**1195,344**
Nombre d'itérations		815			884	
Temps par itération (s)	2,927	0,197	0,268	2,964	0,340	0,397
Stockage de Z / $\widetilde{Z}$ (Go)	10,699	0,768	0,876	10,699	0,881	0,957
Erreur sur la SER	$\emptyset$	$3{,}7 \times 10^{-5}$	$5{,}5 \times 10^{-5}$	$\emptyset$	$1{,}9 \times 10^{-5}$	$2{,}8 \times 10^{-5}$

Tableau 4.2.8 – Comparaison des méthodes de compression dans le cas d'une surface de Weierstrass $3D$.

Ce tableau nous permet de constater que l'ACA est moins rapide que l'HCA, mais cette dernière semble nécessiter une RAM plus importante (bien que du même ordre). Quelle que soit la méthode de compression, le temps total du calcul hiérarchique est plus court que celui du calcul sans compression, et permet une économie de mémoire importante tout en conservant une précision de l'ordre de $\varepsilon_{\text{GMRES}}$ par comparaison avec le calcul sans compression.

Nous pouvons ici aussi remarquer que le fait de travailler sur une structure plus complexe (cas n° 2 par rapport au cas n° 1) implique un temps d'assemblage plus long et un stockage plus lourd.

Pour conclure sur le cas des surfaces de Weierstrass, nous pouvons souligner que ces applications sont très consommatrices de temps et de mémoire. La compression des matrices impédances de ces cas d'étude permet d'envisager de traiter des problèmes avec une grande quantité de DDLs, et donc une meilleure modélisation de la surface dans toutes ses aspérités.

4.3 Fibres à Cristaux Photoniques

Une fibre à cristaux photoniques (en anglais *Photonic-Crystal Fiber*, notée PCF) est une fibre optique constituée d'une gaine et d'une ou plusieurs inclusions, appelées *cœurs*.

Ses propriétés optiques dépendent fortement de sa géométrie, ainsi que de ses caractéristiques diélectriques. Les PCFs sont des structures très utilisées en optique. Cette application, en marge de notre domaine d'étude initial, a toutefois fait l'objet d'une collaboration avec Julien Vincent et Han-Cheng Seat dans le cadre d'un projet du programme *Toulouse Tech Inter Lab (TTIL)*. Cette collaboration est à l'origine d'une communication à la conférence META 2016 [44]. Le problème physique et les développements théoriques de cette application sont disponibles en annexe D.

Dans cette partie, nous considérerons une PCF possédant C cœurs, chacun discrétisé à l'aide de N points, et nous partirons directement du système explicité en annexe D. En particulier, ce système devient grand dès lors que l'on considère un grand nombre de cœurs, comme c'est le cas pour la PCF dite *Hollow Core* ($C = 121$). La figure 4.3.1 en donne une illustration.

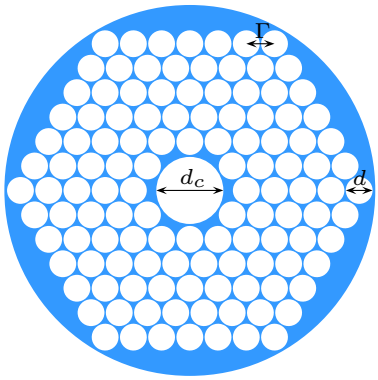


FIGURE 4.3.1 – Représentation en coupe de la PCF HC-1550 à 121 trous.

Nous expliquerons comment nous avons adapté le système à résoudre au format $\mathcal{H}$-matrice, tant pour l'assemblage que pour les opérations nécessaires à la résolution, puis nous donnerons quelques résultats de simulation pour $C = 121$.

4.3.1 Adaptation du problème au format $\mathcal{H}$-matrice

Assemblage

Le problème physique consiste à résoudre, pour un vecteur aléatoire $\boldsymbol{\Phi}$, le système

$$F\boldsymbol{u} = \boldsymbol{\Phi}, \tag{4.3.1}$$

avec

$$F = \begin{bmatrix} W_e & K_e & K_eT & 0 \\ W_i & K_i & 0 & 0 \\ \alpha KeT & 0 & W_e & \beta K_e \\ 0 & 0 & W_i & K_i \end{bmatrix} \text{ et } \boldsymbol{x} = \begin{bmatrix} \boldsymbol{E} \\ \partial_{\nu_i}\boldsymbol{E} \\ \boldsymbol{H} \\ \partial_{\nu_i}\boldsymbol{H} \end{bmatrix}, \tag{4.3.2}$$

α et β sont des coefficients scalaires réels, W_e et K_e sont des blocs matriciels denses de $\mathbb{C}^{S\times S}$ ($S = C \times N$) et W_i, K_i et T sont des matrices diagonales par blocs de $\mathbb{C}^{S\times S}$ (leur C blocs diagonaux sont de taille $N \times N$). F est donc une matrice de $\mathbb{C}^{4S\times 4S}$. Les définitions exactes de ces matrices ainsi que des coefficients α et β sont disponibles en annexe D. La figure 4.3.2 donne une représentation graphique du remplissage de F dans le cas où $C = 6$.

Le nombre de coefficients non-nuls de cette matrice est alors

$$n_F = 2N^2(2C + 3C^2), \tag{4.3.3}$$

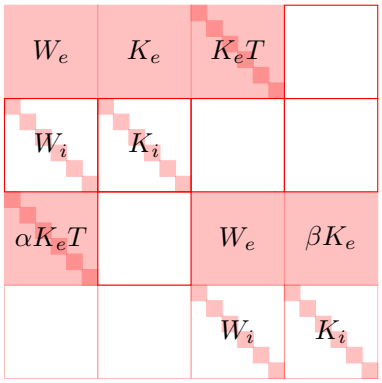


FIGURE 4.3.2 – Représentation schématique de la matrice F.

et la RAM qu'elle occupe est proportionnelle n_F.

L'introduction du format $\mathcal{H}$-matrice peut nous permettre de réduire la RAM nécessaire au stockage de F. En particulier, un assemblage hiérarchique des blocs W_e et K_e denses peut réduire la mémoire utilisée par F.

L'assemblage hiérarchique de F consistera donc à approcher les blocs denses W_e et K_e par des $\mathcal{H}$-matrices $\widetilde{W_e}$ et $\widetilde{K_e}$. On notera alors $\widetilde{F}$ l'approximation de F définie par

$$\widetilde{F} = \begin{bmatrix} \widetilde{W_e} & \widetilde{K_e} & \widetilde{K_e}T & 0 \\ W_i & K_i & 0 & 0 \\ \alpha\widetilde{K_e}T & 0 & \widetilde{W_e} & \beta\widetilde{K_e} \\ 0 & 0 & W_i & K_i \end{bmatrix}. \tag{4.3.4}$$

Remarque 4.3.1. *En pratique, nous avons défini un nouveau type de structure de donnée que nous noterons $\mathcal{F}$-matrice. Cette structure est munie des deux $\mathcal{H}$-matrices $\widetilde{W_e}$ et $\widetilde{K_e}$ de taille $S \times S$, ainsi que des trois listes de C blocs matriciels W_i, K_i et T de taille $N \times N$.*

Arithmétique sur la matrice F

La structure de $\mathcal{F}$-matrice peut en fait être considérée comme une matrice hiérarchique constituée de deux niveaux de hiérarchie. Ainsi, en utilisant les principes de l'arithmétique des $\mathcal{H}$-matrices, il est possible de développer manuellement une arithmétique des $\mathcal{F}$-matrices en se servant des outils déjà développés pour les $\mathcal{H}$-matrices et en ajoutant les opérations arithmétiques adaptées aux matrices diagonales par bloc.

Nous avons ainsi pu définir un produit $\mathcal{F}$-matrice-vecteur, ainsi qu'une décomposition $\mathcal{F} - LU$ suivant le même modèle que les opérations hiérarchiques classiques, limitées à deux niveaux de hiérarchie et avec des étapes adaptées aux différents types de données considérées. Étant donnée la similarité avec les procédés déjà détaillés au chapitre 2, nous ne détaillerons pas ces opérations ici.

4.3.2 Résolution hiérarchique dans le cas d'une PCF à 121 trous

Le système plein est par nature mal conditionné, surtout à proximité de la solution recherchée. Une résolution itérative sans préconditionnement n'est donc pas raisonnable *a priori*. L'utilisation d'un solveur direct n'est envisageable que pour de petites valeurs de N et/ou de C.

Nous allons assembler la matrice pleine F ainsi que la $\mathcal{F}$-matrice $\widetilde{F}$ afin d'en comparer les temps d'assemblage. Nous résoudrons ensuite les deux systèmes matriciels et nous comparerons les performances des solveurs choisis.

Temps d'assemblage

Nous assemblons d'une part la matrice F pleine, et d'autre part la $\mathcal{F}$-matrice $\widetilde{F}$ telle que définie ci-avant. Les blocs $\widetilde{W}_e$ et $\widetilde{K}_e$ sont assemblés par ACA à précision $\varepsilon_{\text{ACA}} = 1 \times 10^{-5}$. On peut alors observer les temps d'assemblage de F et de $\widetilde{F}$ en fonction de $n = 4 \times N \times C$ en figure 4.3.3.

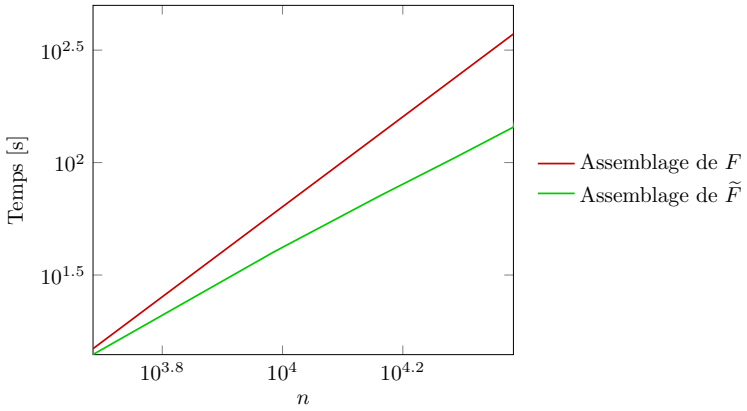


FIGURE 4.3.3 – Comparaison des temps d'assemblage de F et $\widetilde{F}$.

Cette figure nous permet de constater que le temps d'assemblage de $\widetilde{F}$ est favorable dès que $N = 10$ (soit $n = 4840$). La compression par ACA hiérarchique des blocs denses de F permet donc une réduction importante du temps CPU de l'assemblage.

Choix de la méthode de résolution hiérarchique

Comme mentionné précédemment, le système à résoudre est par essence mal conditionné. Ainsi une solveur itératif sans préconditionneur n'est pas envisageable. Par ailleurs, nous avons pu constater lors du développement de la décomposition $\mathcal{H}$-LU, que la solution obtenue avec le solveur direct ne répond pas correctement au critère de précision demandé. En particulier, pour obtenir un résultat précis à ε_{LU} près, il est nécessaire de faire la décomposition $\mathcal{H}$-LU à une précision ε'_{LU} plus fine, ce qui peut vite rendre le temps de résolution long. Cette décomposition est toutefois suffisamment précise pour servir de préconditionneur à un solveur itératif.

Nous avons choisi d'utiliser cette méthode en la comparant au solveur direct pour le système plein, et en essayant plusieurs valeurs de $\varepsilon_{\text{préc.}}$ (*cf.* tableau 4.3.1), mais en fixant $\varepsilon_{\text{GMRES}} = 10^{-5}$, pour $N = 40$ ($n = 19360$).

Ce tableau confirme le décalage de précision de la décomposition $\mathcal{F} - LU$ dans le cas des PCFs. En effet, si la décomposition $\mathcal{F} - LU$ obéissait effectivement à la précision demandée, le solveur GMRES préconditionné ne nécessiterait qu'une itération pour $\varepsilon_{\text{préc.}} = 10^{-5}$.

