Université Mohamed Boudiaf - M'sila

FACULTE DE TECHNOLOGIE

DEPARTEMENT D'ELECTRONIQUE

Numéro de série……

Numéro d'inscription……

Thèse

Présentée pour l'obtention du diplôme de

DOCTORAT EN SCIENCE

Spécialité : Electronique

Option : Communication

THEME

CALCUL HAUTE PERFORMANCE D'ANTENNES MICROSTRIP PAR ÉLÉMENTS FINIES SUR CLUSTER

Présentée Par

Salah Eddine MEZAACHE

Soutenue le 06/06/2018

Devant le jury composé de :

Nom & Prénom	Grade	Etablissement	Qualité
SAIGAA Djamel	**Professeur**	Uni. de **M'sila**	Président
BOUTT Farid	**Professeur**	Uni. de **Bordj Bou Arréridj**	Encadreur
MESSALI Zoubeida	**Professeur**	Uni. de **Bordj Bou Arréridj**	Examinateur
DJOUANE Lotfi	**MCA**	Uni. de **M'sila**	Examinateur
ROUABAH Khaled	**MCA**	Uni. de **Bordj Bou Arréridj**	Invité

Année Universitaire : 2017/2018

Résumé

Dans pratiquement tous les domaines, chercheurs et industriels se rendent compte que le calcul haute performance prend une place essentielle pou rester dans la course mondiale de la recherche et de l'industrie. L'accroissement de la complexité des calculs scientifiques et le besoin croissant en termes de précision font émerger des solutions logiciels et matériels pour le calcul intensif.

Dans cette thèse, nous nous sommes intéressés à l'application des techniques de calcul parallèles aux problèmes de calculs en électromagnétiques. Dans une première étape, nous avons construit un cluster de calcul Beowulf composé d'un nœud principal et de huit nœuds de calcul en utilisant des ordinateurs standards équipés de processeurs Intel® Core™ i5 et 8Gb de RAM interconnectés par un réseau Gigabit. Pour l'exploitation du cluster, nous avons utilisé la solution open source ROCKS basée sur un système de type Linux. Cette distribution contient tous les outils de base pour la maintenance, la gestion, la mise à jour et le déploiement des clusters de calcul haute performance. Ensuite, Trois applications de calcul électromagnétiques ont été développées et implémentées sur le cluster en utilisant parallèle MATLAB. Dans la première application, les caractéristiques d'une antenne micro-bande circulaire multicouches opérant en large bande sont calculées. La parallélisation est effectuées en utilisant le modèle de programmation SPMD. Nous avons ensuite développé et implémenter une méthode systématique pour générer et calculer des expressions sous format symbolique des fonctions de base d'ordre supérieur de la FEM. La dernière application concerne la parallélisation de la méthode des moments spatiale pour le calcul d'un réseau d'antennes 2D de type bowtie. L'utilisation du cluster nous a permis d'augmenter la taille du problème et de réduire le temps de calcul jusqu'à ici impossible à effectuer sur un seul ordinateur.

Ce travail montre l'utilité des clusters Beowulf à moindre coût en conjonction avec MATLAB, un outil très appréciés des scientifiques et ingénieurs et les résultats obtenus montrent un réel gain en performances et en productivité.

Mots-clé :

Calcul électromagnétique, calcul parallèle, calcul haute performance, calcul symbolique, SPMD, FEM ordre supérieur, réseaux antennes, antenne microstrip.

Abstract

In practically every field, researchers and industrials realize that high-performance computing is the key to stay in the global race for research and inovation. The increase in the complexity of scientific computations and the growing need for precision are leading to software and hardware solutions for intensive computing.

In this work, efficient high performance computation of electromagnetic field problems using a Beowulf cluster and MATLAB® is presented. A cluster with one master node, eight identical computing nodes, and a Gigabit Ethernet (GbE) switch is firstly built. An open source Rocks toolkit solution is used to simplify the process of deploying high-performance parallel computing clusters. Then, three computational electromagnetic applications are developed and implemented on the cluster using parallel MatLab. In the first application, is calculated the input impedance of a multilayer multiconductor circular microstrip antenna that is excited with coaxial probe and operating in a wide frequency band. Parallelization is then performed over the frequency vector permitting a speedup ratio of 23. In the second application, is presented a systematic method for generating and computing symbolic expressions of 2D/3D FEM basis functions of higher orders up to 10. The final application is dedicated to the parallelization of the spatial moment method code for 2D array of bowtie antenna. The simulation results have shown that the array size can be increased considerably leading to a linear system with a size up to 100 times greater than the one under study.

This work shows the usefulness of Beowulf clusters at a lower cost in conjunction with MATLAB, a tool highly appreciated by scientists and engineers, and the results obtained show a real gain in performance and productivity.

Keywords :

Computational electromagnetics, parallel computation, high performance computing, symbolique compuation, SPMD, High order FEM, array antenna, microstrip antenna

Remerciements

Je tiens à remercier mon directeur de thèse Pr. Farid BOUTTOUT pour sa disponibilité et la guidance qu'ils a pu m'apporter. Je le remercie surtout pour les heures passées à écrire et déboguer des programmes en MATLAB et les riches discussions autour des concepts de l'électromagnétisme. J'ai énormément apprécié ta rigueur scientifique , et te serais toujours gré de toutes les connaissances que tu m'as apportées.

Je tiens à remercier les membres de mon jury qui ont accepté de s'intéresser à mon travail malgré leurs emplois du temps surchargés. Je leur en suis très reconnaissant. Je tiens à remercier Djamel SAGAI, professeur à l'université de M'Sila, pour m'avoir fait l'honneur de présider ce jury de thèse. Je voudrais, exprimer mes remerciements à Zoubeida MESSALI, professeur de l'univeristé de Bordj Bou Arréridj, et à Lotfi DJOUANE, Maitre de conférence à l'université de M'Sila, pour leurs rapports détaillés témoignant de l'intérêt porté à ces travaux.

Un très grand merci à Khaled ROUABAH, un ami et un frère qui par ses aides techniques, ses conseils et son soutien m'a grandement aidé pour la publication des mes travaux et dans le processus de soutenance.

Je remercie particulièrement Salim ATIA pour ces conseils et sa relecture des publications en anglais.

Je remercie tous les amis qui se sont intéressés à l'avancement de mes travaux et spécialement mon ami et frère Abdenour HACINE GHARBI et tous mes collègues du département d'électronique de l'université de Bordj Bou Arréridj.

Je remercie enfin ma famille et mes amis pour m'avoir soutenue tout au long de ma thèse et tous ceux qui ont fait le déplacement, parfois de loin pour venir assister à ma soutenance.

A mes parents, Je n'aurais pu réussir sans eux, et je tiens ici à les remercier. Pour ma mère décédée, qu'Allah lui pardonne et la place à un rang élevé, parmi ceux qui ont été guidés. Merci père de m'avoir toujours poussé dans mes intérêts.

Enfin, ma femme. Je ne te remercierais jamais assez pour tout ce que tu as fait pour moi. Dans les pires moments de ma vie, j'ai toujours pu compter sur toi. Je voulais que tu saches à quel point ton soutien a été d'une grande aide pour moi. Alors du fond du cœur... merci pour ton soutien.

Table des matières

Table des matières **v**

Table des figures **ix**

Liste des tableaux **xii**

Nomenclature **xiii**

Introduction **1**

1 Le calcul haute performance **6**

1.1 Introduction au calcul parallèle 6

1.2 Principes du parallélisme 9

1.3 Architectures des calculateurs parallèles 9

1.3.1 Taxonomie de Flynn 9

1.3.2 Architecture Matériels des calculateurs parallèles 11

1.4 Modèles de programmation pour les architectures parallèles 14

1.4.1 Modèle en mémoire partagée 15

1.4.2 Modèle en mémoire distribuée 17

1.5 Mesure de performances des architectures multiprocesseurs 20

1.5.1 Modèle de calcul à durée égale 20

1.5.2 Modèle du Calcul parallèle avec des sections sérielles 21

1.6 MATLAB et calcul parallèle 23
1.7 Conclusion 24

2 Calcul Scientifique en électromagnétique 26
2.1 Introduction 26
2.2 Classification des problèmes électromagnétiques 28
2.3 Les équations intégrales 29
2.4 Transformation d'une équation différentielle en une équation intégrale . . 30
2.5 Les fonctions de Green 31
2.6 Les méthodes classiques pour les problèmes aux limites 32
2.6.1 La méthode de Rayleigh-Ritz 33
2.6.2 La méthode de Galerkin 35
2.7 La méthode des moments (MoM) 36
2.7.1 Principe de la méthode 36
2.8 La méthode des différences finies dans le domaine temporelle (FDTD) . . . 38
2.8.1 Les différences finies - un aperçu 38
2.8.2 Principe de la méthode FDTD 39
2.8.3 Algorithme de Yee 40
2.8.4 Précision et stabilité 46
2.8.5 Les conditions absorbante - PML 46
2.9 La méthode des éléments finis (FEM) 49
2.9.1 Les étapes élémentaires de l'analyse par FEM 50
2.9.2 Formulation du système d'équations 52
2.9.3 Solution du système d'équations 54
2.9.4 Aspects calculatoires 55
2.10 Calcul parallèle en électromagnétisme - biographie et état de l'art 56
2.11 Conclusion 61

3 Antenne circulaire multicouches et multiconducteurs 62
3.1 Introduction 62

3.2 Formulation mathématique 63
3.3 Résultats et discussions 66
3.4 Implémentation parallèle 67
3.5 Étude des performances réseaux 71
3.6 Conclusion 73

4 Fonctions de forme FEM d'ordre supérieur 75
4.1 Introduction 75
4.2 Fonctions d'interpolation de base et matrices fondamentales 76
4.2.1 Élément triangulaire pour les fonctions d'interpolation 2D 77
4.2.2 Élément tétraédrique pour les fonctions d'interpolation 3D 78
4.3 Évaluation des matrices d'éléments en coordonnées locales 79
4.4 Implémentation parallèle 81
4.4.1 Méthode 1 pour le calcul de la matrice 82
4.4.2 Méthode 2 pour le calcul de la matrice 83
4.5 Résultats et discussions 84
4.6 Conclusion 86

5 Analyse d'un réseaux d'antennes de type bowtie 88
5.1 Introduction 88
5.2 Structure de l'antenne 89
5.3 Méthode d'analyse et Matrice d'impédance 89
5.4 Processus de parallélisation 90
5.4.1 Partie#1 : calcul des caractéristiques des arêtes 91
5.4.2 Partie#2 : Calcul de la matrice impédance 93
5.4.3 Partie#3 : Détermination des courants et solution de l'équation MoM 97
5.5 Étude de la scalabilty (évolutivité) 99
5.6 Conclusion 101