Cependant la décomposition $\mathcal{F} - LU$ est suffisamment précise pour réduire le nombre d'itérations de la méthode itérative. On constate par ailleurs que le taux de compression de la décomposition est similaire quelle que soit la valeur de $\varepsilon_{\text{préc.}}$, et que le temps de décomposition en lui même reste du même ordre de grandeur pour les trois précisions étudiées.

	$\varepsilon_{\text{préc.}}$			Résolution
	10^{-3}	10^{-4}	10^{-5}	pleine
Assemblage (s)		107,795		238,564
Décomposition $\mathcal{F} - LU$ (s)	2767,84	3037,004	3122,484	$\emptyset$
GMRES (s)	1773,012	93,836	10,308	$\emptyset$
Temps total de résolution (s)	**4540,82**	**3135,94**	**3132,788**	**4616,112**
Nombre d'itérations	2147	165	10	$\emptyset$
Taux de compression $\mathcal{F} - LU$	0,238	0,258	0,277	$\emptyset$

TABLEAU 4.3.1 – Répartition des temps durant la résolution itérative préconditionnée du système creux ($N = 40$).

En revanche, le temps de résolution itérative est très amélioré pour $\varepsilon_{\text{préc.}} = 1 \times 10^{-5}$. Le temps supplémentaire utilisé pour la décomposition est donc largement compensé par le gain de temps dû à la diminution du nombre d'itérations, qui passent de 2147 pour $\varepsilon_{\text{préc.}} = 10^{-3}$ à seulement 10 pour $\varepsilon_{\text{préc.}} = 10^{-5}$.

Enfin, quelle que soit la précision choisie, le temps total de résolution est meilleur dans le cas creux que pour le système plein.

Dans la suite, nous résolvons donc le système en utilisant le solveur GMRES préconditionné avec $\varepsilon_{\text{préc.}} = 10^{-5}$.

Temps de résolution

Une fois les matrices F et $\widetilde{F}$ assemblées, nous pouvons résoudre les systèmes associés. Le système plein utilisera le solveur direct Gauss (issu de la librairie LAPACK), tandis que nous appliquons le GMRES préconditionné avec $\varepsilon_{\text{GMRES}} = \varepsilon_{\text{préc.}} = \varepsilon_{\text{ACA}} = 10^{-5}$. Les temps de résolution sont alors disponibles en figure 4.3.4.

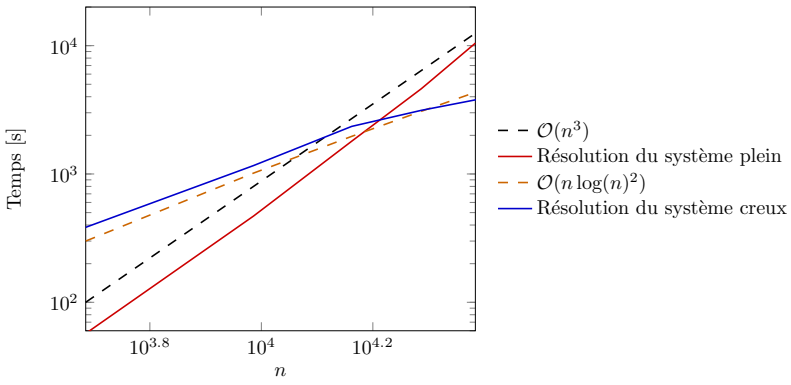


FIGURE 4.3.4 – Comparaison des temps de résolution pour F et $\widetilde{F}$.

On constate que le temps de résolution creux est plus faible que le temps de résolution par méthode de Gauss dès que $N = 40$, soit $n = 19360$. En particulier, le tableau 4.3.2 nous permet d'analyser la répartition du temps de résolution sur les deux étapes de ce solveur (décomposition $\mathcal{F} - LU$ et solveur itératif).

Le temps de résolution est alors surtout occupé par la décomposition $\mathcal{F} - LU$ de la

	$N = 20$	$N = 40$	$N = 60$
Décomposition $\mathcal{F} - LU$ (s)	1131,212	3122,480	4590,364
GMRES (s)	31,512	10,308	12,10
Nombre d'itérations	141	14	10

TABLEAU 4.3.2 – Répartition des temps durant la résolution itérative préconditionnée du système creux.

matrice $\widetilde{F}$. Cette décomposition semble plus précise pour les valeurs de N plus grandes ; en effet, la réduction du nombre d'itérations de GMRES indique un meilleur préconditionnement pour les grandes valeurs de N.

L'application du format $\mathcal{F}$-matrice aux PCFs semble donc réussie en terme de coût temporel. Voyons à présent si cela se vérifie sur la mémoire utilisée.

Performances mémoire

F et $\widetilde{F}$ peuvent toutes deux être stockées de manière réduite. Cependant, la résolution par méthode de Gauss du système plein nous contraint à prendre en compte le coût de la matrice F supposée dense. De même, la résolution par GMRES préconditionné nous impose de prendre en considération l'espace occupé par la décomposition $\mathcal{F} - LU$ de $\widetilde{F}$.

Afin de comparer la mémoire utilisée pour l'assemblage et la résolution dans les deux cas, il suffit de comparer la somme des taux de compression de $\widetilde{F}$ et de sa décomposition $\mathcal{F} - LU$ avec le taux de coefficients non-nuls de F. Cette comparaison est effectuée en figure 4.3.5.

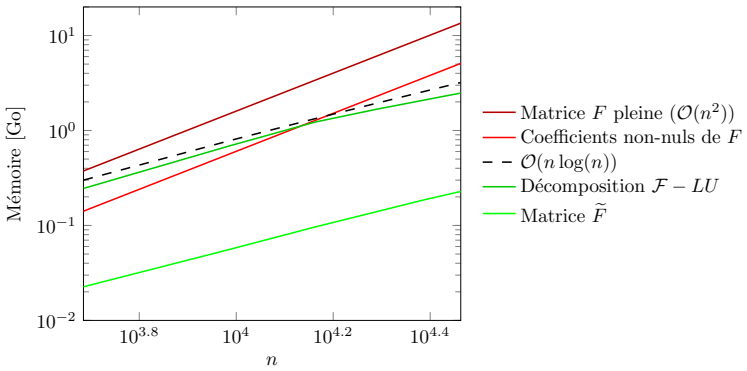


FIGURE 4.3.5 – Comparaison de la mémoire utilisée par F, $\widetilde{F}$ et la décomposition $\mathcal{F} - LU$.

On constate que l'application du format hiérarchique, même partielle, permet de changer la complexité de l'espace mémoire utilisée, passant de $\mathcal{O}(n^2)$ à $\mathcal{O}(n\log(n))$. En particulier, le stockage de $\widetilde{F}$ est toujours inférieur à celui de F, même en n'en considérant que les coefficients non-nuls. La décomposition $\mathcal{F} - LU$ prend quant à elle un espace mémoire très supérieur à $\widetilde{F}$, mais cette RAM est propotionnelle à $n\log(n)$ et est inférieure au stockage plein de F (nécessaire à la résolution directe par méthode de Gauss).

Cette application nous a donc permis de constater que bien qu'il ne soit pas immédiatement possible de mettre toute la matrice F au format $\mathcal{H}$-matrice en vue de la résolution d'un système, il est en revanche possible d'appliquer ce format compressé à certaines parties

d'une plus grande matrice. Une telle manipulation permet d'améliorer le temps d'assemblage et de résolution en changeant leur complexité, permettant là encore le traitement de plus grands systèmes. Ceci permet alors de discrétiser plus finement certains problèmes, et donc d'en avoir une résolution plus précise.

Conclusion du chapitre

Nous avons pu, grâce à ces trois cas pratiques, constater que le format $\mathcal{H}$-matrice est applicable sur divers problèmes physiques concrets, tels que les calculs de SER en aéronautique ou sur des surfaces rugueuses, ou encore la détection de modes de propagation dans des PCFs.

En particulier, la parallélisation sur architecture partagée de certaines opérations peut aider à réduire encore les temps engagés lors de la résolution de grand systèmes linéaires, comme nous avons pu le vérifier dans le cas de l'avion. Les surfaces de Weierstrass nous ont permis de jauger le rôle de la complexité géométrique d'une structure sur le comportement de l'assemblage et de la résolution. Enfin, le cas des PCFs, bien que plus en marge de notre étude initiale, montre que le format $\mathcal{H}$-matrice peut être utilisé comme brique de base pour aider à la résolution de problèmes plus larges.

Conclusion

Durant cette thèse, notre objectif a été d'augmenter le nombre d'applications qu'il est possible de traiter par la méthode des éléments finis de frontière (en anglais *Boundary Element Method*, notée BEM) en réduisant son coût de résolution et en conservant une précision raisonnable. Il est alors possible d'appliquer la BEM à des cas encore inenvisageables il y a quelques temps, soit par leur nombre important de degrés de liberté (DDLs) , soit par un grands nombre de seconds membres.

Le chapitre 1 nous a permis de rappeler les équations de Maxwell et d'en écrire une formulation intégrale, l'Équation Intégrale en Champ Électrique (en anglais *Electric Field Integral Equation*, notée EFIE), afin d'y appliquer la méthode des éléments finis de frontière (BEM). Nous avons ensuite, au chapitre 2, défini le contexte théorique sur lequel se base le format $\mathcal{H}$-matrice, qui constitue le cadre des méthodes de compression utilisées. Notamment, le produit $\mathcal{H}$-matrice-vecteur passe d'une complexité de $\mathcal{O}(n^2)$ pour une matrice dense à $\mathcal{O}(n\log(n))$ pour une $\mathcal{H}$-matrice. Cette opération nous permet d'adapter le solveur itératif GMRES au formalisme des $\mathcal{H}$-matrices. Nous avons également développé une décomposition $\mathcal{H}$-*LU*, permettant une résolution directe, ainsi que le préconditionnement du solveur itératif GMRES.

Nous avons ensuite validé l'association de ce format avec la méthode de compression Approximation en Croix Adaptative (en anglais *Adaptive Cross Approximation*, notée ACA) sur des cas d'application canoniques, puis nous avons introduit l'Approximation en Croix Hybride (HCA). Cette méthode permet de pallier à certains problèmes posés par l'ACA, et notamment le manque de fiabilité dans certains cas de la méthode d'estimation *a priori* de l'erreur d'approximation. Nous avons également constaté que pour le nombre de points d'intégration de Gauss que nous avons choisi d'appliquer à nos calculs ($n_{\text{Gauss}} = 3$), l'HCA est plus rapide que l'ACA pour une précision comparable. Après en avoir détaillé les principes d'application sur l'EFIE, nous avons validé cette méthode de compression encore rare en électromagnétisme, en établissant des règles d'accord entre la précision souhaitée et le nombre de points d'interpolation de Tchebychev nécessaires à réaliser la première étape de cette méthode. Pour finir nous avons également adapté des méthodes de résolution au format $\mathcal{H}$-matrice : d'une part le solveur direct $\mathcal{H}$-*LU*, et d'autre part le solveur itératif GMRES, éventuellement aidé d'un préconditionneur obtenu par décomposition $\mathcal{H}$-*LU* d'une version recompressée par *coarsening* de la $\mathcal{H}$-matrice à traiter. Nous avons pu vérifier que les résultats de ces méthodes, appliquées à des cas canoniques répondaient à nos attentes. La validation des méthodes de compression et de résolution nous permet d'aborder des cas d'applications plus complexes que ces deux cas canoniques.

Nous avons alors pu tester ces méthodes sur divers cas plus concrets. Le premier cas d'application, un avion comportant un grand nombre de DDLs, nous a permis d'utiliser quelques résultats de parallélisation sur architecture partagée qu'il faudra toutefois approfondir. L'application du format $\mathcal{H}$-matrice aux surfaces de Weierstrass nous a permis d'observer les effets de la complexité d'une géométrie sur les temps d'assemblage et de résolution. En particulier, plus un cas est complexe et plus le solveur direct $\mathcal{H}$-*LU* est favorable en terme de temps, même si ce solveur peut nuire à la précision de la solution selon le conditionnement du système à résoudre. Enfin, le traitement d'une fibre à cristaux photoniques (en anglais *Photonic-Crystal Fiber*, notée PCF) nous a permis de nous rendre compte de la versatilité du format $\mathcal{H}$-matrice, qu'il est possible d'utiliser pour certains

blocs d'un système creux avec succès.