Conclusion Générale 102

A L'interface de passage de message MPI 104

1.1 Introduction 104

1.2 Les communicateurs 104

1.2.1 Groupes de tâches 105

1.2.2 Communicateur par défaut 105

1.2.3 Rang des tâches 105

1.2.4 Groupe de communicateurs 106

1.3 Topologie virtuelle 107

1.3.1 La topologie cartésienne 107

1.3.2 Topologie en graphe 108

1.4 Les communications MPI 109

1.4.1 Communications point à point 110

1.4.2 Les communications en mode non bloquant 111

1.4.3 Synchronisation 112

1.4.4 Les communications collectives 113

B Rocks, une boite à outils open-source pour cluster réel et virtuel 117

2.1 Réalisation d'un cluster Beowulf 117

2.1.1 Les parties Matérielles 117

2.1.2 Les parties logicielles 117

2.2 La solution open source Rocks cluster 118

2.2.1 Noeud principal 118

2.2.2 Noeud de calcul 119

2.3 Réalisation du cluster Rocks-UBBA 119

2.3.1 Partie hardware 119

2.3.2 Installation du noeud principal 119

2.3.3 Installation des nœuds de calcul 120

2.3.4 Ganglia un outil de surveillance 120

2.4 Commandes utiles - administration cluster 122

Bibliography **139**

Table des figures

1 Le calcul haute performance **6**

1.1 Les quatres classes d'architectures de la taxonomie de Flynn 10

1.2 Le supercalculateur CDC 6600 12

1.3 Evolution des architectures du Top500 13

1.4 Le supercalculateur chinois Tianhe–2 13

1.5 Une grappe d'ordinateurs - cluster 14

1.6 Courbe de l'accélération idéale (sans surcoût des communications) selon la loi d'Amdahl pour différentes valeurs de f 22

1.7 Courbe de l'accélération avec prise en compte du surcoût des communications selon la loi d'Amdahl pour différentes valeurs de t_c/t_s pour $f = 0.05$. 23

1.8 Architecture générale d'un cluster Matlab 25

2 Calcul Scientifique en électromagnétique **26**

2.1 Position des composants du champ autour d'une cellule élémentaire du treillis de Yee 42

2.2 Transmission sans réflexion d'une onde plane à une interface PML/espace libre. 47

2.3 Éléments finis de base 51

3 Antenne circulaire multicouches et multiconducteurs **62**

3.1 Géométrie de l'antenne microbande circulaire à trois couches. 63
3.2 Coefficient de réflexion S_{11} en dB de l'antenne microruban multicouche en fonction de la fréquence en GHz pour différentes valeurs de d_4 67
3.3 Temps d'exécution vs nombre de workers. 69
3.4 Accélération vs nombre de workers. 70
3.5 Efficacité vs nombre de workers. 71
3.6 Schéma d'ajout d'un nouveau worker pour le scénario 1. 72
3.7 Schéma d'ajout d'un nouveau worker pour le scénario 2. 73
3.8 Temps d'exécution des deux scénarios en fonction du nombre de workers . 74

4 Fonctions de forme FEM d'ordre supérieur 75

4.1 Famille des éléments triangulaires : (a) Linéaire, (b) Quadratique , (c) Cubique. 77
4.2 Famille des éléments tétraèdres : (a) Linéaire, (b) Quadratique , (c) Cubique. 78
4.3 Distribution de la matrice suivant les colonnes 82
4.4 Distribution des éléments de la matrice en mode 1D 84
4.5 Temps d'exécution en fonction du nombre de workers 85
4.6 Accélération en fonction du nombre de workers 86
4.7 Efficacité en fonction du nombre de workers 87

5 Analyse d'un réseaux d'antenne de type bowtie 88

5.1 Structure du réseau d'antennes étudié par Makarov [156] 89
5.2 Antenne bowtie 90
5.3 Schéma d'un élément arête RWG et l'interprétation en dipôle 91
5.4 Organigramme du calcul des caractéristiques de l'antenne 92
5.5 Schémas de la numérotation des éléments du réseau d'antennes 93
5.6 Temps d"exécution de la partie 1 en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes 94
5.7 Accélération de la partie 1 en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes 95

5.8 Temps d'exécution de la partie calcul de la matrice d'impédance en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes. 96

5.9 Exemple de paramètres S pour quatre éléments en fonction de la fréquence d'un réseau Bowtie de 16 × 16 éléments 97

5.10 Diagramme de rayonnement de l'antenne de 16×16 éléments à la fréquence 160Mhz 98

5.11 Temps d"exécution de la partie#1 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers 99

5.12 Temps d'exécution de la partie#2 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers 100

5.13 Temps d"exécution de la partie#3 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers 101

A L'interface de passage de message MPI 104

A.1 Graphe de l'exemple étudié. 109

A.2 Synchronisation globale : `MPI_BARRIER()`. 112

A.3 Réduction répartie `MPI_Reduce()` avec l'opérateur somme et `MPI_AllReduce()` avec l'opérateur produit 113

A.4 Diffusion générale : `MPI_BCAST()`. 114

A.5 Diffusion sélective : `MPI_SCATTER()`. 115

A.6 Collecte : `MPI_GATHER()`. 116

B Rocks, une boite à outils open-source pour cluster réel et virtuel 117

B.1 Diagramme de connexion des éléments d'un cluster -Rocks- 118

B.2 Photographie du cluster Rocks-BBA 120

B.3 Capture d'écrans de la procédure d'installation du noeud principal sous Rocks 121

B.4 Procédure de capture de la demande d'installation du nœud de calcul . . . 121

B.5 Exemples de graphes du rapport de surveillance d'un noeud de calcul tirés de Ganglia 122

B.6 Exemple de mesures de l'activité réseaux d'un noeud de calcul 123

Liste des tableaux

2 Calcul Scientifique en électromagnétique **26**

2.1 Fonctions de Green dans l'espace libre en 3D. 33

3.1 Fréquences de résonances supérieures FU et inférieures FL de l'antenne pour différentes valeurs de d_4. 66

4 Fonctions de forme FEM d'ordre supérieur **75**

4.1 Les fonctions de formes sous format symbolique pour P=3,m=10 84

4.2 Les fonctions de formes sous format symbolique pour P=3,m=10 85

5 Analyse d'un réseaux d'antenne de type bowtie **88**

5.1 Impédance terminal ×100 à la fréquence $160 MHz\Omega$ 96

A L'interface de passage de message MPI **104**

A.1 Liste des fonctions de gestion des groupes MPI. 106

B Rocks, une boite à outils open-source pour cluster réel et virtuel **117**

B.1 Quelques commandes en ligne de commande pour l'administration d'un cluster Rocks. 123

Nomenclature

Abréviations

ABC Abosorbante Boundary Condition

CEM Computational Electromagnetics

DDM Domain Decomposition Method

EFIE Electric Field Integral Equation

EMC Eletromagnetic Compatibility

EMI Eletromagnetic Interference

FDTD Finite Difference Time Domain

FEM Finite Element Method

FMM Fast Multipole Method

HPC High Performance Computing

IE Integral Equation

MLFMA MultiLevel Fast Multipole Algorithm

MoM Method of Moment

MPI Message Passing Interface

PDE Partial Differential Equation

PML Perfectly matched layer

PVM Parallel Virtual Machine

RBC Radiation Boundary Conditions

RWG Rao Wilton Glisson

SPMD Single Program Multiple Data

Introduction

Les dispositifs et systèmes électromagnétiques, allant des appareils de bureau ordinaires aux téléphones cellulaires, font partie intégrante de la vie moderne. Le développement continu de la nouvelle technologie dépend grandement de l'analyse technique et de la synthèse des systèmes électromagnétiques. Ceux-ci sont basés sur l'obtention de solutions précises des équations de Maxwell pour le système étudié. L'intensification de la recherche et du développement dans une grande variété d'applications, notamment la conception d'antennes, les circuits hyperfréquences, la photonique, l'imagerie médicale, les communications sans fil, la compatibilité/interférence électromagnétique (eletromagnetic compatibility/electromagnetic interference EMC/EMI), etc. ont conduit à des systèmes de plus en plus complexes. Dans de nombreux cas, un seul dispositif est devenu une structure très complexe qui comprend un certain nombre de conducteurs, de diélectriques et de semiconducteurs de formes arbitraires et de nature physique complexe. Les technologies de fabrication onéreuses excluent la possibilité de modifier un dispositif si ses performances ne sont pas conformes aux spécifications de conception. Par conséquent, il est nécessaire de développer des méthodes capables de caractériser de manière extrêmement précise ces systèmes pour analyser les structures qui nous intéressent. Cependant, il est difficile d'obtenir de telles méthodes en raison de la nature complexe de n'importe quel des dispositifs électromagnétiques modernes. En effet, les solutions analytiques exactes ne sont connues que pour un nombre limité de cas particuliers, qui ne sont pratiquement jamais directement applicables à des applications réelles [1–3]. L'impossibilité de dériver des solutions analytiques exactes à partir des équations de Maxwell pour des applications réelles a conduit à une recherche intensive sur les techniques numériques. L'application de méthodes numériques aux problèmes de champs électromagnétiques est connue sous le nom de calcul en électromagnétisme (computational electromagnetics CEM) [4, 5].

La simulation numérique constituent aujourd'hui l'un des piliers de la recherche scienti-

fique et de certains domaine de l'ingénierie et devenue incontournable pour la modélisation des systèmes très variés allant de la physique, la chimie, la biologie, mais également pour la modélisation des sciences humaines et les sciences de l'ingénieur. Dans tous ces domaines, l'amélioration des modèles et la nécessité d'intégrer un nombre croissant de données, ou d'utiliser un modèle couplant de nombreux phénomènes, ou encore de résoudre numériquement des équations particulièrement complexes, requièrent une grande puissance de calcul et des calculateurs de plus en plus performants. Pour répondre à cette demande grandissante de performance, les centres de calcul se sont tournés vers l'utilisation du calcul parallèle ou calcul haute performance (High Performance Computing HPC) et se sont équipés de machines toujours plus puissantes, suscitant une véritable course à la performance entre les constructeurs .

Le calcul haute performance s'impose ainsi comme un outil essentiel de la recherche scientifique, technologique et industrielle. Avec les progrès fulgurants des calculateurs, sa pratique s'est généralisée à toutes les disciplines, fondamentales et appliquées, au point de devenir en quelque sorte le troisième pilier de la science, au côté de la théorie et de l'expérimentation.

Depuis l'ENIAC (Electronic Numerical Integrator Analyser and Computer) conçu pendant la deuxième guerre mondiale et passant pour être le premier ordinateur du monde complètement électronique jusqu'aux années 80, les supercalculateurs étaient essentiellement destinés à des applications relevant des secteurs de la défense, de la fusion nucléaire et de l'énergie. Cependant, depuis, le domaine du calcul haute performance a subi plusieurs révolutions matérielles. Les ordinateurs ont évolué vers des architectures multiprocesseurs et des super-calculateurs dédiés qui ont atteint leur âge d'or dans les années 80. Grâce au développement de nouvelles technologies réseau dans les années 90, les super-calculateurs très onéreux ont cédé leur monopole aux grappes de calcul (cluster) , composées de plusieurs machines "standard" (nœuds) collaborant et communiquant par le biais de réseaux rapides, et dont le rapport performance/prix s'avère plus abordable. La popularité des grappes a cependant introduit un bouleversement des techniques de programmation et la mise en place de schémas de programmation spécifiques aux architectures distribuées.