Perspectives

Les méthodes exploitées au cours de cette thèse ont fait leurs preuves sur quelques exemples, mais il serait intéressant d'en étendre les possibilités.

Notre première priorité sera d'améliorer les performances de la parallélisation de nos opérations hiérarchiques sur architecture partagée. Ce procédé offre en théorie des possibilités que nous n'avons pu qu'effleurer en pratique. Une étude plus approfondie des problèmes d'échange de données entre les processeurs pourrait permettre, au prix de quelques modifications dans le code initial, de profiter pleinement des possibilités offertes par ce processus.

Par ailleurs, nous nous sommes limités au cas de la formulation EFIE ; le développement de la formulation Équation Intégrale en Champ Magnétique (en anglais *Magnetic Field Integral Equation*, notée MFIE) en $2D$ et en $3D$ nous permettrait d'utiliser la formulation Équation Intégrale en Champs Combinés (en anglais *Combined Field Integral Equation*, notée CFIE) et d'appliquer l'HCA sur des noyaux plus complexes. En particulier nous pourrions l'appliquer à des cas sur lesquels l'ACA a été mise en défaut, comme c'est le cas pour les noyaux comportant des opérateurs différentiels.

Par ailleurs, dans tous les cas que nous avons traités (à part les PCFs), nous avons considéré que l'objet diffractant était parfaitement métallique. L'introduction de matériaux diélectriques permettraient d'étendre considérablement le champ des applications possibles.

Annexes

A Résolution d'un système linéaire

La résolution d'un problème électromagnétique par la méthode des éléments finis de frontière (en anglais *Boundary Element Method*, notée BEM), aussi appelée méthode des moments (en anglais *Method of Moments*, notée MoM), peut s'avérer coûteuse. Il s'agit de résoudre le système linéaire

$$Z\boldsymbol{j} = \boldsymbol{e}, \tag{A.1}$$

avec $Z \in \mathbb{C}^{n\times n}$.

On distingue alors les solveurs *directs* des solveurs *itératifs*.

A.1 Solveurs directs

Principe des solveurs directs

Ces solveurs reposent sur une factorisation de la matrice à inverser. Il peut s'agir d'une factorisation LU ou encore d'une factorisation LDL^T [19] pour les matrices symétriques définies positives. Le cas de la factorisation LU est détaillée dans la partie suivante.

La factorisation permet alors d'écrire la matrice à inverser sous une forme qui en facilite la résolution. Cette première étape est suivie d'une étape de résolution.

Si la complexité algorithmique de la factorisation est en $\mathcal{O}(n^3)$, la résolution a quant à elle un coût numérique en $\mathcal{O}(n^2)$.

De tels solveurs sont notamment appréciés parce qu'ils ne génèrent pas d'approximations autres que celles dues à la discrétisation du problème et à la précision machine. De ce fait, la solution de l'équation (1.1.81) sera précise, à condition que la matrice Z soit bien conditionnée.

Par ailleurs, en terme de temps, la plus grande part du calcul d'un solveur direct est occupée par la factorisation de la matrice. Une fois celle-ci effectuée, il devient aisé de résoudre le système linéaire pour un nombre de seconds membres $p \geqslant 1$. La complexité algorithmique de la résolution d'un seul second membre étant de $\mathcal{O}(n^2)$, elle passe à $\mathcal{O}(pn^2)$ pour p seconds membres. Le temps de résolution est donc relativement prévisible, ce qui ne sera pas le cas avec les solveurs itératifs.

Solveur direct *LU*

Le solveur LU est un solveur direct qui suit donc les deux étapes précédemment citées. La première étape consiste à effectuer la décomposition LU de la matrice Z.

Définition A.1 (Décomposition LU d'une matrice)**.** *On appelle* décomposition LU *d'une matrice* $A \in \mathbb{C}^{n\times n}$ *toute écriture de* A *sous la forme d'un produit* $A = LU$ *où :*

- L *est une matrice triangulaire inférieure (*Lower triangular matrix*);*
- U *est une matrice triangulaire supérieure (*Upper triangular matrix*).*

Remarque A.1. *Une matrice* $A \in \mathbb{C}^{n\times n}$ *n'admet pas toujours une décomposition* LU*. Cependant dans certains cas, en permutant des lignes de* A*, la décomposition devient possible. On obtient alors une décomposition de la forme*

$$A = PLU, \tag{A.2}$$

où P *est une matrice de permutation.*

Les décompositions LU *et* PLU *conduisent à des matrices* L *et* U *distinctes. Par abus de langage, nous appelerons factorisation* LU *le processus de décomposition* PLU*.*

Remarque A.2. *On peut par ailleurs imposer que la diagonale de L soit unitaire ; de ce fait, lorsqu'une décomposition LU existe, il y a unicité de la décomposition.*

On considère donc le système d'équations

$$\begin{cases} Z = LU, & \text{(A.3)} \\ Z\boldsymbol{x} = \boldsymbol{b}. & \text{(A.4)} \end{cases}$$

On peut résoudre l'équation (A.4) en deux fois, en utilisant l'équation (A.3) :

- d'une part, on résout
$$L\boldsymbol{y} = \boldsymbol{b}, \tag{A.5}$$
qui est un système linéaire triangulaire inférieur ;
- d'autre part, on résout
$$U\boldsymbol{x} = \boldsymbol{y}, \tag{A.6}$$
qui est un système linéaire triangulaire supérieur.

Il est possible d'optimiser la décomposition LU, notamment en développant une version hiérarchique, dont nous parlons en détails en partie 2.5.4.

A.2 Solveurs itératifs

Principe des solveurs itératifs

Les solveurs itératifs permettent de résoudre un système linéaire en n'effectuant pas d'opérations qui soient numériquement plus lourdes que le produit matrice-vecteur.

Soit $n_{\text{ité.}}$ le nombre d'itérations nécessaires pour que le solveur converge vers une solution acceptable. Alors la complexité algorithmique de la résolution est de $\mathcal{O}(n_{\text{ité.}}n^2)$. De fait, contrairement aux solveurs directs, on peut difficilement prévoir à l'avance le temps nécessaire à la résolution du système. On montre toutefois que l'on atteint une résolution exacte du système linéaire en au plus $n_{\text{ité.}} = n$ itérations. Ce résultat, valable en arithmétique exacte, n'est cependant pas toujours valable dès que la précision machine entre en jeu.

Il est également possible de définir un critère d'arrêt de la méthode itérative $\varepsilon_{\text{ité.}}$, ce qui permet de réduire le nombre d'itérations nécessaires à la convergence de la résolution. Cette réduction du temps de résolution se fait évidemment au prix de la précision sur le résultat final. On s'attend en effet à une erreur de l'ordre de $\varepsilon_{\text{ité.}}$. Il s'agit donc ici de faire un compromis entre rapidité de convergence et précision souhaitée sur le résultat.

L'une des plus importantes limitations de certaines méthodes itératives est le traitement d'un cas de résolution à plusieurs seconds membres. En effet, la résolution doit alors avoir lieu entièrement pour chacun des seconds membres. De ce fait les méthodes itératives ne semblent pas à privilégier s'il y a plusieurs seconds membres dans le système à résoudre. Il existe toutefois des méthodes itératives capable de résoudre simultanément les différents problèmes définis par les seconds membres.

De manière générale, le temps de convergence vers une solution précise dépend du conditionnement de la matrice (*cf.* définition A.2) : plus son conditionnement est élevé, plus le nombre d'itérations nécessaires pour atteindre la convergence est important. Cette dépendance au conditionnement de la matrice est l'un des plus gros points faibles des méthodes itératives. Pour palier à ce problème, il existe des méthodes pour améliorer les conditionnement du système linéaire : les *préconditionneurs*.

Préconditionnement des méthodes itératives

Définition A.2 (Conditionnement d'une matrice [20])**.** *Soit A une matrice de $\mathbb{C}^{n\times n}$. Le conditionnement de A est le réel noté $\kappa(A)$ et défini par*

$$\kappa(A) = \|A\| \, \|A^{-1}\| \,, \tag{A.7}$$

où $\|\cdot\|$ est une norme matricielle.

On appelle alors *préconditionneur* à droite (resp. à gauche) de $A \in \mathbb{C}^{n\times n}$ toute matrice $P \in \mathbb{C}^{n\times n}$ telle que le conditionnement de la matrice $A' = P^{-1}A$ (resp. $A' = P^{-1}A$) soit plus petit que celui de A.

Considérons P un préconditionneur à gauche d'une matrice A. la résolution du système (1.1.81) se ramène donc à la résolution de

$$P^{-1}A\boldsymbol{x} = P^{-1}\boldsymbol{b}, \tag{A.8}$$

ce qui permet de réduire le nombre d'itérations nécessaires à la convergence du solveur itératif utilisé. En pratique, on essaie de trouver un préconditionneur P qui soit proche de A, et donc tel que le nouveau système linéaire à résoudre soit proche de

$$\boldsymbol{x} = A^{-1}\boldsymbol{b}. \tag{A.9}$$

En général, on n'explicite ni P, ni même P^{-1}, mais on effectue uniquement les produits matrice-vecteur nécessaires à la résolution. Il existe plusieurs techniques de préconditionnement, parmi lesquelles on peut notamment citer les plus populaires.

Tout d'abord, la méthode inverse creux approché (en anglais *Sparse Approximate Inverse*, noté SPAI) [18], est une technique qui consiste à trouver une matrice creuse P et minimise la quantité $\|P^{-1}A - U\|_{\mathrm{F}}$, où $\|\cdot\|_{\mathrm{F}}$ est la norme de Frobenius (norme sur les espaces matriciels, *cf.* définition 1.4.2). En théorie, on peut choisir P aussi creuse que l'on veut. En pratique, un niveau trop restrictif de parcimonie n'est pas souhaitable pour que le préconditionnement soit efficace ; inversement, une matrice qui n'est pas assez creuse va augmenter le coût de calcul, le rapprochant de celui engrangé pour le calcul de l'inverse de la matrice A. Il faut donc trouver un compromis raisonnable.

Mentionnons également le préconditionneur *Sparse LU*, qui consiste en la réalisation d'une factorisation LU creuse de la matrice. L'équation (A.9) revient alors à la résolution successive de deux systèmes triangulaires. Nous détaillons ce mode de préconditionnement dans le cadre des $\mathcal{H}$-matrices en section 3.3.

D'autres méthodes de préconditionneme,t telles que la méthode (block-)diagonal ou (Block-)Jacobi [20] sont également très populaires, mais ne sont pas nécessairement applicables au cas des équations intégrales.

Généralisation de la Méthode de Minimisation du Résidu (GMRES)

La méthode GMRES, développée par Yousef Saad et Martin H. Schultz en 1985 [45], est devenue un standard des solveurs itératifs.

La méthode est décrite pour la résolution du système linéaire

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{A.10}$$

avec $\boldsymbol{x}$ et $\boldsymbol{b}$ dans $\mathbb{C}^n$ et A dans $\mathbb{C}^{n\times n}$. On suppose de plus que A est inversible (autrement dit, que le système admet une solution) et que $\|\boldsymbol{b}\|_2 = 1$. On choisit une tolérance $\varepsilon_{\mathrm{GMRES}} > 0$. Cette tolérance limitera l'erreur de précision.

On définit pour $1 \leqslant k \leqslant n$ le $k^{\text{ième}}$ espace de Krylov par

$$K_k = \text{Vect}\{(A^{l-1}\boldsymbol{b})_{1\leqslant l\leqslant k}\}. \tag{A.11}$$

La méthode donne alors une approximation de la solution exacte de l'équation (A.10) par le vecteur $\boldsymbol{x}_k$ de K_k qui minimise la quantité $\|A\boldsymbol{x}_k - \boldsymbol{b}\|_2$.

On garantit que l'espace K_k est libre en utilisant la méthode d'Arnoldi [20] pour trouver les vecteurs orthonormaux $(\boldsymbol{q}_l)_{l\in[\![1,k]\!]}$ qui constituent la base K_k. Ainsi il existe un vecteur $\boldsymbol{y}_k$ de $\mathbb{C}^k$ tel que $\boldsymbol{x}_k = Q_k\boldsymbol{y}_k$, avec $Q_k \in \mathbb{C}^{m\times k}$ la matrice dont les colonnes sont les vecteurs $(\boldsymbol{q}_l)_{l\in[\![1,k]\!]}$.

La méthode d'Arnoldi engendre alors la matrice de Heisenberg supérieure $\widetilde{H}_k$ de taille $k(k+1)$ et telle que

$$AQ_k = Q_{k+1}\widetilde{H}_k. \tag{A.12}$$

La matrice Q_k étant orthogonale, on a

$$\|A\boldsymbol{x}_k - \boldsymbol{b}\|_2 = \|\widetilde{H}_k\boldsymbol{y}_k - \beta\boldsymbol{e}_1\|_2 \tag{A.13}$$

où $\boldsymbol{e}_1$ est le premier vecteur de la base canonique de $\mathbb{R}^{k+1}$, et

$$\beta = \|\boldsymbol{b} - A\boldsymbol{x}_0\|_2\,, \tag{A.14}$$

où $\boldsymbol{x}_0$ est le vecteur d'initialisation (en pratique, on prend $\boldsymbol{x}_0 = \boldsymbol{e}_1$). Ainsi, $\boldsymbol{x}_k$ peut être trouvé en minimisant la norme du résidu

$$\boldsymbol{r}_k = \widetilde{H}_k\boldsymbol{y}_k - \beta\boldsymbol{e}_1. \tag{A.15}$$

C'est un problème linéaire de moindres carrés de taille k.

Ainsi, chaque itération de l'algorithme consiste à :

1. effectuer une étape de l'algorithme d'Arnoldi ;
2. trouver $\boldsymbol{y}_k$ qui minimise $\|\boldsymbol{r}_k\|_2$;
3. calculer $\boldsymbol{x}_k = Q_k\boldsymbol{y}_k$;
4. recommencer tant que le résidu est plus grand que $\varepsilon_{\text{GMRES}}$.

À chaque itération, un produit matrice-vecteur $A\boldsymbol{q}_k$ doit est effectué. Cela génère un coût en calcul de $2n^2$ opérations pour les matrices pleines de taille n. Cependant ce coût peut être réduit en fonction du nombre d'itérations $n_{\text{ité.}}$ nécessaires à arriver à convergence.

Remarque A.3. *La méthode GMRES est souvent* restartée. *Après un nombre n_r d'itérations, la solution obtenue est récupérée comme point de départ pour démarrer un nouveau cycle de n_r itérations. Cette méthode permet de ne pas stocker $n_{ité}$ vecteurs dans la base, qui en possédera donc un maximum de n_r.*

L'algorithme A.1 donne les étapes de ce solveur muni d'un *restart*.

Nous voyons dans la partie 2.5.2 que ce solveur s'adapte bien au format des matrices hiérarchiques car la seule opération arithmétique nécessaire pour la formuler est le produit $\mathcal{H}$-matrice-vecteur.

Algorithme A.1 Résolution GMRES du système $\widetilde{Z}_{|b}\boldsymbol{j} = \boldsymbol{e}$.

procedure GMRES($Z_{|b}$,$\boldsymbol{j}$,$\boldsymbol{e}$,$\varepsilon_{\text{GMRES}}$)
  $n_{\text{its}} = 0$ ▷ Initialisation du nombre d'itérations.
  $\boldsymbol{j}_0 \coloneqq \boldsymbol{0}$
  call MAT_VECT($Z_{|b}, \boldsymbol{j}_0, \boldsymbol{e}_0$) ▷ Produit matrice-vecteur.
  $\boldsymbol{r}_0 \coloneqq \boldsymbol{e} - \boldsymbol{e}_0$
  $\beta \coloneqq \|\boldsymbol{r}_0\|_2$
  $\boldsymbol{r}_1 \coloneqq \boldsymbol{r}_0/\beta$
  for $j \in [\![1, n_r]\!]$ **do** ▷ Début de l'algorithme d'Arnoldi.
    for $i \in [\![1, j]\!]$ **do**
      call MAT_VECT($Z_{|b}, \boldsymbol{j}, \boldsymbol{j}'$) ▷ Produit matrice-vecteur.
      $h_{ij} \coloneqq \boldsymbol{j}' \cdot \boldsymbol{i}$ ▷ Coefficient de la matrice de Hessemberg $\bar{H_{n_r}} = (h_{ij})_{i\in[\![1,n_r+1]\!],j\in[\![1,n_r]\!]}$.
    end for
    $\boldsymbol{w}_j \coloneqq \boldsymbol{j}' - \sum_{i=1}^{j} h_{ij}\boldsymbol{i}$
    $h_{j+1,j} \coloneqq \|\boldsymbol{w}_j\|_2$
    if $h_{j+1,j} = 0$ **then**
      stop
    end if
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}_j/h_{j+1,j}$
  end for ▷ Fin de l'algorithme d'Arnoldi.
  Générer $\boldsymbol{y}_{n_r}$ qui minimise $\|\beta\boldsymbol{e}_1 - Z_{|b}\boldsymbol{y}_{n_r}\|_2$
  $\boldsymbol{j}_{n_r} = \boldsymbol{j}_0 + V_{n_r}\boldsymbol{y}_m$ ▷ Avec $V_m = (\boldsymbol{v}_i)_{i\in[\![1,n_r]\!]}$.
  $n_{its} \coloneqq n_{its} + 1$
  if $\boldsymbol{r}_0/\beta < \varepsilon_{\text{GMRES}}$ ou $n_{its} \geqslant n_r$ **then**
    stop
  else
    $\boldsymbol{j}_0 \coloneqq \boldsymbol{j}_m$
    go to 8 ▷ *Restart* si le résidu dépasse la tolérance $\varepsilon_{\text{GMRES}}$.
  end if
end procedure

B Nombre d'opérations arithmétiques et complexité algorithmique

Dans cette annexe, nous exprimerons le nombre d'opérations élémentaires (somme et multiplications) nécessaires à l'exécution d'opérations usuelles plus complexes.

On se place dans le corps $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$.

B.1 Sommes

Le nombre d'opérations de la somme de deux éléments est tout simplement égale au nombre d'unités de mémoires à sommer. On en déduit le nombre d'opérations élémentaires et la complexité algorithmique des opérations listées dans le tableau B.1.

	Nature de A	Nature de B	Nature de $A+B$	Nombre d'opérations	Complexité algorithmique
Vecteurs	$\mathbb{K}^n$	$\mathbb{K}^n$	$\mathbb{K}^n$	n	$\mathcal{O}(n)$
Matrices	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{m\times n}$ dense	mn	$\mathcal{O}(mn)$
	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{m\times n}$ diagonale	$\mathbb{K}^{m\times n}$ dense	$\min(m,n)$	$\mathcal{O}(\min(m,n))$
	$\mathbb{K}^{m\times n}$ diagonale	$\mathbb{K}^{m\times n}$ diagonale	$\mathbb{K}^{m\times n}$ diagonale	$\min(m,n)$	$\mathcal{O}(\min(m,n))$

Tableau B.1 – Nombre d'opérations et complexité algorithmique des sommes.

B.2 Produits

Soient $A = (a_{ij})_{i\in[\![1,m]\!],j\in[\![1,n]\!]} \in \mathbb{K}^{m\times n}$ et $B = (b_{ij})_{i\in[\![1,n]\!],j\in[\![1,p]\!]} \in \mathbb{K}^{n\times p}$ deux matrices, et $C = (c_{ij})_{i\in[\![1,m]\!],j\in[\![1,p]\!]} \in \mathbb{K}^{m\times p}$ le résultat de la multiplication AB. Alors

$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,p]\!] \quad c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}. \tag{B.1}$$

En particulier :

- Si A est diagonale, on peut écrire $A = \mathrm{diag}((\alpha_i)_{i\in[\![1,k]\!]})$ avec $k = \min(m,n)$. Alors (B.1) devient
$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,p]\!] \quad c_{ij} = \alpha_i b_{ij}, \tag{B.2}$$
ce qui revient à multiplier la k-ème ligne de B par a_{kk} pour obtenir la k-ème ligne de C.
- De même, si B est diagonale, $B = \mathrm{diag}((\beta_i)_{i\in[\![1,k]\!]})$ avec $k = \min(m,n)$ et (B.1) devient
$$\forall i \in [\![1,m]\!] \quad \forall j \in [\![1,p]\!] \quad c_{ij} = a_{ij}\beta_j, \tag{B.3}$$
ce qui revient à multiplier la k-ème colonne de A par b_{kk} pour obtenir la k-ème colonne de C.
- Si $p = 1$, alors on peut considérer que $B = (b_{ij})_{i\in[\![1,n]\!],j\in[\![1,1]\!]} = (b_i)_{i\in[\![1,n]\!]}$ et $C = (c_{ij})_{i\in[\![1,m]\!],j\in[\![1,1]\!]} = (c_i)_{i\in[\![1,m]\!]}$ sont des vecteurs, et (B.1) devient
$$\forall i \in [\![1,m]\!] \quad c_i = \sum_{k=1}^{n} a_{ik}b_k. \tag{B.4}$$

	Nature de A	Nature de B	Nature de AB	Nombre d'opérations	Complexité algorithmique
Matrice × Matrice	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{n\times p}$ dense	$\mathbb{K}^{m\times p}$ dense	$mp(2n-1)$	$\mathcal{O}(mnp)$
	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{n\times n}$ diagonale	$\mathbb{K}^{m\times n}$ dense	mn	$\mathcal{O}(mn)$
	$\mathbb{K}^{n\times n}$ diagonale	$\mathbb{K}^{n\times p}$ dense	$\mathbb{K}^{n\times p}$ dense	np	$\mathcal{O}(np)$
	$\mathbb{K}^{n\times n}$ diagonale	$\mathbb{K}^{n\times n}$ diagonale	$\mathbb{K}^{n\times n}$ diagonale	n	$\mathcal{O}(n)$
Matrice × Vecteur	$\mathbb{K}^{m\times n}$ dense	$\mathbb{K}^{n}$ vecteur	$\mathbb{K}^{m}$ vecteur	$m(2n-1)$	$\mathcal{O}(mn)$
	$\mathbb{K}^{m\times m}$ diagonale	$\mathbb{K}^{m}$ vecteur	$\mathbb{K}^{m}$ vecteur	m	$\mathcal{O}(m)$

Tableau B.2 – Nombre d'opérations et complexité algorithmique des multiplications.

Le tableau B.2 décrit le nombre d'opérations élémentaires et la complexité algorithmique des opérations de multiplications et de produits.

Remarque B.1. *Il peut également être nécessaire de connaître le coût de l'opérations $A = U\Sigma V^H$, avec $U \in \mathbb{K}^{m\times k}$, $\Sigma \in \mathbb{R}^{k\times k}$ et $V \in \mathbb{K}^{n\times k}$. D'après le tableau B.2, on obtient le nombre d'opérations*

$$n_{opé.} = mk(2k-1) + nk, \tag{B.5}$$

soit une complexité algorithmique en $\mathcal{O}(mk^2 + nk)$. Il s'agit donc d'une opération peu coûteuse si $k \ll m, n$.

B.3 Factorisations

La factorisation QR et la SVD réduites sont moins élémentaires que les opérations décrites ci-avant, mais constituent des briques élémentaires de certains de nos algorithmes. Gene H. Golub et Charles F. van Loan en donnent le coût numérique théorique dans [17] (*cf.* tableau B.3). Pour ces deux opérations, on a $n \geqslant m$.

Factorisation	Nature de la matrice	Nombre d'opérations	Complexité algorithmique
QR réduite	$\mathbb{K}^{m\times n}$ dense	$4n^2m - \frac{4}{3}n^3$	$\mathcal{O}(mn^2)$
SVD réduite	$\mathbb{K}^{m\times n}$ dense	$14mn^2 + 8n^3$	$\mathcal{O}(mn^2)$

Tableau B.3 – Nombre d'opérations et complexité algorithmique des factorisations réduites.