Parmi une variété de méthodes de simulation numérique en CEM qui fournissent une solution complète aux équations de Maxwell, beaucoup sont basés sur la méthode des moments (Method of Moment MoM) [6], la méthode des différences finies dans le domaine temporel (Finite Difference Time Domain FDTD) [7] et la méthode des éléments finis (Finite Element Method FEM) [8]. D'autres méthodes, telles que la méthode des lignes de transmission et la technique d'intégration finie, peuvent être identifiées comme étant soit

une variante, soit l'équivalent de l'un des trois premiers.

Parmi les trois techniques numériques majeures, la méthode des moments est considéré comme l'une des méthodes numériques basée sur l'équation intégrale les plus utilisées. La méthode est basée sur la formulation d'équations intégrales en termes de fonctions de Green comme solution fondamentale aux équations de Maxwell. De plus, en raison de l'utilisation des fonctions de Green, la méthode du moment produit une matrice dense dont le calcul et la solution sont associés à une complexité de calcul très élevées. Par conséquent, la méthode du moment traditionnel est considérée très consommatrice des ressources en temps et en mémoire pour l'analyse des grandes structures antennaires, en particulier les réseaux d'antennes. Outre la méthode des moments, un certain nombre de variantes de cette dernière ont été développées. Par exemple, la méthode multipolaire rapide (Fast Multipole Method FMM) et l'algorithme multipôle multi-niveaux (MultiLevel Fast Multipole Algorithm, MLFMA) ont été proposés pour accélérer le produit matrice-vecteur qui apparaît dans la solution itérative des équations MoM [3, 9].

La méthode FDTD, inventée au milieu des années 1960, résout les équations de Maxwell discrétisées sur une grille rectangulaire directement dans le domaine temporel. La méthode peut facilement gérer les matériaux anisotropes et non homogènes et est devenue très puissante et de plus en plus populaire en raison de sa simplicité dans la formulation, l'implémentation et la discrétisation [10].

La méthode FEM a son origine dans le domaine de l'analyse structurelle. Bien que le traitement mathématique antérieur de la méthode ait été fourni par Courant [8] en 1943, la méthode n'a été appliquée aux problèmes électromagnétiques (EM) qu'en 1968. C'est une méthode utilisée pour résoudre les problèmes électromagnétiques aux limites dans le domaine fréquentiel en utilisant une forme variationnelle. Elle peut être utilisé avec des éléments canoniques bidimensionnels et tridimensionnels de formes différentes, permettant une discrétisation très précise du domaine de la solution. Bien que la méthode des différences finies et la méthode des moments sont conceptuellement plus simples et plus faciles à programmer que la FEM, celle-ci reste une technique numérique plus puissante et plus polyvalente pour traiter des problèmes impliquant des géométries complexes et des milieux non homogènes. Comme dans la méthode FDTD, le domaine de la solution doit être tronqué, ce qui rend le FEM inadapté aux problèmes de rayonnement ou de diffusion, à moins d'être combinée avec une approche avec une solution aux limites [11–13].

La motivation de cette thèse n'est pas à proprement parler de développer de nouvelles méthodes de calcul, mais de prouver l'utilité d'un cluster de calcul conçu à partir de composants disponible dans n'importe quelle salle d'ordinateurs combiné à un outil très ap-

précié par la communauté scientifique qui est MATLAB pour la résolution des problèmes de calculs intensifs en électromagnétiques. Cette solution passe par la parallélisation des programmes de calcul des méthodes numériques rencontrées en électromagnétiques en exploitant au mieux les architectures des calculateurs parallèles à mémoire distribuée et, bien évidemment, pour répondre au mieux au problème posé. Les méthodes proposées devront également répondre à un double impératif d'extensibilité (« scalability » en anglais) numérique et parallèle. En d'autres termes, celles-ci doivent conserver les mêmes propriétés (convergence,...), quelle que soit la taille du problème traité, tout en assurant une diminution du temps de calcul en fonction du nombre de processeurs mis en jeu.

La première étape de ce travail consiste à maîtriser le processus de construction d'un cluster de calcul haute performance de type Beowulf du point de vue matériel et logiciel. En effet, et par manque de solution clé à la main, nous avons construit un cluster composé d'un nœud principal ou master node en anglais et huit (08) nœuds de calculs interconnectés par un réseau gigabit. Le deuxième point est la mise en marche du cluster par l'installation de tous les outils nécessaire à l'exploitation de ce cluster. Nous avons utilisé une solution open source Rocks [14], une distribution Linux basée sur Red Hat avec des paquets supplémentaires et une configuration programmée pour automatiser le déploiement de cluster Linux hautes performances. Le système de base étant installé, nous avons installer et configurer MATLAB pour une utilisation sur un système de calcul à mémoire distribué.

Dans une deuxième étape, nous avons développer trois applications de calculs électromagnétiques en utilisant le calcul parallèle sur le cluster conçu dans la première étape. Nous avons ainsi implémenter la méthode des moment spatiale pour le calcul des caractéristiques d'une antenne microstrip circulaire multi-couches alimentée par un cable coaxiale. Le code développé est un code nécessitant un calcul intensif ce qui nous amener à le paralléliser en utilisant le modèle de programmation (Single Program Multiple Data, SPMD). Le gain atteint exprimé en accélération définie comme le ratio du temps d'exécution séquentiel et le temps d'exécution parallèle peut atteindre une valeur de 23 (pour une valeur théorique maximale de 32). Dans la deuxième application, le calcul symbolique des fonctions de base d'ordre supérieur de la FEM peut devenir excessif. Ainsi, nous avons pu implémenter une version parallèle qui présente une excellente extensibilité et un gain satisfaisant. Outre, le problème du temps d'exécution, il y a des problèmes où l'espace mémoire nécessaire aux calculs est énorme à tel point qu'il est impossible de tenir sur une seule machine. La parallélisation dans ce cas offre la possibilité de distribuer les données et les calculs de façon à gagner dans les deux directions. C'est cet objectif que nous avons cherché à satisfaire dans le développement de la troisième application qui consiste à analyser un réseau d'antenne

bowtie.

Le présent manuscrit est organisé comme suit :

Le premier chapitre introduit d'abord le calcul haute performance. Les architectures parallèles y sont présenté ainsi que le modèle de classification et les modèles de programmation de ces architectures. Il décrit aussi les principaux critères de mesure des performance utilisés pour ces systèmes à savoir l'accélération et l'efficacité.

Le deuxième chapitre est consacré à une présentation des techniques et méthodes de calcul en électromagnétique. Il présente les trois méthodes les plus utilisées dans le CEM, à savoir la méthode des moments, la méthode des différences finis dans le domaine temporel et la méthode des éléments finis. Une biographie assez complète des travaux de recherches sur le calcul parallèle dans le domaine de la CEM durant les dernières décennies conclue ce chapitre.

Les chapitres 3, 4 et 5 détaillent les contributions de cette thèse. Le chapitre 3 présente le processus de modélisation d'une antenne microstrip circulaire et multi-couches. Dans un premier temps, un développement mathématique est réalisé et des résultats de simulations sont données. Ensuite, le processus de parallélisation est détaillé et les mesures de performances du code ainsi parallélisé sont analysées. Dans le chapitre 4, nous présentons une méthode systématique pour la génération et le calcul des fonctions de formes d'ordre supérieur de la FEM sous format symbolique. La parallélisation du code est une solution que nous proposons pour pouvoir calculer des ces fonctions pour des ordres supérieur à trois dans un temps moindre. Dans le chapitre 5, nous discutons du processus de parallélisation des codes pour la caractérisation d'un réseau d'antennes bowtie 2D. La méthode des moments spatiale basée sur les éléments RWG (Rao Wilton Glisson) est utilisée. Le problème nécessite une quantité de mémoire de travail grande de sorte qu'une solution sur une seule machine est impossible pour une structure complexe et un nombre d'éléments grand. La parallélisation permet de résoudre le problème de mémoire ainsi qu'une diminution du temps de calcul. Les résultats sont présentées et discutées pour démontrer la pérennité de notre approche.

Enfin, nous terminons par une conclusion générale et des perspectives pour de future travaux.

Chapitre 1

Le calcul haute performance

Sommaire

1.1 Introduction au calcul parallèle 6
1.2 Principes du parallélisme 9
1.3 Architectures des calculateurs parallèles 9
1.4 Modèles de programmation pour les architectures parallèles 14
1.5 Mesure de performances des architectures multiprocesseurs 20
1.6 MATLAB et calcul parallèle 23
1.7 Conclusion 24

1.1 Introduction au calcul parallèle

De nombreuses applications numériques scientifiques nécessitent des ressources informatiques de plus en plus performantes. Cette croissance des besoins a pour origine la volonté des scientifiques de résoudre des problème de plus en plus complexes, ou bien de résoudre le même problème de plus en plus rapidement. Dans tous les domaines, l'amélioration des modèles et la nécessité d'intégrer un nombre croissant de données, ou d'utiliser un modèle couplant de nombreux phénomènes, ou encore de résoudre numériquement des équations particulièrement complexes, requièrent une grande puissance de calcul et des calculateurs de plus en plus performants.

En permettant la résolution des équations les plus complexes ou l'étude des modèles les plus sophistiqués, le calcul haute performance HPC (High Performance Computing) couvre des applications de plus en plus importantes. Le calcul haute performance s'impose ainsi

comme un outil essentiel de la recherche scientifique, technologique et industrielle. Avec les progrès fulgurants des calculateurs, sa pratique s'est généralisée à toutes les disciplines, fondamentales et appliquées, au point de devenir en quelque sorte le troisième pilier de la science, au côté de la théorie et de l'expérimentation.

La parallélisation peut se faire sur une machine possédant plusieurs cœurs ou processeurs, ou bien sur plusieurs machines en réseau, ou bien encore grâce à une combinaison de ces deux méthodes ou même sur des processeurs graphique. Le problème doit pouvoir être découpé en parties indépendantes et pouvant être exécutée simultanément. De fait, tous les algorithmes ne peuvent être parallélisés.

Les premières plate-formes possédant une architecture parallèle sont les machines désignées fréquemment par le terme de supercalculateurs. Ils ont fait leur entrée dans les centres de calcul des années 70, et connus leur apogée dans les années 80. Ces architectures, dont la conception est propre aux divers constructeurs, sont caractérisées par un parallélisme interne et sont aujourd'hui dotées de plusieurs milliers ou millions d'unités de calcul.

Cependant, les super-calculateurs présentent un inconvénient majeur puisqu'ils nécessitent des coûts de développement prohibitifs et l'implantation et l'entretien d'une machine de ce type posent des problèmes logistiques concrets qui induisent un coût non négligeable en plus de la machine proprement dite.