En particulier, si $m = n$, ces factorisations sont proportionnelles à n^3. Nous veillerons donc à ne les réaliser que sur des matrices de petites dimensions.

C Introduction au calcul parallèle

Traditionnellement, la simulation numérique se traduit dans un premier temps par des algorithmes séquentiels : chaque problème est divisé en une liste d'instructions qui sont exécutées en série, l'une après l'autre, chacune sur un unique processeur. Seule une instruction peut être exécutée à la fois.

Or en calcul scientifique, la modélisation demande des ressources en mémoire vive et en temps de calcul de plus en plus importantes, et ces ressources sont vite limitées dans le cas d'un programme séquentiel. Pour répondre à cette demande, le marché de l'informatique a connu des progrès rapides, si bien qu'aujourd'hui même le plus basique des ordinateurs possède au moins deux processeurs et permet donc de paralléliser les tâches qui peuvent s'y prêter.

Dans un contexte d'optimisation numérique, se limiter à n'utiliser qu'un processeur revient donc à ne pas utiliser pleinement le potentiel de notre outil de calculs.

C.1 Définition générale du parallélisme

Un *ordinateur parallèle* est un ordinateur qui possède une architecture constituée de plusieurs processeurs pouvant participer simultanément à l'exécution d'une tâche. On évalue la performance d'une architecture parallèle en fonction des performances indépendantes de chacun de ses processeurs, ainsi que de la manière dont ils sont agencés.

Une telle architecture permet, en théorie, d'augmenter la mémoire disponible à mesure que le nombre de processeurs est élevé. Toujours en théorie, il n'y a pas de limitation quant au nombre de processeurs que l'on peut utiliser lors d'un calcul, si ce n'est le nombre de processeurs disponibles. La division des tâches sur plusieurs unités de calcul permet par ailleurs l'accélération de calculs coûteux en temps ; cette division est d'autant plus efficace lorsque les données traitées peuvent être gérées indépendamment et sont stockées dans un ensemble structuré. C'est en particulier le cas des $\mathcal{H}$-matrices, qui semblent donc se prêter au calcul parallèle.

C.2 Limites du parallélisme

Les algorithmes parallèles possèdent toutefois certaines limitations qu'il est important de signaler.

Tout d'abord, lorsque l'on traite simultanément plusieurs données, il faut s'assurer de leur indépendance. Aussi, si certaines données à traiter ne répondent pas à ce critère, il convient d'ordonner les instructions afin de ne pas modifier simultanément deux données interdépendantes.

Par ailleurs il n'est pas toujours avantageux de paralléliser une application. Les communications et d'autres opérations annexes peuvent dans certains cas augmenter le temps d'exécution par processeur. Il convient donc de vérifier que ce facteur ne soit pas un frein avant d'entreprendre la parallélisation d'un algorithme. Un programme est dit *extensible* si sa mise en parallèle permet effectivement un gain de temps.

C.3 Efficacité du parallélisme

Il existe plusieurs facteurs permettant d'évaluer les performances d'un algorithme parallèle en comparaison avec son équivalent séquentiel.

Définition C.1 (Débit de traitement)**.** *On appelle* débit de traitement*, noté P, le nombre d'opérations exécutables par unité de temps, soit en notant $n_{\text{op.}}$ le nombre d'opérations et*

t le temps total d'exécution,

$$P = \frac{n_{\text{op.}}}{t}. \tag{C.1}$$

Définition C.2 (Accélération)**.** *Afin de quantifier le gain en terme de temps d'un algorithme parallèle par rapport à son homologue séquentiel, on définit l'*accélération*, notée a, en fonction du temps d'exécution parallèle* $t_{\text{para.}}$ *et du temps d'exécution séquentiel* $t_{\text{séq.}}$*, par*

$$a = \frac{t_{\text{séq.}}}{t_{\text{para.}}}. \tag{C.2}$$

Définition C.3 (Efficacité)**.** *On définit alors l'*efficacité *d'un code parallèle, notée e, par*

$$e = \frac{a}{n_{\text{proc.}}}, \tag{C.3}$$

où $n_{\text{proc.}}$ *désigne le nombre de processeurs utilisés.*

La parallélisation d'un code est considérée parfaite si l'accélération a est égale au nombre de processeurs utilisés pour le calcul, ce qui revient à une efficacité maximale $e = 1$.

C.4 Classification des architectures parallèles

Le parallélisme se manifeste soit en superposant les performances de plusieurs processeurs séquentiels, soit en exécutant simultanément des instructions indépendantes.

Il existe différents types d'architectures parallèles. On utilise communément la *Taxonomie de Flynn* pour distinguer ces architectures selon le type d'organisation du flux de données et du flux d'instructions. Cette classification est résumée dans le tableau C.1.

	Single instruction	*Multiple instructions*
Single data	SISD	MISD
Multiple data	SIMD	MIMD

Tableau C.1 – Récapitulatif de la taxonomie de Flynn.

SISD (*Single Instruction on Single Data*)

Dans cette architecture, une seule donnée est traitée par une unique instruction à un instant t. Une telle architecture est en fait un ordinateur séquentiel qui n'exploite pas de parallélisme, que ce soit au niveau des instructions ou de la mémoire. On l'appelle également *architecture de von Neumann*, que l'on peut schématiser comme indiqué en figure C.1a.

MISD (*Multiple Instructions on Single Data*)

Il s'agit d'un ordinateur dans lequel une donnée unique est traitée par plusieurs unités de calcul en parallèle. Cette architecture est assez rarement implémentée en pratique (*cf.* figure C.1b).

SIMD (*Single Instruction on Multiple Data*)

Dans ce second cas, plusieurs données sont traitées simultanément par une unique instruction. Il s'agit d'un ordinateur qui utilise le parallélisme au niveau de la mémoire. Cette architecture est schématisée en figure C.1c.

MIMD (*Multiple Instructions on Multiple Data*)

Dans ce dernier cas, plusieurs unités de calcul, possédant chacune une mémoire propre, traitent des données différentes. Il s'agit de l'architecture parallèle la plus utilisée. Il en existe en particulier deux variantes.

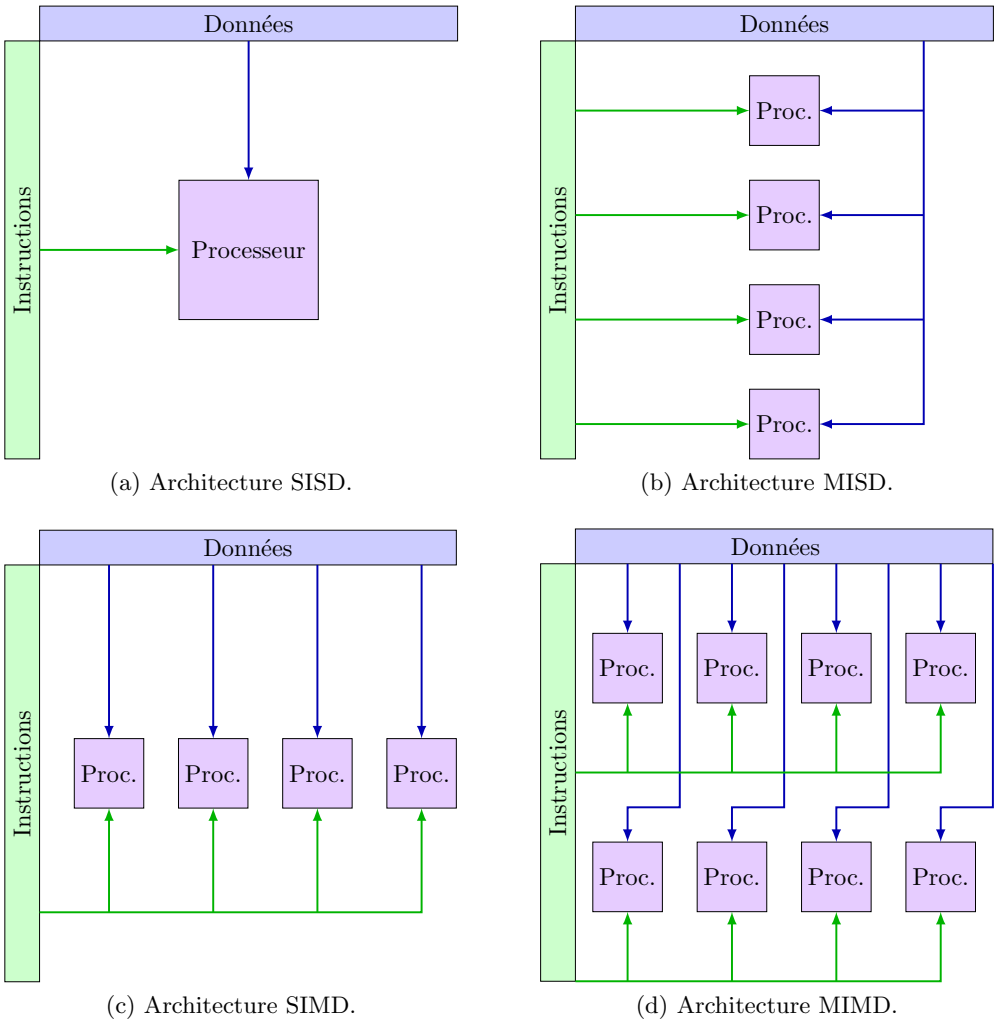
D'une part, l'architecture *MIMD à mémoire partagée* est telle que les unités de calcul ont accès à la mémoire comme un espace d'adressage global. Tout changement dans une case mémoire est vu par les autres unités de calcul. La communication entre les unités de calcul est effectuée via la mémoire globale.

D'autre part, dans l'architecture *MIMD à mémoire distribuée*, chaque unité de calcul possède sa propre mémoire et son propre système d'exploitation. Ce second cas de figure nécessite des outils supplémentaires, appelés *middleware*, pour synchroniser les données et communiquer entre les différents processeurs.

Les superordinateurs utilisent souvent une architecture *MIMD hybride*, étant à la fois à mémoire partagée et distribuée. Ces systèmes hybrides possèdent l'avantage d'être très extensibles, performants et à faible coût.

Dans le cas général, on peut représenter l'architecture MIMD comme schématisé en figure C.1d.

Dans notre cas, nous disposons d'une architecture MIMD à mémoire partagée sur 16 processeurs, avec 192 Go de RAM. Dans le cas d'une mémoire partagée, l'utilisation d'instructions OpenMP permet de parvenir à une parallélisation du code de façon relativement aisée. C'est l'option que nous avons choisie.



(a) Architecture SISD.

(b) Architecture MISD.

(c) Architecture SIMD.

(d) Architecture MIMD.

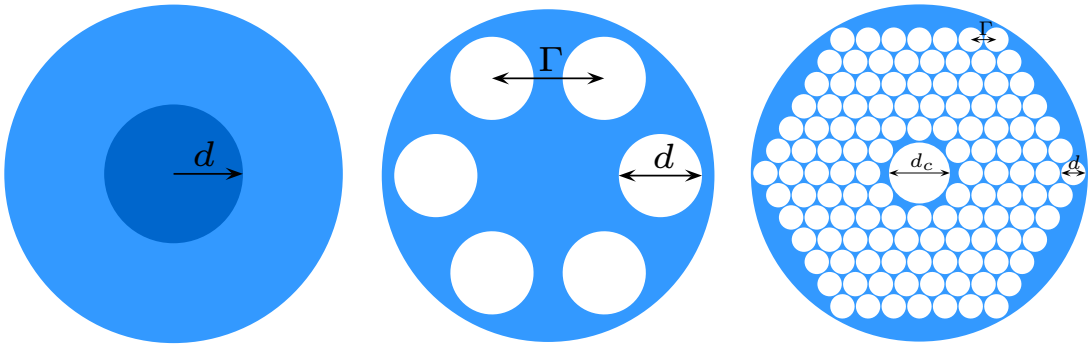
FIGURE C.1 – Différents types d'architectures parallèles.

D Fibres à cristaux photoniques (PCFs)

Les fibres à cristaux photoniques (PCFs) [46] sont très utilisées en optique, pour des applications diverses. Leurs propriétés optiques dépendent fortement de la géométrie et de ses caractéristiques diélectriques (indices de réfraction $n = \sqrt{\varepsilon_r} \geqslant 1$). On considérera dans notre cas que les PCF sont constituées d'une gaine et d'une ou plusieurs inclusions ou cœurs. La propagation de la lumière peut se faire d'au moins deux manières différentes :

- Si l'indice de réfraction du cœur n_i est supérieur à celui de la gaine n_e, alors le champ électromagnétique sera confiné à l'intérieur du cœur.
- Si $n_i < n_e$, alors le champ se déplacera dans la gaine. On utilise alors plusieurs inclusions pour créer un espace (cercle, hexagone, *etc.*) dans lequel le champ est concentré et puisse se déplacer dans une direction donnée.

Il est important de connaître/calculer l'indice de réfraction effectif n_{eff} qui caractérise la propagation dans la PCF et qui tient compte des matériaux, de la géométrie et de la source. Des exemples de fibre en coupe sont représentés sur la figure D.1.



(a) Fibre optique à saut d'indice (b) PCF à 6 trous en hexagone (c) PCF à 121 trous en nid d'abeille

FIGURE D.1 – Représentation de plusieurs PCF, vue en coupe.

Afin de pouvoir étudier la propagation de la lumière dans des PCFs qui peuvent faire intervenir des inclusions de formes, de tailles et de matériaux différents, on privilégie l'utilisation de méthodes numériques afin d'obtenir rapidement le résultat souhaité. Le développement de ce code a été mené par Julien Vincent et Adrien Calvez au laboratoire LAPLACE [47]. La méthode des équations intégrales de frontière pour la modélisation de fibres à cristaux photoniques y est abordée. Afin d'obtenir diverses solutions pour comparer les résultats obtenus, d'autres méthodes numériques et logiciels sont également présentés. Avant cela il faut poser correctement le problème, c'est l'objet de la partie suivante.

D.1 Problème de transmission

On considère une fibre de longueur infinie orientée selon $\boldsymbol{u}_z$ et invariante suivant z, dont la coupe transversale est représentée sur la figure D.2.

Le milieu Ω_0 est considéré comme homogène infini et chaque milieu fermé $\underset{i\geqslant 1}{\Omega_i}$ est caractérisé par son indice de réfraction n_i. Les vecteurs localement normaux et tangentiels aux surfaces sont respectivement définis par ν et τ.

Dans le plan transverse (O,$\boldsymbol{u}_x$, $\boldsymbol{u}_y$), le champ électrique $\boldsymbol{E}$ transverse et le champ magnétique $\boldsymbol{H}$ transverse doivent satisfaire l'équation d'Helmholtz. Ils doivent également

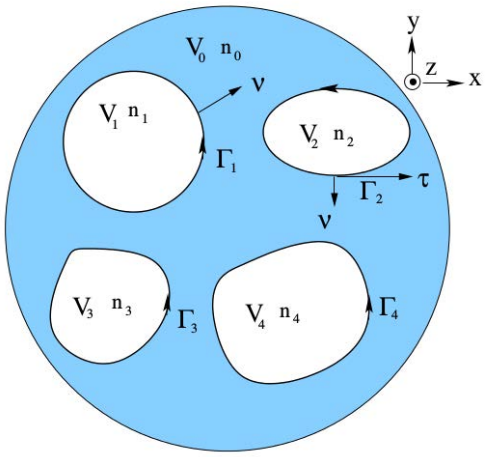


FIGURE D.2 – Schéma du problème de transmission.

satisfaire les conditions de transmission à travers les frontières Γ qui font intervenir les dérivées normales et tangentielles de $\boldsymbol{E}$ et $\boldsymbol{H}$. Ceci permet d'écrire les systèmes (D.1) et (D.2)) suivants :

- dans $\bigcup_{i\geqslant 0}\Omega_i$:

$$\begin{cases} \left[\Delta + (k^2 - \beta^2)\right] \boldsymbol{E} & = 0, \\ \left[\Delta + (k^2 - \beta^2)\right] \boldsymbol{H} & = 0, \end{cases} \tag{D.1}$$

- à travers $\Gamma = \bigcup_{i\geqslant 1}\Gamma_i$:

$$\begin{cases} [\boldsymbol{E}]_\Gamma & = 0, \\ [\boldsymbol{H}]_\Gamma & = 0, \\ \left[\frac{n_{eff}}{n^2 - n_{eff}^2}\partial_\tau \boldsymbol{E}\right]_\Gamma & = \left[\frac{1}{n^2 - n_{eff}^2}\partial_\nu \boldsymbol{H}\right]_\Gamma, \\ \left[\frac{n_{eff}}{n^2 - n_{eff}^2}\partial_\tau \boldsymbol{H}\right]_\Gamma & = -\left[\frac{n^2}{n^2 - n_{eff}^2}\partial_\nu \boldsymbol{E}\right]_\Gamma, \end{cases} \tag{D.2}$$

avec les grandeurs suivantes :

- $k_0 = \frac{2\pi}{\lambda_0}$ la constante de propagation de l'onde magnétique dans le vide, avec λ_0 la longueur d'onde dans le vide ;
- n_j l'indice de réfraction du milieu dans Ω_j ;
- $k = k_0 n_j$ la constante de propagation du milieu dans Ω_j ;
- $\beta = k_0 n_{eff}$

D.2 Équation intégrale de frontières pour les PCF

On reprend ici la méthode décrite dans [48] et [49]. L'écriture des potentiels de simple et de double couches à l'intérieur et à l'extérieur des inclusions, permet d'établir les relations entre les champs électrique et magnétique ainsi que leurs dérivées normales. Ces opérateurs étant singuliers lorsque l'on considère deux points sur la même frontière d'une inclusion (sur la diagonale de la matrice), on utilisera la décomposition établie par Kress [50] pour outrepasser le problème.

Représentation des champs sur les surfaces

Dans un domaine Ω_l fermé, de frontière $\partial\Omega_l$ suffisamment régulière (*i. e.* lipschitzienne), une fonction u satisfaisant l'équation d'Helmholtz s'écrit avec la formule de Green :

$$u(\mathbf{r}) = \int_{\partial\Omega_l} \left[G(\mathbf{r},\tilde{\mathbf{r}})\partial_\nu u(\tilde{\mathbf{r}}) - \frac{\partial G(\mathbf{r},\tilde{\mathbf{r}})}{\partial\nu(\tilde{\mathbf{r}})} u(\tilde{\mathbf{r}}) \right] ds(\tilde{\mathbf{r}}) \tag{D.3}$$

avec $\mathbf{r} \in \Omega_l$, ν est la normal sortante à la surface $\partial\Omega_l$ et G la fonction de Green :

$$G(\mathbf{r},\tilde{\mathbf{r}}) = \frac{j}{4} H_0^{(1)}(k_i||\mathbf{r}-\tilde{\mathbf{r}}||). \tag{D.4}$$

Construction du système à résoudre. On commence par considérer l'intérieur des inclusions. En dérivant (D.4) par rapport au vecteur normal et par passage à la limité, on obtient l'équation intégrale de frontière suivante :

$$(Id - J_i)\partial_{\nu i} u = -W_i u \qquad \text{sur } \partial\Omega_l \tag{D.5}$$

où J_i et W_i sont les opérateurs singuliers suivants :

$$\begin{cases} (J_i\psi)(\mathbf{r}) & = 2\displaystyle\int_{\partial\Omega_l} \frac{\partial G(\mathbf{r},\tilde{\mathbf{r}})}{\partial\nu(\mathbf{r})}\psi(\tilde{\mathbf{r}})ds(\tilde{\mathbf{r}}) \ , \ \mathbf{r} \in \partial\Omega_l \\ (W_i\psi)(\mathbf{r}) & = 2\displaystyle\int_{\partial\Omega_l} \frac{\partial^2 G(\mathbf{r},\tilde{\mathbf{r}})}{\partial\nu(\mathbf{r})\partial\nu(\tilde{\mathbf{r}})}\psi(\tilde{\mathbf{r}})ds(\tilde{\mathbf{r}}) \ , \ \mathbf{r} \in \partial\Omega_l \end{cases} \tag{D.6}$$

Pour l'extérieur des inclusions, c'est-à-dire le milieu Ω_0, il est nécessaire de prendre en compte les interactions qui existent entre toutes les inclusions. En considérant un nombre C d'inclusions, on peut écrire, selon la même démarche réalisée précédemment, l'équation intégrale de frontière suivante :

$$(Id + J_e)\partial_{\nu e} u = W_e u \tag{D.7}$$

où J_e et W_e sont les opérateurs singuliers suivants :

$$\begin{cases} (J_e\psi)(\mathbf{r}) & = 2\displaystyle\sum_{l=1}^{C}\int_{\partial\Omega_l} \frac{\partial G(\mathbf{r},\tilde{\mathbf{r}})}{\partial\nu(\mathbf{r})}\psi(\tilde{\mathbf{r}})ds(\tilde{\mathbf{r}}) \\ (W_e\psi)(\mathbf{r}) & = 2\displaystyle\sum_{l=1}^{C}\int_{\partial\Omega_l} \frac{\partial^2 G(\mathbf{r},\tilde{\mathbf{r}})}{\partial\nu(\mathbf{r})\partial\nu(\tilde{\mathbf{r}})}\psi(\tilde{\mathbf{r}})ds(\tilde{\mathbf{r}}) \end{cases} \tag{D.8}$$

En utilisant ces deux relations pour le champ électrique E et le champ magnétique H en surface, et les deux relations de passage dans (D.2), on obtient le système à six inconnues suivant :

$$\begin{bmatrix} W_e & 0 & -(Id+J_e) & 0 & 0 & 0 \\ W_i & Id-J_i & 0 & 0 & 0 & 0 \\ 0 & -\frac{n_i^2}{n_i^2-n_{eff}^2}Id & \frac{n_e^2}{n_e^2-n_{eff}^2}Id & \left(\frac{n_{eff}}{n_e^2-n_{eff}^2}-\frac{n_{eff}}{n_i^2-n_{eff}^2}\right)T & 0 & 0 \\ 0 & 0 & 0 & W_e & 0 & -(Id+J_e) \\ 0 & 0 & 0 & W_i & Id-J_i & 0 \\ \left(\frac{n_{eff}}{n_e^2-n_{eff}^2}-\frac{n_{eff}}{n_i^2-n_{eff}^2}\right)T & 0 & 0 & 0 & \frac{1}{n_i^2-n_{eff}^2}Id & -\frac{1}{n_e^2-n_{eff}^2}Id \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{E} \\ \partial_{\nu i}\boldsymbol{E} \\ \partial_{\nu e}\boldsymbol{E} \\ \boldsymbol{H} \\ \partial_{\nu i}\boldsymbol{H} \\ \partial_{\nu e}\boldsymbol{H} \end{bmatrix} = 0 \tag{D.9}$$

où l'opérateur T permet de faire la différenciation de E et de H pour travailler sur les champs tangentiels. Tout comme les opérateurs W_i et J_i, T intervient uniquement lorsque l'on considère une même inclusion, ce sont des opérateurs diagonaux par bloc.