En marge des progrès réalisés dans les supercalculateurs, les micros-ordinateurs ont eux aussi connu de profondes mutations, se sont standardisés, et ont peu à peu conquis les entreprises, les laboratoires, les universités et les foyers. L'ensemble de ces stations de travail s'est avéré être une alternative intéressante pour les centres ne disposant pas de moyens financiers suffisants. En effet, en réunissant la puissance de calcul de chacune des ces machines grâce à un réseau d'interconnexion, il est possible de réaliser l'équivalent d'un super calculateur communément appelé grappe de machines ou cluster.

Une grappe de machines désigne un ensemble de d'ordinateurs, appelés nœuds, tous interconnectés, dans le but de partager des ressources informatiques. Une grappe peut être constituée d'ordinateurs de bureaux, de « racks » de machines constituées de composants standards ou de « lames » également constituées de composants standards afin d'optimiser l'espace physique. Une grappe est généralement composée de machines homogènes en termes d'architecture et de système d'exploitation. Elle ne regroupe que des machines appartenant au même domaine d'administration réseau et les nœuds communiquent entre eux en utilisant un réseau de communication rapide.

Les grappes de calcul, avec un rapport très agressif coût/performance, se sont rapidement

positionnées comme des concurrentes directes des architectures parallèles. Contrairement aux super-calculateurs spécialisés, les grappes se basent sur un parallélisme externe : plutôt que de multiplier les unités de calcul à l'intérieur de la machine, elles font collaborer un ensemble d'ordinateurs indépendants (nœuds) qui communiquent au travers d'un réseau dédié. L'utilisation de grappes de calcul dans le calcul scientifique a émergé dans le milieu des années 90 avec les projets NOW [15] et Beowulf [16]. Ce concept a ensuite pris de l'ampleur grâce à l'arrivée des réseaux rapides, avec une amélioration substantielle du débit (plusieurs ordres de grandeur) et une forte réduction de la latence. En effet, l'utilisation de réseaux « classiques » implique un coût d'interconnexion entre les processeurs important et constituait un inconvénient considérable par rapport aux architectures massivement parallèles. Les réseaux rapides ont ainsi réduit les coûts de communication sur grappes, permettant l'utilisation des systèmes distribués pour le calcul intensif.

Cependant, en dépit de ces avantages, les grappes présentent des aspects plus problématiques dont le premier d'entre eux est la difficulté intrinsèque de programmation. En effet, une exploitation efficace du matériel disponible est souvent un défit car les nœuds peuvent être multi-processeurs, voire intégrer du multithreading. De plus, les outils logiciels disponibles se veulent souvent génériques et portables (à cause de la diversité du matériel), et le niveau de performances est inférieur à celui d'un supercalculateur qui peut se permettre d'offrir des outils très spécialisés et non portables.

En parallèle des CPU, les GPU (Graphics Processing Unit) circuits initialement conçus pour l'affichage graphique (notamment la 2D et dans une plus large mesure la 3D), ont connu un essor fulgurant. La rivalité des constructeurs NVIDIA et ATI (racheté par AMD) a été bénéfique et elle a mené à l'évolution des GPU vers des circuits dédiés au calcul hautement parallèle. Avec plusieurs centaines de processeurs scalaires et une bande passante mémoire importante, le GPU dispose d'une puissance de calcul théorique dépassant largement celle des processeurs. L'exploitation de ces cartes graphiques dans le calcul général scientifique connue sous la terminologie General-Purpose computations on GPU (GPGPU) était au début assez fastidieuse à cause des contraintes de programmation adaptées a un environnement de développement purement graphique. Cependant, avec l'apparition d'API (Application Programming Interface) dédiées comme celle d'AMD-ATI appelée CTM pour Close-To-Metal devenant par la suite AMD STREAM technology [17] et ou bien celle de NVIDIA appelée CUDA pour Compute Unified Device Architecture [18], le GPGPU est devenu de plus en plus facile à exploiter. Ces nouvelles API ont ouvert un accès direct de bas niveau aux nombreux processeurs de calcul parallèles des GPU ainsi qu'à leur mémoire.

1.2 Principes du parallélisme

Le parallélisme est défini comme l'ensemble des techniques logicielles et matérielles permettant l'exécution simultanée de séquences d'instructions indépendantes sur des processeurs et/ou cœurs différents ou comme l'ensemble d'éléments de traitement qui coopèrent et communiquent pour résoudre un problème plus rapidement [19]. Les techniques matérielles représentent les différentes architectures de calculateur parallèle et les techniques logicielles, les différents modèles de programmation parallèle.

1.3 Architectures des calculateurs parallèles

1.3.1 Taxonomie de Flynn

La classification des architectures des ordinateurs la plus populaire a été définie par Flynn en 1966 [20, 21]. Le système de classification de Flynn est basé sur la notion de flux d'informations. Deux types d'informations circulent dans un processeur : les instructions et les données. Le flux d'instructions est défini comme la séquence d'instructions exécutée par l'unité de traitement et le flux de données est défini comme le trafic de données échangé entre la mémoire et l'unité de traitement. Selon que l'un ou l'autre des flux d'instructions ou de données est unique ou multiple, l'architecture informatique dans la taxonomie de Flynn, peut être classée dans les quatre catégories distinctes suivantes 1.1 [22] :

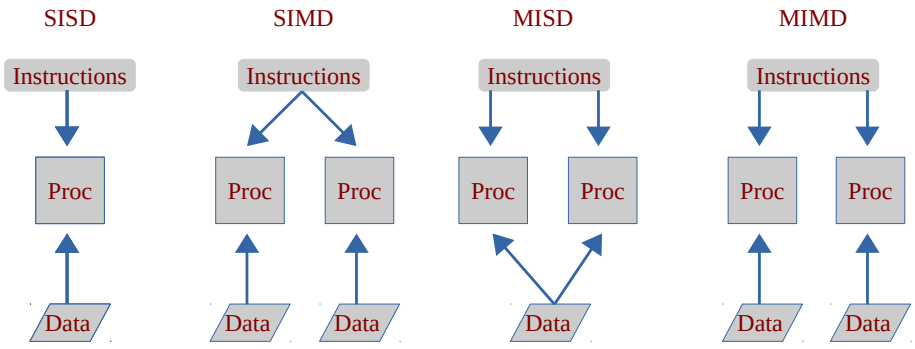


FIGURE 1.1 – Les quatres classes d'architectures de la taxonomie de Flynn

1.3.1.1 Single Instruction, Single Data (SISD)

Cette architecture correspond au cas où il n'y a qu'une seule unité d'exécution qui a un accès à une mémoire de programme et de données. Dans chaque étape, l'unité d'exécution charge une instruction et les données correspondantes et exécute l'instruction. Le résultat est stocké dans la mémoire de données.les résultats. Cette catégorie correspond à l'architecture séquentielle dite de Von Neumann.

1.3.1.2 Multiple Instruction, Single Data (MISD)

Dans cette architecture, Il y a plusieurs (multiple) unité d'exécution possédant chacune une mémoire de programme privé, avec un accès partagé à un seul mémoire de données globale. Ceci correspond à l'application de traitements différents exécutés en parallèle sur les mes données. Ce modèle d'exécution est très restrictif et il n'existe pas un ordinateur parallèle commerciale basé sur ce type d'architecture.

1.3.1.3 Single Instruction, Multiple Data (SIMD)

Cette architecture désigne le cas où il y a des unité d'exécution multiples ayant chacune un accès privé à une mémoire de données (partagée ou distribuée). Cependant, il n'y a qu'une seule mémoire programme à partir de laquelle un processeur de contrôle spécial récupère et distribue les instructions. Ainsi, Le même traitement est appliquée en parallèle de façon synchrone par tous les unités d'exécution sur différents éléments de données. Pour les applications avec un degré important de parallélisme de données, l'approche SIMD peut être très efficace.

1.3.1.4 Multiple Instruction, Multiple Data (MIMD)

Cette catégorie regroupe les architectures ayant plusieurs unités d'exécution possédant chacune son propre flot de données. Les unités d'exécution fonctionnent de manière asynchrone les uns avec les autres. Cette architecture regroupe la plupart des systèmes parallèles actuels dont les processeurs multicœurs ou les systèmes de grappes de calcul.

1.3.2 Architecture Matériels des calculateurs parallèles

En 30 ans, les machines dédiées au calcul intensif ont été marquées par de véritables révolutions technologiques. Ces transformations sont intervenues à de multiples niveaux :

amélioration et multiplication des processeurs ; développement de réseaux efficaces ; distribution de la mémoire et production de nouvelles technologies d'interconnexion ; ou encore miniaturisation engendrant l'augmentation du nombre de composants. Dans cette section, nous allons présenter les principales architectures qui ont marqué l'évolution du paysage du calcul intensif.

1.3.2.1 Les supercalculateurs

La chronologie de l'évolution du matériel de calcul hautes-performances, commence par la domination des super-calculateurs depuis le milieu des années 60 jusqu'aux années 90. Durant cette période, l'architecture de ces machines a beaucoup évoluée. Il s'agit là d'un matériel très conséquent, aussi bien au niveau de la taille que des moyens nécessaires à leur exploitation. Les super-calculateurs nécessitent des coûts de développement prohibitifs et l'implantation et l'entretien d'une machine de ce type posent des problèmes logistiques qui induisent un coût non négligeable.

les super-calculateurs ont fait leur entrée dans les centres de calcul des années 70, et connus leur apogée dans les années 80. Ces architectures, dont la conception est propre aux divers constructeurs, sont caractérisées par un parallélisme interne ; qui consiste en la présence de plusieurs processeurs communiquant à travers un bus interne et sont aujourd'hui dotées de plusieurs milliers d'unités de calcul.

Le père de la première machine qui reçu le nom de "supercalculateur" est Seymour Cray avec le **CDC 6600** [23] (figure 1.2) sortie en 1964. Sa puissance était de 4.58 MFLOPS (Mega Operation Flottantes Par Secondes), soit plus de 10 fois la puissance de l'IBM 7030 Stretch (sortie en 1961) et plus de 140 fois la puissance de l'IBM 7044 sorti en 1963. Sa puissance colossale justifie son nom de supercalculateur car pour la première fois la puissance d'un ordinateur est nettement supérieure à celle des anciennes machines mais aussi celle des machines récentes. Pour la puissance de calcul, le processeur central dispose de 10 unités de calcul arithmétique indépendantes (1 unité d'addition flottante, 2 de multiplication flottante, 1 pour la division flottante, 1 addition entière, 2 incrémenteurs, 1 pour le décalage binaire, 1 pour les branchements conditionnels et 1 qui se charge du calcul booléen) ainsi que d'une mémoire principale. Puis, au cours des années 70, les constructeurs ont intégré des processeurs vectoriels capables d'appliquer une même instruction à un ensemble de données, favorisant ainsi les calculs s'appliquant à des tableaux de nombres, comme la manipulation d'images. Afin d'accroître les performances des supercalculateurs, plusieurs de ces processeurs vectoriels sont associés pour fonctionner en parallèle et constituent ainsi

FIGURE 1.2 – Le supercalculateur CDC 6600

la génération suivante.

À l'approche des années 90, ces unités de calcul vectorielles ont laissé la place à des processeurs scalaires plus « simples », donc moins coûteux, mais plus nombreux. Ce type d'architectures est souvent désigné sous le terme de système massivement parallèle ou Massively Parallel Processing (MPP) en anglais, le nombre de processeurs pouvant atteindre plusieurs centaines ou même plusieurs centaines de milliers après.

Bien que moins présents de nos jours, ces systèmes tendent à réapparaître : ils représentent 13.2% des machines de calcul les plus puissantes, répertoriées par le Top500 [1] (Liste de Juin 2015) (voir figure 1.3). Citons ainsi le supercalculateur **Tianhe-2** (figure 1.4) du Centre National de Calcul de Tianjin (Chine), construite par le NUDT (National University of Defense Technology) qui occupe actuellement la première place du Top500 (il occupe cette place depuis 2012). Le Tianhe-2 dispose de 48 000 cartes Xeon Phi et 32 000 sockets pour Xeon Ivy Bridge pour un total de 3 120 000 cœurs. Sa puissance atteignant les 33,86 PetaFLOPS ($33.86 \cdot 10^{15}$ opérations en virgule flottante par secondes). Les supercalculateurs sont presque toujours conçus spécifiquement pour un certain type de tâches (le plus souvent des calculs numériques scientifiques : calcul matriciel ou vectoriel) et ne cherchent pas de performance particulière dans les autres domaines. En plus, leur architecture mémorielle est étudiée pour fournir en continu les données à chaque processeur afin d'exploiter au maximum sa puissance de calcul.

1. http://www.top500.org/

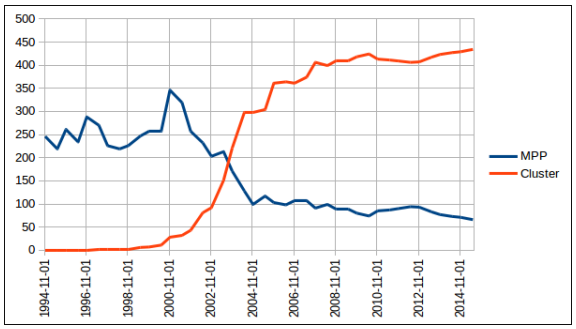


FIGURE 1.3 – Evolution des architectures du Top500

FIGURE 1.4 – Le supercalculateur chinois Tianhe–2

1.3.2.2 Les grappes de calcul - les cluster

Contrairement aux supercalculateurs spécialisés, les grappes se basent sur un parallélisme externe : plutôt que de multiplier les unités de calcul à l'intérieur de la machine, elles font collaborer un ensemble d'ordinateurs indépendants (nœuds) qui communiquent au travers d'un réseau dédié (figure 1.5). La mémoire n'est donc plus ici partagée entre les différents processeurs mais privée à chaque nœud. Nous sortons donc du cadre du calcul à proprement dit parallèle, pour entrer dans celui du calcul distribué. C'est une solution véritable-

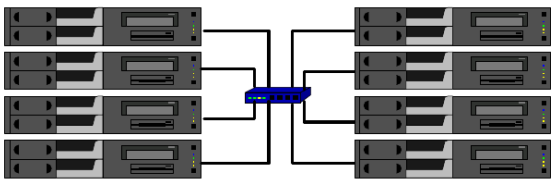


FIGURE 1.5 – Une grappe d'ordinateurs - cluster

ment avantageuse en comparaison des supercalculateurs en raison du rapport performances/prix et des capacités d'extension avec des composants standards. L'ajout de nouveaux nœuds de calculs permet d'augmenter la puissance de calcul globale. La principale difficulté est de réussir à exploiter efficacement un tel système, là où les supercalculateurs disposent d'outils spécifiquement optimisés pour l'architecture. Malgré cela, les grappes se sont rapidement imposées dans le monde du calcul parallèle : au dernier recensement (c'est-à-dire en Juin 2015), elles constituent plus de 85% du parc des machines les plus puissantes du monde (figure 1.3).

1.4 Modèles de programmation pour les architectures parallèles

Un modèle de programmation parallèle spécifie la vue du programmeur sur un ordinateur parallèle en définissant comment le programmeur peut coder un algorithme. Cette vue est influencée par la conception architecturale et le langage, le compilateur ou les bibliothèques d'exécution et, par conséquent, il existe de nombreux modèles de programmation parallèle différents pour une même architecture. Il existe plusieurs critères selon lesquels les modèles de programmation parallèles peuvent différer [24] :

— le niveau de parallélisme qui est exploité dans l'exécution parallèle (niveau de l'instruction, niveau procédural ou boucles parallèles) ;

— la spécification explicite ou implicite du parallélisme définie par l'utilisateur ;

— la façon dont les parties de programme parallèles sont spécifiées ;

— le mode d'exécution des unités parallèles (SIMD ou SPMD, synchrone ou asynchrone) ;

— les modes et les schémas de communication entre les unités de calcul pour l'échange d'informations (communication explicite ou variables partagées) ;

— les mécanismes de synchronisation pour organiser les calculs et les communications entre les unités de calcul parallèles.

Cette section répertorie les principaux modèles de programmation parallèle existants et présente des travaux menés sur ces modèles pour résoudre les problématiques causées par les architectures hiérarchiques.

1.4.1 Modèle en mémoire partagée

Les modèles de programmation par mémoire partagée se basent sur un ensemble de fils d'exécution qui évoluent de façon concurrente dans un même espace d'adressage. Ce système propose des échanges implicites et efficaces entre les tâches en utilisant la mémoire de la machine comme zone de transfert dans laquelle les données sont déposées ou lues. Le coût de communication s'apparente ainsi à ceux des temps d'accès mémoire, néanmoins, pour assurer la consistance et la cohérence des données, l'emploi de méthodes de synchronisation et de verrouillage (synchronisation explicite, utilisation d'instructions atomiques, exclusion mutuelle, attente sur condition, ect.) est indispensable. Ce type de modèle est clairement adapté à des machines de type multiprocesseur symétrique (Symmetric MultiProcessing,SMP), où plusieurs processeurs vont partager une mémoire physique commune.

1.4.1.1 Le multithreading

Une première méthode pour créer des applications parallèles en mémoire partagée consiste à exprimer le parallélisme à la main, à l'aide de bibliothèques de processus légers (threads) qui fournissent des primitives de création, gestion et destruction de threads. Afin d'unifier les différentes interfaces des implémentations des bibliothèques de gestion des threads, le standard IEEE POSIX 1003.1c-1995 [2] fut proposé. Pour les implémentations qui respectent le standard on parle alors de POSIX threads ou Pthreads [25].

1.4.1.2 Open Multi-Processing (OpenMP)

En octobre 1997, un consortium d'industriels et de constructeurs définit le standard OpenMP [26, 27] rassemblant ainsi certains efforts réalisés par le passé dans le domaine de la parallélisation basée sur des directives de compilation (*pragmas*). Celles-ci sont interprétées par le compilateur pour générer automatiquement le parallélisme et la gestion des threads sous-jacente (création, destruction, synchronisation, etc.). Les spécifications d' OpenMP sont gérées par le consortium OpenMP Architecture Review Board qui est composé de grands noms parmi les sociétés de l'industrie logicielle et matérielle. Un compilateur compatible OpenMP , grâce aux informations fournies par les directives de compilation, produit des binaires multi-thread qui reposent le plus souvent sur la bibliothèque de threads POSIX fournie avec le système d'exploitation.

2. https://standards.ieee.org/findstds/interps/1003-1c-95_int/index.html

L'intérêt premier de ce modèle tient dans le peu d'efforts à fournir par le programmeur pour rapidement paralléliser un code écrit en C, C++ ou Fortran. L'ajout d'une directive OpenMP au niveau d'une boucle permet ainsi de répartir les indices de boucles entre les threads gérés par le support exécutif OpenMP. Plusieurs mots-clés placés avec les directives déterminent le type de répartition, la visibilité des variables (partagées ou privées) ou encore le nombre de threads d'une section parallèle.

1.4.1.3 TBB ou Thread Building Blocks

La bibliothèque TBB [28] d'INTEL fournit une abstraction de haut niveau pour décrire le parallélisme d'un code C++ destiné à être exécuté sur des architectures multicœurs. TBB n'est pas seulement une bibliothèque de remplacement de threads, mais elle représente un parallélisme basé sur les tâches proche du programmeur, qui fait abstraction des détails de la plateforme et les mécanismes de gestion des thread pour accroître les performances et l'évolutivité [28]. Le programmeur manipule des objets, des itérateurs et des conteneurs C++ parallèles pour utiliser TBB : ce sont des outils usuels pour un programmeur C++. La bibliothèque offre un environnement d'exécution capable de récupérer des informations sur la topologie de l'architecture sous-jacente. Il crée ainsi un nombre de threads égal au nombre de processeurs virtuels de la machine et s'adapte de façon transparente aux différentes architectures multicœurs. TBB propose de décrire les applications sous forme de tâches et de les appeler via une combinaison de templates parallèles (*parallel_for, parallel_reduce, parallel_scan, parallel_sort, parallel_while*).

1.4.1.4 Cilk

Cilk [29, 30] est un langage multithread pour la programmation parallèle qui généralise la sémantique de C en introduisant des constructions spécifiques par l'ajout de mots clés pour le contrôle parallèle. L'objectif derrière le développement de Cilk a été de faire du langage Cilk une véritable extension parallèle de C, à la fois en terme sémantique qu'en termes de performances. Sur un ordinateur parallèle, les constructions de contrôle Cilk permettent au programme de s'exécuter en parallèle. Si les mots-clés des constructions parallèles sont supprimés d'un programme Cilk, on obtient alors un programme C syntaxiquement et sémantiquement correct.

Au démarrage du programme, le support exécutif crée quelques processus légers qui auront la charge d'exécuter les tâches Cilk définies par le programmeur. Chaque processus léger possède sa propre file de travaux qui contient les tâches Cilk.

En 2006, la société Cilk Arts a acquis la licence de Cilk dans le but de développer une implémentation commerciale avec la prise en charge du C++. Cilk++ v1.0 est sorti en décembre 2008 avec une prise en charge complète du C++, y compris les exceptions et l'ajout d'une construction pour les boucles parallèles. Un nouveau type d'objet a été ajouté pour la gestion de l'accès concurrent aux variables globales sans utiliser les verrous.

En 2009, Intel Corporation a racheté Cilk Arts. La technologie Cilk a été fusionnée avec Array Notation pour fournir une extension complète du langage afin d'implémenter à la fois le parallélisme des tâches et celui des vecteurs. Intel Cilk Plus a été publié par Intel en 2010 et a été inclus dans la suite Intel C++ Composer XE. Les principales caractéristiques comprennent :

— Support du C et C++ ;

— Compatible avec les débogueurs standard ;

— Utilisation des conventions d'appel standard - Les pages mémoires des fonctions Cilk sont alloués sur la pile afin que les fonctions C/C++ puissent appeler les fonctions Cilk librement.

En plus, Intel a rendu les spécifications de Intel Cilk Plus disponibles gratuitement sur le Web. En 2012, Intel a proposé Intel Cilk Plus comme standard C++.