Si on prend un exemple de deux inclusions notées A et B, il faudra considérer 4 interactions pour construire W_e (idem pour J_e) : celle de A sur A, A sur B, B sur A et B sur B. Pour l'opérateur W_i, chaque conclusion ne considère qu'elle même (idem pour J_i et T), dans ce cas on aura uniquement A sur A et B sur B.

$$W_e = \begin{bmatrix} W_e^{AA} & W_e^{AB} \\ W_e^{BA} & W_e^{BB} \end{bmatrix} \quad \text{et} \quad J_e = \begin{bmatrix} J_e^{AA} & J_e^{AB} \\ J_e^{BA} & J_e^{BB} \end{bmatrix} \tag{D.10}$$

$$W_i = \begin{bmatrix} W_i^{AA} & 0 \\ 0 & W_i^{BB} \end{bmatrix} \quad \text{et} \quad J_i = \begin{bmatrix} J_i^{AA} & 0 \\ 0 & J_i^{BB} \end{bmatrix} \quad \text{et} \quad T = \begin{bmatrix} T^{AA} & 0 \\ 0 & T^{BB} \end{bmatrix}. \tag{D.11}$$

Discrétisation On reprend ici la discrétisation mise au point par Kress [50] à l'aide d'une décomposition des noyaux et d'une méthode de Nyström (ou méthode de quadrature). À l'intérieur des inclusions, on doit isoler l'hypersingularité pour réaliser la discrétisation. on considère la réprésentation paramétrique suivante, de la surface $\partial\Omega_l$:

$$\mathbf{r}(t) = [x(t), y(t)], \qquad 0 \leqslant t \leqslant 2\pi \tag{D.12}$$

où l'angle t est compté dans le sens trigonométrique, $x(t)$ et $y(t)$ sont des fonctions analytiques et 2π-périodiques de t et $|\mathbf{r}'(t)| > 0$. Avec cette notation, les opérateurs W_i et J_i peuvent s'écrirent

$$(W_i u)(\mathbf{r}(t)) = \frac{1}{|\mathbf{r}'(t)|} \int_0^{2\pi} \left\{ \frac{1}{2\pi} \cot\left(\frac{t-\tilde{t}}{2}\right) u'(\mathbf{r}(\tilde{t})) - Q(t,\tilde{t}) u(\mathbf{r}(\tilde{t})) \right\} d\tilde{t} \tag{D.13}$$

$$(J_i \partial_\nu u)(\mathbf{r}(t)) = \int_0^{2\pi} \frac{|\mathbf{r}'(\tilde{t})|}{|\mathbf{r}'(t)|} P(t,\tilde{t}) \partial_\nu u(\mathbf{r}(\tilde{t})) d\tilde{t} \tag{D.14}$$

Les expressions complètes de P et de Q sont données dans [50] et dans les annexes de [49]. En décomposant les singularités logarithmiques de P et Q et en faisant intervenir des formules de quadratures [50] (attention : les formules de $R_j(t)$ et de $T_j(t)$ données dans [48] sont fausses, utiliser plutôt celles de [49]), on obtient le système matriciel suivant :

$$(Id - J_i)\partial_{\nu i}\mathbf{u} = -W_i \mathbf{u}, \tag{D.15}$$

où $\partial_{\nu i}\mathbf{u}$ et $\mathbf{u}$ désignent les vecteurs colonnes des surfaces de toutes les inclusions $\partial_{\nu i} u(\mathbf{r}(t_l))$ et $u(\mathbf{r}(t_l))$. si N est le nombre de point de discrétisation par trou et C le nombre d'inclusion, ces vecteurs colonnes sont de taille CN.

Pour écrire les matrices des opérateurs W_e et J_e qui correspondent à l'extérieur des inclusions, il faut considérer deux cas selon les inclusions. Si on considère une même inclusion (*i. e.* sur les blocs qui constituent la diagonale de la matrice), il faut reprendre le schéma précédemment écrit pour l'intérieur. Informatiquement, on peut utiliser une même fonction pour créer ses blocs car seule la valeur de l'indice de réfraction peut varier.

Dans le cas où on considère deux inclusions A et B, les interactions des N points de l'inclusion B sur le point $\mathbf{r}_0$ de l'inclusion A seront calculés sous la forme des intégrales suivantes :

$$(W\partial_{\nu e}u)(\mathbf{r}_0) = 2\int_0^{2\pi} \frac{\partial G(\mathbf{r}_0, \mathbf{r}(t))}{\partial \nu(\mathbf{r}_0)} \partial_{\nu e}u(\mathbf{r}(t))|\mathbf{r}'(t)|dt \tag{D.16}$$

$$(Ju)(\mathbf{r}_0) = 2\int_0^{2\pi} \frac{\partial^2 G(\mathbf{r}_0, \mathbf{r}(t))}{\partial \nu(\mathbf{r}_0)\partial\nu(\mathbf{r})} u(\mathbf{r}(t))|\mathbf{r}'(t)|dt \tag{D.17}$$

Les deux intégrales précédentes sont approximées par une méthode des trapèzes, on obtient donc :

$$(W\partial_{\nu e}u)(\mathbf{r}_0) \approx \frac{4\pi}{N}\sum_{j=0}^{N-1} \frac{\partial G(\mathbf{r}_0, \mathbf{r}(t_j))}{\partial \nu(\mathbf{r}_0)} \partial_{\nu e}u(\mathbf{r}(t_j))|\mathbf{r}'(t_j)| \tag{D.18}$$

$$(Ju)(\mathbf{r}_0) \approx \frac{4\pi}{N}\sum_{j=0}^{N-1} \frac{\partial^2 G(\mathbf{r}_0, \mathbf{r}(t_j))}{\partial \nu(\mathbf{r}_0)\partial\nu(\mathbf{r})} u(\mathbf{r}(t_j))|\mathbf{r}'(t_j)| \tag{D.19}$$

Les expressions des dérivées partielles de la fonction de Green sont calculées analytiquement d'après (D.4) comme suit :

$$\frac{\partial G(\mathbf{r}, \tilde{\mathbf{r}})}{\partial \nu(\mathbf{r})} = -\frac{jk_e}{4} H_1^{(1)}(k_e||\mathbf{r} - \tilde{\mathbf{r}}||) \frac{(\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nu(\mathbf{r})}{||\mathbf{r} - \tilde{\mathbf{r}}||} \tag{D.20}$$

$$\begin{aligned} \frac{\partial^2 G(\mathbf{r}, \tilde{\mathbf{r}})}{\partial \nu(\mathbf{r})\partial\nu(\tilde{\mathbf{r}})} &= -\frac{jk_e^2}{4} H_0^{(1)}(k_e||\mathbf{r} - \tilde{\mathbf{r}}||) \frac{(\tilde{\mathbf{r}} - \mathbf{r}) \cdot \nu(\tilde{\mathbf{r}}) \times (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nu(\mathbf{r})}{||\mathbf{r} - \tilde{\mathbf{r}}||^2} \\ &+ \frac{jk_e}{4} H_1^{(1)}(k_e||\mathbf{r} - \tilde{\mathbf{r}}||) \left(\frac{\nu(\tilde{\mathbf{r}}) \cdot \nu(\mathbf{r})}{||\mathbf{r} - \tilde{\mathbf{r}}||} + 2\frac{(\tilde{\mathbf{r}} - \mathbf{r}) \cdot \nu(\tilde{\mathbf{r}}) \times (\mathbf{r} - \tilde{\mathbf{r}}) \cdot \nu(\mathbf{r})}{||\mathbf{r} - \tilde{\mathbf{r}}||^3} \right) \end{aligned} \tag{D.21}$$

Simulations de PCF

Le système matriciel principal obtenu après discrétisation de toutes les frontières des inclusions est constitué de 6 relations entre les 6 inconnues $\boldsymbol{E}$, $\partial_{\nu i}\boldsymbol{E}$, $\partial_{\nu e}\boldsymbol{E}$, $\boldsymbol{H}$, $\partial_{\nu i}\boldsymbol{H}$ et $\partial_{\nu e}\boldsymbol{H}$. Dans le cas où on est en présence d'un mode de propagation, la valeur de n_{eff} permettra d'avoir le système suivant :

$$F(n_{eff}) \cdot \begin{bmatrix} \boldsymbol{E} \\ \partial_{\nu i}\boldsymbol{E} \\ \partial_{\nu e}\boldsymbol{E} \\ \boldsymbol{H} \\ \partial_{\nu i}\boldsymbol{H} \\ \partial_{\nu e}\boldsymbol{H} \end{bmatrix} = 0 \tag{D.22}$$

avec

$$F(neff) = \begin{bmatrix} W_e & 0 & -(Id + J_e) & 0 & 0 & 0 \\ W_i & Id - J_i & 0 & 0 & 0 & 0 \\ 0 & -\frac{n_i^2}{n_i^2 - n_{eff}^2} Id & \frac{n_e^2}{n_e^2 - n_{eff}^2} Id & \left(\frac{n_{eff}}{n_e^2 - n_{eff}^2} - \frac{n_{eff}}{n_i^2 - n_{eff}^2} \right) T & 0 & 0 \\ 0 & 0 & 0 & W_e & 0 & -(Id + J_e) \\ 0 & 0 & 0 & W_i & Id - J_i & 0 \\ \left(\frac{n_{eff}}{n_e^2 - n_{eff}^2} - \frac{n_{eff}}{n_i^2 - n_{eff}^2} \right) T & 0 & 0 & 0 & \frac{1}{n_i^2 - n_{eff}^2} Id & -\frac{1}{n_e^2 - n_{eff}^2} Id \end{bmatrix} \tag{D.23}$$

Pour résoudre ce système, on cherche les zéros de la fonction $f(n_{eff}))$ définie par

$$f(n_{eff}) = \frac{1}{\boldsymbol{\Psi}^T F^{-1}(n_{eff})\boldsymbol{\Phi}} \tag{D.24}$$

où $\boldsymbol{\Psi}$ et $\boldsymbol{\Phi}$ sont deux vecteurs aléatoires.

La taille du système à résoudre devient plus importante lorsque l'on considère 121 trous, le temps d'inversion du système augmente donc. Pour éviter cela, on manipule la matrice à six inconnues pour que le système n'en intègre plus que quatre sans inverser un seul opérateur. On obtient donc le système suivant :

$$F(neff) = \begin{bmatrix} W_e & -\frac{n_i^2}{n_e^2}\frac{n_e^2-n_{eff}^2}{n_i^2-n_{eff}^2}(Id+J_e) & \frac{n_{eff}}{n_e^2}\left(1+\frac{n_e^2-n_{eff}^2}{n_i^2-n_{eff}^2}\right)(Id+J_e)T & 0 \\ W_i & Id-J_i & 0 & 0 \\ n_{eff}\left(1+\frac{n_e^2-n_{eff}^2}{n_i^2-n_{eff}^2}\right)(Id+J_e)T & 0 & W_e & -\frac{n_e^2-n_{eff}^2}{n_i^2-n_{eff}^2}(Id+J_e) \\ 0 & 0 & W_i & Id-J_i \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{E} \\ \partial_{\nu i}\boldsymbol{E} \\ \boldsymbol{H} \\ \partial_{\nu i}\boldsymbol{H} \end{bmatrix} = 0 \tag{D.25}$$

Par souci de simplification d'écriture, nous écrirons cette matrice plus simplement sous la forme

$$F = \begin{bmatrix} W_e & K_e & K_eT & 0 \\ W_i & K_i & 0 & 0 \\ \alpha K_eT & 0 & W_e & \beta K_e \\ 0 & 0 & W_i & K_i \end{bmatrix}, \tag{D.26}$$

et notons

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{E} \\ \partial_{\nu i}\boldsymbol{E} \\ \boldsymbol{H} \\ \partial_{\nu i}\boldsymbol{H} \end{bmatrix}. \tag{D.27}$$

La section 4.3 s'attache à résoudre le système

$$F\boldsymbol{u} = \boldsymbol{\Phi}, \tag{D.28}$$

où $\boldsymbol{\Phi}$ est un vecteur aléatoire.

Bibliographie

[1] Roger F. Harrington. "Matrix Methods for Field Problems". In : *Proceedings of the IEEE* 55 (1967), p. 136–149.

[2] Jeonghwa Lee, Jun Zhang et Cai-Cheng Lu. "Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems". In : *Journal of Computational Physics* 185 (2003), p. 158–175.

[3] Vladimir Rokhlin. "Rapid Solution of Integral Equations of Classical Potential Theory". In : *Journal of Computational Physics* 60 (1983), p. 187–207.

[4] Nader Engheta et al. "The Fast Multipole Method (FMM) for Electromagnetic Scattering Problems". In : *IEEE Transactions on Antennas and Propagation* 40 (1992), p. 634–641.