1.4.2 Modèle en mémoire distribuée

La mémoire distribuée est caractéristique des architectures de grappes, composées de machines interconnectées (cluster) qui n'ont pas de mémoire commune. Dans ce modèle, chaque processus accède seul à sa mémoire privée. Contrairement au modèle à mémoire partagée, le transfert d'information entre les processus est donc fait par communication explicite. De par la structure de la machine, chaque processeur dispose de ses données propres, le partage d'une donnée locale doit donc se faire en la communiquant aux autres nœuds via le réseau d'interconnexion. Cette communication est basée sur des émissions et des réceptions de messages.

Le modèle le plus adapté à ce type de machine est le modèle de programmation par échange de messages. Ce modèle assure la portabilité de l'application sur toutes les plateformes grâce à l'emploi des librairies d'échanges de messages standard (PVM, MPI).

1.4.2.1 La bibliothèque de communication Parallel Virtual Machine - PVM

La bibliothèque de communication Parallel Virtual Machine [31, 32] (PVM) est issu d'un projet interne de l'Oak Ridge National Laboratory (ORNL) en 1989. Elle permet de rassembler un ensemble des machines hétérogènes d'un réseau en une unique machine virtuelle à mémoire distribuée reposant sur le passage de messages pour les communications. PVM a été conçu pour combiner des machines qui diffèrent au niveau des systèmes d'exploitation, des représentations de données (nombre de bits et ordre des octets), des architectures matérielles (multiprocesseur, monoprocesseur et même supercalculateurs), des langages et des réseaux et les faire travailler ensemble sur un seul problème de calcul [16].

L'idée de base derrière PVM était de créer une collection de programmes qui pourrait être chargé sur n'importe quel ensemble d'ordinateurs de telle manière à ce que cet ensemble apparaît comme un seul ordinateur parallèle à mémoire distribuée. PVM fournit un moyen d'agréger la puissance et la mémoire des ressources de calcul distribuées. La fiabilité et la facilité d'utilisation de PVM ont rendu cet outil très populaire pour avoir un ordinateur parallèle virtuel qui fournit plusieurs fois plus de puissance qu'ils ne l'auraient fait autrement. PVM était si populaire qu'il est devenu un standard de facto pour l'informatique distribuée hétérogène dans le monde entier.

Le système PVM est composé de deux parties. La première partie est un démon, appelé *pvmd*, qui doit être installé sur tous les ordinateurs constituant la machine virtuelle. La deuxième partie du système est une bibliothèque de fonctions d'interface PVM. Elle contient une collection complète de primitives fonctionnelles nécessaires à la collaboration entre les tâches d'une application. Cette bibliothèque contient des fonctions qui peuvent être appelées par l'utilisateur pour la détection de pannes, la transmission de messages, la création des processus, la coordination des tâches et la modification de la machine virtuelle [16].

L'environnement d'exécution de la machine virtuelle parallèle est basé sur les concepts suivant :

Pool de machines configuré par l'utilisateur c'est l'ensemble de machines configuré par l'utilisateur pour l'exécution des taches des applications PVM. Le pool peut être modifié à n'importe quel moment par l'ajout ou la suppression de machines qui est une caractéristique importante pour la tolérance aux pannes.

Accès transparent au matériel Les programmes d'application peuvent voir l'environnement matériel comme une ressource transparente ou au contraire, ils peuvent exploiter les caractéristiques spécifiques des machines dans le pool en positionnant certaines tâches sur les ordinateurs les plus appropriés. Par exemple, Sur les clusters

de grande taille, les nœuds d'E/S peuvent exécuter les tâches de surveillance et les nœuds de calcul avoir la plus grande partie de la charge de calcul.

Passage de message explicite PVM offre des opérations de communication de base bloquante, non bloquante et collective. Pour accroitre les performances, PVM utilise les fonctions natives de transmission de messages sur les multiprocesseurs pour tirer parti du matériel sous-jacent.

Modèle de programmation dynamique PVM supporte un modèle de programmation dynamique où les noeuds du pool et les tâches peuvent être ajoutés et supprimés à n'importe quel moment.

Groupes dynamiques Dans quelques applications, il est logique de raisonner en terme de groupes de tâches. PVM inclut le concept de groupes nommés par l'utilisateur. Lorsqu'une tâche rejoint un groupe, elle reçoit un numéro d'"instance" unique dans ce groupe. Les numéros d'instance commencent à 0.

1.4.2.2 La bibliothèque de passage de messages - MPI

Le standard Message Passing Interface [33], annoncé en 1994 par le MPI Forum, est l'aboutissement des travaux d'un consortium de constructeurs, d'industriels et d'universitaires, réunis pour proposer une solution standard dédiée au calcul parallèle sur grappes et architectures parallèles. Cette interface de passage de messages définit un ensemble de fonctionnalités et de spécifications génériques, entièrement indépendantes des architectures. Elle s'articule autour de primitives de communications point-à-point et d'opérations collectives.

MPI propose une interface de programmation définissant un ensemble de fonctions de communication ainsi que des recommandations pour leurs implantations. Plus précisément, MPI propose plusieurs méthodes de communication point à point telles que *MPI_Send* (envoi) et *MPI_Recv* (réception) ainsi que des communications collectives comme, par exemple, *MPI_Barrier* (barrière) et *MPI_Broadcast* (diffusion). En outre, afin d'accroître la portabilité des applications, MPI propose des types de données de base et donne la possibilité de créer des types complexes et structurés [34].

La version 2.0 de MPI apparue en juillet 1997, apportait la gestion dynamique des processus, le support des communications dites unilatérales (réalisant un accès direct à une mémoire distante sans intervention du processus distant) ou le support d'entrées/sorties parallèles. La version 3.0 apparue en septembre 2012 a apporté des changements et des ajouts importants par rapport à la version 2. Les plus importants sont les communications collectives non bloquantes, la révision de l'implémentation des copies mémoire à mémoire, l'ajout de

l'interface du Fortran (2003-2008) et l'interfaçage avec des d'outils externes (pour le débogage et les mesures de performance). La prochaine version de MPI (4.0) est en cours de discussion et doit apporter des fonctionnalités très importantes, parmi lesquelles : des extensions pour mieux supporter les modèles de programmation hybrides, la prise en charge de la tolérance aux pannes et les accès directe au mémoires.

Les applications MPI créent plusieurs processus qui peuvent exécuter le même programme (modèle SPMD) ou un programme différent (modèle MPMD : Multiple Program Multiple Data) avec des données différentes. Au cours de l'exécution les processus s'échangent des messages suivant différents modes de communications : bloquant, non-bloquant, bufferisé, synchrone.

Il existe de nombreuses implémentations du standard, cependant la plupart d'entre elles sont dérivées de l'une des deux implémentations libres MPICH2 [35, 36] et OpenMPI [37].

1.5 Mesure de performances des architectures multiprocesseurs

Nous allons dans cette section introduire le concept des modèles de calcul liés aux machines multiprocesseurs. L'accent est mis ici sur les aspects de calcul des éléments de traitement (processeurs) et non sur les autres éléments à savoir les types de communications. Deux modèles de calcul sont évoqués, Le premier concerne les processus à durée égale ou en anglais embarrassingly parallel problem tandis que le deuxième concerne le modèle de calcul parallèle contenant des sections en série. Deux mesures seront introduite dans cette étude, ce sont l'accélération et l'efficacité. L'impact de la surcharge due aux communications inter-processeurs sur les performances de l'accélération des systèmes multiprocesseurs est souligné dans ces modèles [38].

1.5.1 Modèle de calcul à durée égale

Dans ce modèle, On admettra qu'une tâche donnée peut être divisée en n sous-tâches égales, chacune pouvant être exécutée par un processeur. Si t_s est le temps d'exécution de la tâche entière en utilisant un seul processeur, alors le temps nécessaire à chaque processeur pour exécuter sa sous-tâche est $t_m = t_s/n$. Le facteur d'accélération d'un système parallèle peut être défini comme le rapport entre le temps d'exécution d'un seul processeur pour résoudre un problème donnée et le temps d'exécution d'un système parallèle composé de n

processeurs pour résoudre le même problème [38–40].

$$S(n) = \frac{t_s}{t_m} \tag{1.1}$$

$$= \frac{t_s}{t_s/n} = n \tag{1.2}$$

L'équation (1.1) indique que, selon le modèle de calcul à durée égale, le facteur d'accélération résultant de l'utilisation de n processeurs est égal au nombre de processeurs utilisés, n. Cependant, Un facteur important a été négligé dans la dérivation ci-dessus. Ce facteur est le surcoût de communication, qui résulte du temps nécessaire aux processeurs pour communiquer et éventuellement échanger des données lors de l'exécution de leurs sous-tâches. Supposons que le temps dû aux communications est t_c, alors le temps réel pris par chaque processeur pour exécuter sa sous-tâche est donné par :

$$S(n) = \frac{t_s}{t_m} \tag{1.3}$$

$$= \frac{t_s}{t_s/n + t_c} = \frac{n}{1 + n \times \frac{t_c}{t_s}} \tag{1.4}$$

Afin de normaliser la valeur de l'accélération dans l'intervalle $[0,1]$, on divise sa valeur par n. La valeur obtenue est appelée *efficacité*, ξ.

Bien que simple, le modèle de durée égale est cependant irréaliste, du fait qu'il est basé sur l'hypothèse pratiquement irréalisable qu'un problème donnée peut être divisée en un nombre de sous-tâches à durée égales qui peuvent être exécutées par un certain nombre de processeurs en parallèle. En effet, les algorithmes pratiques contiennent toujours des parties séquentielles qui ne peuvent pas être parallélisées. Ceci nous amène à l'étude du modèle du Calcul parallèle avec des sections sérielles.

1.5.2 Modèle du Calcul parallèle avec des sections sérielles

Dans ce modèle de calcul, on suppose qu'une fraction f de la tâche donnée (calcul) n'est pas divisible en sous-tâches concurrentes. Alors, la partie restante $(1-f)$ est parallélisable. En suivant le même raisonnement, le temps d'exécution sur n processeurs est : $t_m = f t_s + (1-f)(t_s/n)$. Le facteur d'accélération est donné par :

$$S(n) = \frac{t_s}{f t_s/n + (1-f)\frac{t_s}{n}} = \frac{n}{1 + (n-1)f} \tag{1.5}$$

Selon cette équation, l'accélération potentielle due à l'utilisation de n processeurs est déterminé principalement par la fraction de code qui ne peut être parallélisée. Si la tâche (programme) est complètement sérielle, c'est-à-dire $f = 1$, aucune accélération ne peut être atteinte quel que soit le nombre de processeurs utilisés. Ce principe est connu comme la loi d'Amdahl [41]. Il est intéressant de noter que selon cette loi, le facteur d'accélération maximum est donné par

$$\lim_{n\to\infty} S(n) = \frac{1}{f} \tag{1.6}$$

Par conséquent, selon la loi d'Amdahl, l'amélioration de la performance (vitesse) d'un algorithme parallèle sur un algorithme séquentiel n'est pas limitée par le nombre de processeurs employés mais par la fraction de l'algorithme qui ne peut pas être parallélisée.

La figure 1.6 présente la courbe de l'accélération en fonction du nombre de processeurs dans le cas où le surcoût des communications n'est pas pris en compte. Comme indiqué

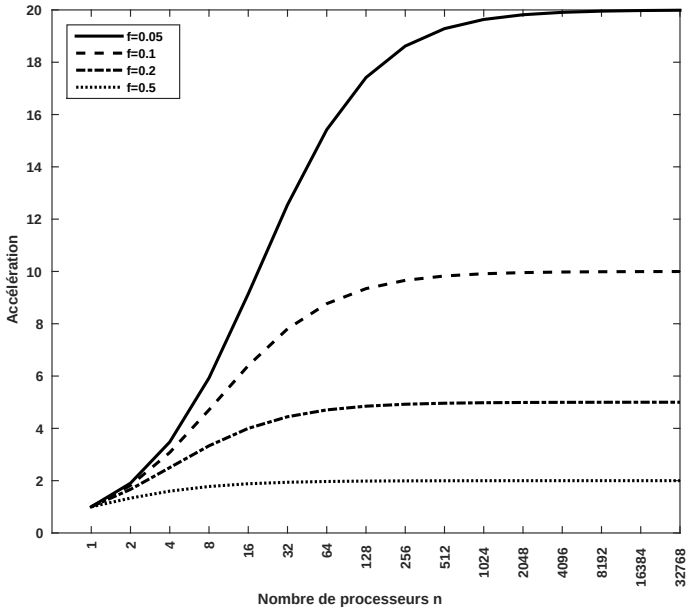


FIGURE 1.6 – Courbe de l'accélération idéale (sans surcoût des communications) selon la loi d'Amdahl pour différentes valeurs de f

précédemment, le surcoût des communications doit être inclue dans le temps de traitement. Ainsi, le facteur d'accélération est donné par

$$S(n) = \frac{t_s}{f t_s + (1-f)(t_s/n) + t_c} = \frac{n}{1 + (n-1)f + n(t_c/t_s)} \tag{1.7}$$

Et le facteur d'accélération maximum est

$$\lim_{n\to\infty} S(n) = \frac{1}{f + (t_c/t_s)} \tag{1.8}$$

La figure 1.7 présente la courbe de l'accélération en fonction du nombre de processeurs dans le cas réél, c'est-à-dire, dans le cas où le surcoût des communications est pris en compte pour une valeur de $f = 0.05$. On notera que l'allure des clurbes reste la même pour les autres valeurs de f.

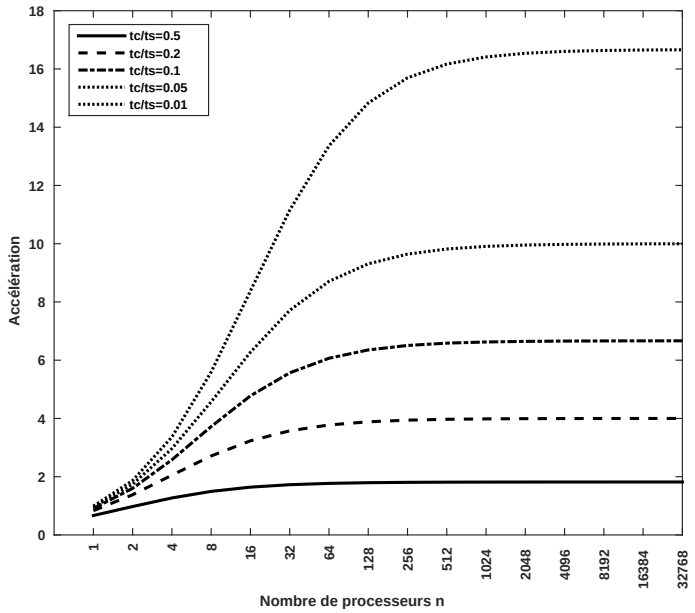


FIGURE 1.7 – Courbe de l'accélération avec prise en compte du surcoût des communications selon la loi d'Amdahl pour différentes valeurs de t_c/t_s pour $f = 0.05$

1.6 MATLAB et calcul parallèle

MATLAB est largement utilisé dans les domaines académique, recherche scientifique et industriel, et est devenu un outil important utilisé par de nombreux scientifiques et ingénieurs de traitement du signal et de l'image. L'attrait de MATLAB est qu'il combine un langage de programmation scientifique facile à utiliser avec un environnement commun pour le prototypage, le codage et la visualisation. Une grande variété de boîtes à outils (toolbox) complémentaires traitant de plusieurs domaines d'application spécialisés, ainsi que d'une communauté d'utilisateurs très large et active, font de MATLAB la plate-forme logicielle de choix pour de nombreuses applications.

De nombreux algorithmes largement utilisés nécessitent des calculs intensifs et peuvent ainsi bénéficier des plateforme de calcul intensif pour leur exécution. L'approche traditionnelle pour gérer ces grands problèmes était de traduire le code MATLAB en C/C++ ou FORTRAN, paralléliser le code résultant en utilisant MPI ou OpenMP, puis l'exécuter sur une plate-forme HPC. Cependant, ce processus s'avère une activité coûteuse, propice aux erreurs et qui prend beaucoup de temps. En outre, il est très difficile de propager les modifications du code MATLAB dans code C correspondant, d'autant plus qu'une seule ligne de code MATLAB peut correspondre à plusieurs lignes de code C [42].

En 2004, MATLAB a introduit le Distributed Computing Toolbox (DCT) qui permet l'utilisation du code MATLAB dans un cluster sous le contrôle d'un planificateur (ou scheduler en anglais) offrant la possibilité de "paralléliser" un code donné avec des modifications minimales. Le DCT est exécuté sur le poste client alors qu'un autre programme appelé Matlab Distributed Computing Engine (MDCE) est déployé dans les nœuds du cluster. Plus tard, le DCT a été étendue pour l'utilisation des machines multicores [43].

Dans la version 2008 et ultérieurs, le toolbox a été renommé en PCT pour parallel computing toolbox. Dans les versions antérieur à R2014a, PCT autorisait jusqu'à 12 workers locales sur une machine multi-cœurs. Cependant, dans les versions ultérieurs, le nombre de workers locales n'est pas limités. MatLab a également introduit MDCS ou (Matlab Distributed Computing Server) pour le déploiement sur une grille de calcul ou cluster d'ordinateurs. L'architecture générale d'un cluster de calcul Matlab (Figure 1.8) est basé sur l'installation de MDCS sur le nœud principal et les nœuds de calcul. Sur le poste client ; où l'utilisateur développe les applications et soumets les calculs au cluster, seul PCT est nécessaire.

1.7 Conclusion

Dans ce chapitre, nous avons présenté les notions fondamentales du calcul parallèle. Les architectures matérielles peuvent être classées selon plusieurs critères, mais la plus répandue est celle de Flynn basée sur la notion de flux de données et d'instructions. Une grappe de calcul ou cluster se construit autour d'un ensemble de nœuds de calcul interconnectés par un réseau à haute performance. Cette architecture popularisé par les clusters de type Beowulf est la plus utilisée dans le domaine de calcul scientifique haute performance. L'ensemble des applications de calcul haute performance, s'exécutant sur ces nœuds, utilise la bibliothèque par passage de message MPI.

Les différents modèles de programmation mis au point au cours des précédentes décen-

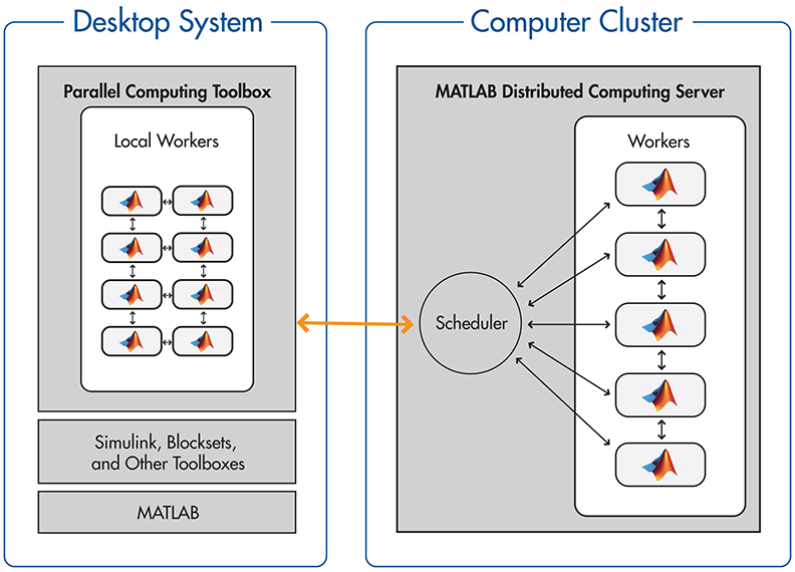


FIGURE 1.8 – Architecture générale d'un cluster Matlab

nies ont été conçus pour des architectures homogènes et rarement hybrides. Ces modèles de programmation sont utilisés pour programmer les architectures parallèles contemporaines. Ils se heurtent au problème complexe des affinités matérielles qui varient selon le matériel et les constructeurs et exhibent des effets parfois difficiles à prédire.

Il existe de nombreux critères pour caractériser les performances d'une application parallèle. Les plus utilisés sont le temps d'exécution ou de communication, l'accélération, l'efficacité, ...).

Chapitre 2

Calcul scientifique en électromagnétique CEM

Sommaire

2.1 Introduction 26
2.2 Classification des problèmes électromagnétiques 28
2.3 Les équations intégrales 29
2.4 Transformation d'une équation différentielle en une équation intégrale 30
2.5 Les fonctions de Green 31
2.6 Les méthodes classiques pour les problèmes aux limites 32
2.7 La méthode des moments (MoM) 36
2.8 La méthode des différences finies dans le domaine temporelle (FDTD) 38
2.9 La méthode des éléments finis (FEM) 49
2.10 Calcul parallèle en électromagnétisme - biographie et état de l'art . . 56
2.11 Conclusion 61

2.1 Introduction

Les dispositifs et systèmes électromagnétiques, allant des appareils de bureau ordinaires aux téléphones cellulaires, font partie intégrante de la vie moderne. Le développement continu de la nouvelle technologie dépend grandement de l'analyse technique et de la synthèse des systèmes électromagnétiques. Ceux-ci sont basés sur l'obtention de solutions précises des équations de Maxwell pour le système étudié. L'intensification de la recherche

et du développement dans une grande variété d'applications, notamment la conception d'antennes, les circuits hyperfréquences, la photonique, l'imagerie médicale, les communications sans fil, la compatibilité/interférence électromagnétique, etc. ont conduit à des systèmes de plus en plus complexes. Dans de nombreux cas, un seul dispositif est devenu une structure très complexe qui comprend un certain nombre de conducteurs, de diélectriques et de semi-conducteurs de formes arbitraires et de nature physique complexe. Les technologies de fabrication onéreuses excluent la possibilité de modifier un dispositif si ses performances ne sont pas conformes aux spécifications de conception. Par conséquent, il est nécessaire de développer des méthodes capables de caractériser de manière extrêmement précise ces systèmes pour analyser les structures qui nous intéressent. Cependant, il est difficile d'obtenir de telles méthodes en raison de la nature complexe de n'importe quel des dispositifs électromagnétiques modernes.En plus, l'analyse de tels systèmes ne conduit pas généralement à des formules analytiques exactes [3].