[5] D. Brunner et al. "Comparison of the Fast Multipole Method with Hierarchical Matrices for the Helmholtz-BEM". In : *Computer Modeling in Engineering and Sciences* 58 (2010), p. 131–158.

[6] Mario Bebendorf. "Approximation of boundary element matrices". In : *Numerische Mathematik* 86 (2000), p. 565–589.

[7] Kezhong Zhao, Marinos N. Vouvakis et Jin-Fa Lee. "The Adaptive Cross Approximation Algorithm for Accelerated Method of Moments Computations of EMC Problems". In : *IEEE Transactions on Electromagnetic Compatibility* 47 (2005), p. 763–773.

[8] John Shaeffer. "Direct Solve of Electrically Large Integral Equations for Problem Sizes to 1M Unknowns". In : *IEEE Transactions on Antennas and Propagation* 56 (2008), p. 2306–2313.

[9] Steffen Börm et Lars Grasedyck. "Low-Rank Approximation of Integral Operators by Interpolation". In : *Computing* 72 (2004), p. 325–331.

[10] Steffen Börm et Lars Grasedyck. "Hybrid cross approximation of integral operators". In : *Numerische Mathematik* 101 (2005), p. 221–249.

[11] Steffen Börm, Lars Grasedyck et Wolfgang Hackbusch. 2003 revised in 2006.

[12] Jacques Hadamard. "Sur les problèmes aux dérivées partielles et leur signification physique". In : *Princeton University Bulletin* (1902), p. 49–52.

[13] David Colton et Rainer Kress. "Using fundamental solutions in inverse scattering". In : *Inverse Problems* 22 (2006), p. 49–69.

[14] Philip L. Huddleston, Louis N. Medgyesi-Mitschang et John M. Putnam. "Combined Field Integral Equation Formulation for Scattering by Dielectrically Coated Conducting Bodies". In : *IEEE Transactions on Antennas and Propagation* 34 (1986), p. 510–520.

[15] Jean-Claude Nédelec. *Ondes acoustiques et électromagnétiques. Équations intégrales.* École Polytechnique, 1996.

[16] Sergej Rjasanow et Olaf Steinbach. *The Fast Solution of Boundary Integral Equations.* Springer, 2007.

[17] Geen H. Golub et Charles F. van Loan. *Matrix Computations - third edition*. Johns Hopkins, 1996.

[18] Michele Benzi et Miroslav Tuma. "A comparative study of sparse approximate inverse preconditioners". In : *Applied Numerical Mathematics* 30 (1999), p. 305–340.

[19] Zhou Bing et S. A. Greenhalgh. "Finite element three-dimensional direct current resistivity modelling : accuracy and efficiency considerations". In : *Geophysical Journal International* 145 (2001), p. 679–688.

[20] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Siam, 2002.

[21] Wolfgang Hackbusch et Z. P. Nowak. "On the Fast Matrix Multiplicationin the Boundary Element Method by Panel Clustering". In : *Numerische Mathematik* 54 (1999), p. 463–491.

[22] S. A. Sauter. "Variable Order Panel Clustering". In : *Computing* 64 (2000), p. 223–261.

[23] Jean-René Poirier. "Modélisation électromagnétique des effets de rugosité surfacique". INSA, Toulouse, France, 2000.

[24] Axel Breuer, Pierre Borderies et Jean-René Poirier. "A Multilevel Implementation of the QR Compression for Method of Moments". In : *IEEE Transactions on Antennas and Propagation* 51 (2003), p. 2520–2522.

[25] Mario Bebendorf et Sergej Rjasanow. "Matrix Compression for the Radiation Heat Transfer in Exhaust Pipes". In : *Multifield Problems*. Springer, 2000, p. 183–192.

[26] Julien Maurin et al. "Domain Decomposition Method Using Integral Equations and Adaptive Cross Approximation IE-ACA-DDM for studying Antenna Radiation and Wave Scattering From Large Metallic Platforms". In : *IEEE Transactions on Antennas and Propagation* 63 (2015), p. 5698–5708.

[27] José M. Tamayo, Alexander Heldring et Juan M. Rius. "Multilevel Adaptive Cross Approximation (MLACA)". In : *IEEE Transactions on Antennas and Propagation* 59 (2011), p. 4600–4608.

[28] Claude Delannoy. *Programmer en Fortran 90. Guide complet*. Eyrolles, 1997.

[29] Alexandre Ern et Jean-Luc Guermond. *Theory and Practice of Finite Elements*. Springer, 2000.

[30] Mario Bebendorf. *Hierarchical Matrices - A Means to Efficiently Solve Elliptic Boundary Value Problems*. Lecture Notes in Computational Science and Engineering. Springer, 2008.

[31] Steffen Börm, Lars Grasedyck et Wolfgang Hackbusch. "Introduction to hierarchical matrices with applications". In : *Engineering Analysis with Boundary Elements* 27 (2003), p. 405–422.

[32] Steffen Börm. *Efficient Numerical Methods for Non-local Operators - H2-Matrix Compression, Algorithms and Analysis*. EMS Tracts in Mathematics 14, 2010.

[33] Lars Grasedyck et Wolfgang Hackbusch. "Construction and Arithmetics of H-Matrices". In : *Computing* 70 (2003), p. 295–334.

[34] Mario Bebendorf. "A Hierarchical LU Decomposition-based Preconditioners for BEM". In : *Computing* 74 (2005), p. 225–247.

[35] Lars Grasedyck. “Adaptive Recompression of H-Matrices for BEM”. In : *Computing* 74 (2005), p. 205–223.

[36] Kieran Delamotte. “Une étude du rang du noyau de l’équation de Helmholtz : application des H-matrices á l’EFIE”. Université Paris 13, Paris, France.

[37] Mario Bebendorf et Stefan Kunis. “Recompression techniques for adaptive cross approximation”. In : INS Preprint no. 0708. Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität, Bonn. 2007.

[38] Sadasiva M. Rao, Donald R. Wilton et Allen W. Glisson. “Electromagnetic Scattering by Surfaces of Arbitrary Shape”. In : *IEEE Transactions on Antennas and Propagation* 30 (1982), p. 409–417.

[39] Benoît Lizé. “Résolution directe rapide pour les éléments finis de frontière en électromagnétisme et en acoustique : H-Matrices. Parallélisme et applications industrielles.” Laboratoire Analyse, Géométrie et Applications, Université Paris-Nord - Paris XIII, 2014.

[40] Simon Tournier. “Contribution à la modélisation de la diffusion électromagnétique par des surfaces rugueuses à partir de méthodes rigoureuses”. Institut Supérieur de l’Aéronautique et de l’Espace.

[41] Giorgio Franceschetti, Maurizio Migliaccio et Daniele Riccio. “An electromagnetic fractal-based model for the study of fading”. In : *Radio Science* 31 (1996), p. 1749–1759.

[42] Stéphane Rouvier. “Utilisation des Fractales pour la Caractérisation Électromagnétique des Scènes Naturelles”. Université Paul Sabatier, Toulouse, France.

[43] K. Falconer. *Fractal Geometry : Mathematical Foundations and Applications*. John Wiley & Sons, 2013.

[44] Julien Vincent et al. “Accelerating Performances of a Waveguide Mode Solver Based on Boundary Integral Equations”. In : META’16. 2016.

[45] Yousef Saad. *GMRES : a Generalized Minimal Residual algorithm for solving non-symmetric linear systems*. 1985.

[46] P. St. J. Russell. “Photonic crystal fibers”. In : *Science* 299 (2003), p. 358–362.

[47] Adrien Calvez. *Méthodes intégrales numériques pour la modélisation de fibres à cristaux photoniques*. Rapport de Stage de Master 2. LAPLACE, 2015.

[48] W. Lu et Y. Y. Lu. “Efficient Boundary Integral Equation Method for Photonic Crystal Fibers”. In : *Journal of Lightwave Technology* 30.11 (Juin 2012), p. 1610–1616.

[49] W. Lu et Y. Y. Lu. “Efficient High Order Waveguide Mode Solvers Based on Boundary Integral Equations”. In : *Journal of Computational Physics* 272 (Avril 2014), p. 507–525.

[50] R. Kress. “On the Numerical Solution of a Hypersingular Integral Equation in Scattering Theory”. In : *Journal of Computational and Applied Mathematics* 61 (Juin 1995), p. 345–360.

Liste des algorithmes

Liste des tableaux

Liste des figures

Méthodes quasi-optimales pour la résolution des équations intégrales de frontière en électromagnétisme

Il existe une grande quantité de méthodes numériques adaptées d'une part à la modélisation, et d'autre part à la résolution des équations de Maxwell. En particulier, la méthode des éléments finis de frontière (BEM), ou méthode des Moments (MoM), semble appropriée pour la mise en équation des phénomènes de diffraction par des objets parfaitement conducteurs, en limitant le cadre de l'étude à la frontière entre l'objet diffractant et le milieu extérieur. Cette méthode mène systématiquement à la résolution d'un système linéaire dense, que nous parvenons à compresser en l'approchant numériquement par une matrice hiérarchique creuse, appelée $\mathcal{H}$-matrice. Cette approximation peut être complétée d'une ré-agglomération permettant d'améliorer la parcimonie de la $\mathcal{H}$-matrice et ainsi d'optimiser davantage la résolution du système traité. La hiérarchisation du système s'effectue en considérant la matrice traitée par blocs, que l'on peut ou non compresser selon une condition d'admissibilité. L'Approximation en Croix Adaptative (ACA) et l'Approximation en Croix Hybride (HCA) sont deux méthodes de compression que l'on peut alors appliquer aux blocs admissibles.

Le travail de cette thèse consiste dans un premier temps à valider le format $\mathcal{H}$-matrice en $2D$ et en $3D$ en utilisant l'ACA, puis d'y appliquer la méthode HCA, encore peu exploitée. Nous pouvons alors résoudre le système linéaire issu de la BEM en utilisant différents solveurs, directs ou non, adaptés au format hiérarchique. En particulier, nous pourrons constater l'efficacité du préconditionnement LU hiérarchique sur un solveur itératif. Nous pourrons alors appliquer ce formalisme au cas des surfaces rugueuses ou encore des fibres à cristaux photoniques (PCFs). Il sera également possible de paralléliser certaines opérations sur architecture partagée afin de réduire de nouveau le coût temporel et mémoire de la résolution.

Mots-clés : Méthodes numériques, équations intégrales de frontière, matrices hiérarchiques, *Adaptive Cross Approximation*, *Hybrid Cross Approximation*, parallélisation.

Quasi-optimal methods for solving integral equations in electromagnetics

A lot of numerical methods are available for the modelization as well as the solution of the Maxwell's equations. In particular the boundary element method (BEM), also known as Method of Moments (MoM), seems appropriate to put in equation the scattering problems by perfectly conducting objects, by restricting the study to the frontier between the diffracting object and its surrounding. This method automatically leads to a dense linear system which we are able to compress, numerically approaching it by a hierarchical sparse matrix, called $\mathcal{H}$-matrix. This approximation can be completed with a coarsening which enhance the sparsity of the $\mathcal{H}$-matrix and thus optimizes again the solution of the concerned system. The hierarchization of the system is done considering the concerned matrix by its blocks, which can or cannot be compressed according to an admissibility condition. The Adaptive Cross Approximation (ACA) and the Hybrid Cross Approximation (HCA) are among the possible compression methods available to compress the admissible blocks.

This PhD thesis first focuses on the validation of the $\mathcal{H}$-matrix format both in $2D$ and $3D$ using the ACA. We then apply to this format the HCA method, which is still quite unmined. Thus we can solve the linear system coming from the BEM using different direct and iterative solution methods which are adapted to suit the hierarchical format. In particular, we will observe the efficiency of the hierarchical LU preconditionning used to enhance an iterative solver. Thus we will be able to apply this formalism on cases such as rough surfaces or photonic crystal fibers (PCFs). It will also be possible to make some operations parallel in order to further reduce the time cost of the solution.

Keywords : *Numerical methods, boundary element method, hierarchical matrices, Adaptive Cross Approximation, Hybrid Cross Approximation, parallel computing.*