Deux types de méthode existent pour la résolution « exacte » des équations de Maxwell, résolues dans ce cas en respectant exactement les expressions des équations différentielles d'origine. La méthode dite « intégrale » est utilisée par la MoM car elle résout les équations de Maxwell sous leur forme intégrale. Elle ne nécessite généralement pas de mailler le volume de calcul total mais seulement les structures filaires et les surfaces. Les méthodes dites différentielles résolvent les équations aux dérivées partielles (FDTD et FEM). Ce sont des méthodes dites volumiques car elles travaillent généralement sur un volume englobant l'objet à traiter et fermés par des frontières absorbantes simulant l'espace libre [44].

En effet, les solutions analytiques exactes ne sont connues que pour un nombre limité de cas particuliers, qui ne sont pratiquement jamais directement applicables à des applications réelles. L'impossibilité de dériver des solutions analytiques exactes à partir des équations de Maxwell pour des applications réelles a conduit à une recherche intensive sur les techniques numériques. L'application de méthodes numériques aux problèmes de champs électromagnétiques est connue sous le nom de calcul en électromagnétisme (Computational ElectroMagnetics,CEM), et de ce fait, elle est une extension naturelle de l'approche analytique pour résoudre les équations de Maxwell [2, 3].

Parmi les méthodes rapportés dans cette partie, la MoM est considéré comme l'une des méthodes numériques basée sur l'équation intégrale les plus utilisées. Elle coïncide formellement avec la méthode des résidus pondérés car les sources (inconnues) sont développées par une somme de certaines fonctions de base multipliées par des coefficients inconnus. Le résidu de l'équation intégrale est pondéré en utilisant un produit interne approprié et un ensemble de fonctions de pondération. Cela résulte en un ensemble d'équations linéaires

pouvant être résolues par les méthodes classiques de l'algèbre linéaire.

Pour l'analyse du rayonnement des structures conductrices, il est difficile de surpasser la précision et l'efficacité d'une méthodologie de calcul basé sur l'équation intégrale. Pour les structures composites contenant à la fois des conducteurs et des diélectriques, où l'inhomogénéité diélectrique n'est pas grande, la méthode de l'équation intégrale est toujours une excellente procédure numérique. Outre la méthode des moments, un certain nombre de variantes cette dernière ont été développées. Par exemple, la FMM et le MLFMA ont été proposés pour accélérer le produit matrice-vecteur qui apparaît dans la solution itérative des équations MoM [3, 9].

La FEM est une méthode numérique utilisée pour résoudre des problèmes aux limites caractérisés par une équation différentielle partielle et un ensemble de conditions aux limites. La méthode a été proposée par Courant en 1943 [8] pour résoudre les problèmes variationnels de la théorie potentielle [12]. Par la suite, la méthode a été développée et largement appliquée aux problèmes d'analyse structurelle et de plus en plus aux problèmes dans d'autres domaines. Aujourd'hui, la méthode des éléments finis est reconnue comme une méthode générale prééminente applicable à une grande variété de problèmes d'ingénierie et de mathématiques, y compris ceux de l'ingénierie des antennes et des micro-ondes.

La méthode FDTD est une méthode de calcul de différences finies dans le domaine temporel pour les problèmes électromagnétiques. La méthode a été développée par K.S., Yee en 1966 [45]. Avec l'avènement d'ordinateurs puissants et peu coûteux et l'amélioration de la méthode elle-même, la FDTD est devenue l'une des méthodes les plus utilisées pour résoudre les équations de Maxwell pour des applications scientifiques et d'ingénierie.

2.2 Classification des problèmes électromagnétiques

Les problèmes électromagnétiques sont classés en termes d'équations les décrivant. Les équations peuvent être différentielles ou intégrales ou les deux. La plupart des problèmes électromagnétiques peuvent être énoncés en termes d'équation avec opérateur (2.1) [11].

$$L(\phi) = g \tag{2.1}$$

où L est un opérateur (différentiel, intégral ou intégra-différentiel), g est l'excitation ou la source connue, et ϕ est la fonction inconnue à déterminer. Un exemple typique est le problème électrostatique impliquant l'équation de Poisson. En forme différentielle, l'équation

(2.1) devient (2.2)

$$-\nabla^2 V = \frac{\rho_v}{\varepsilon} \tag{2.2}$$

Nous avons alors $L = -\nabla^2$ l'opérateur laplacien, $g = \rho_v/\varepsilon$ est le terme de la source, et $\phi = V$ est le potentiel électrique. Sous forme intégrale, l'équation de Poisson est s'écrit sous la forme (Équation 2.3).

$$V = \int \frac{\rho_v \partial v}{4\pi\varepsilon r^2} \tag{2.3}$$

2.3 Les équations intégrales

L'objectif de la méthode de l'équation intégrale (Integral Equation, IE) pour le rayonnement ou la diffusion est de formuler la solution pour la densité de courant inconnue, induite à la surface du dispositif radiateur/diffuseur, sous la forme d'une équation intégrale où la densité de courant induit fait partie de l'intégrante. L'équation intégrale est ensuite résolue pour la densité de courant induite inconnue en utilisant des techniques numériques telles que la MoM [46]. Parmi les exemples simples d'équations intégrales, on cite les transformés de Fourier, Laplace et Hankel [11].

Les équations intégrales linéaires les plus fréquemment étudiées appartiennent à deux catégories nommées Fredholm et Volterra [47]. La première classe comprend les équations de Fredholm du premier, deuxième et troisième type données par [11] :

$$f(x) = \int_a^b K(x,t)\phi(t)dt \tag{2.4}$$

$$f(x) = \phi(x) - \lambda \int_a^b K(x,t)\phi(t)dt \tag{2.5}$$

$$f(x) = a(x)\phi(x) - \lambda \int_a^b K(x,t)\phi(t)dt \tag{2.6}$$

Dans tous les cas, $\phi(t)$ est la fonction inconnue. $K(x,t)$, que nous appelons le noyau de l'équation intégrale, et $f(x)$ sont supposés être connus. Lorsque $f(x) = 0$, l'équation est dite homogène.

La deuxième classe d'équations intégrales sont les équations de Volterra de la première,

deuxième et troisième type, données par :

$$f(x) = \int_a^x K(x,t)\phi(t)dt \tag{2.7}$$

$$f(x) = \phi(x) - \lambda \int_a^x K(x,t)\phi(t)dt \tag{2.8}$$

$$f(x) = a(x)\phi(x) - \lambda \int_a^x K(x,t)\phi(t)dt \tag{2.9}$$

Les équations (2.5) à (2.9) sont toutes des équations linéaires par rapport à la fonction ϕ.

2.4 Transformation d'une équation différentielle en une équation intégrale

Souvent, nous pouvons représenter le problème physique soit par une équation différentielle ou par une équation intégrale. La plupart des équations différentielles ordinaires peuvent être exprimées en équations intégrales, mais l'inverse n'est pas vrai. Alors que les conditions aux limites sont imposées de manière externe dans les équations différentielles, elles sont incorporées dans une équation intégrale. Prenons l'exemple d'une équation différentielle ordinaire (Ordinary Differential Equation ODE) de la forme [47] :

$$y'' + A(x)y' + B(x)y = g(x) \tag{2.10}$$

avec les conditions initiales :

$$y(a) = y_0, \qquad y'(a) = y'_0$$

Si on intègre, nous obtenons :

$$y'(x) = -\int_a^x A(t)y'(t)dt - \int_a^x B(t)y(t)dt + \int_a^x g(t)dt + y'_0 \tag{2.11}$$

En intégrant le premier intégrale de droite par parties, nous obtenons :

$$y'(x) = -Ay(x) \int_a^x (B - A')y(t)dt + \int_a^x g(t)dt + A(a)y_0 + y'_0 \tag{2.12}$$

Les conditions initiales sont incorporées dans la nouvelle version de l'équation 2.12. Si on effectue une deuxième intégration, nous obtenons :

$$y(x) = -\int_a^x Aydx - \int_a^x du \int_a^u [B(t) - A'(t)]\, y(t)dt + \int_a^x du \int_a^u g(t)dt + [A(a)y_0 + y_0'](x-a) + y_0 \quad (2.13)$$

Si on utilise la relation (2.14) dans l'équation (2.13), nous obtenons :

$$\int_a^x du \int_a^u f(t)dt = \int_a^x (x-t)f(t)dt \quad (2.14)$$

$$y(x) = -\int_a^x (A(t) + (x-t)[B(t) - A'(t)])\, y(t)dt + \int_a^x (x-t)g(t)dt + [A(a)y_0 + y_0'](x-a) + y_0 \quad (2.15)$$

Si on introduit les abréviations :

$$\begin{aligned} K(x,t) &= (t-x)[B(t) - A'(t)] - A(t) \quad (2.16)\\ f(x) &= \int_a^x (x-t)g(t)dt + [A(a)y_0 + y_0'](x-a) + y_0 \end{aligned}$$

L'équation (2.15) devient :

$$y(x) = f(x) + \int_a^x K(x,t)y(t)dt \quad (2.17)$$

L'équation (2.17) est une équation Voltera de type deux.

2.5 Les fonctions de Green

Un moyen plus systématique d'obtenir une IE à partir d'une équation différentielle partielle (Partial Differential Equation PDE) consiste à construire une fonction auxiliaire connue sous le nom de fonction de Green pour ce problème [11]. Une telle fonction a été introduite par George Green dès 1828. Par la suite, la méthode des fonctions de Green est devenue une méthode analytique très utile en mathématiques et dans de nombreuses sciences appliquées [48]. La méthode des fonctions de Green peut être appliquée, en général, à n'importe quelle équation différentielle partielle, non homogène, à coefficient constant et linéaire, avec n'importe quel nombre de variables indépendantes.

Pour obtenir le champ produit par une source distribuée par la technique des fonction de Green, il faut déterminer l'effet de chaque partie élémentaire de la source et les additionner. Si $G(\boldsymbol{r}, \boldsymbol{r}')$ est le champ au point d'observation (ou point de champ) $\boldsymbol{r}$ provoqué par une source ponctuel au point source $\boldsymbol{r}'$, alors le champ en $\boldsymbol{r}$ produit par une distribution de source $g(\boldsymbol{r}')$ est l'intégrale de $g(\boldsymbol{r}')G(\boldsymbol{r}, \boldsymbol{r}')$ sur la plage de $\boldsymbol{r}'$ occupée par la source. La fonction G est la fonction de green.

Le vrai problème revient à construire les fonctions de Green pour le problème étudié. Considérons la PDE linéaire de second ordre

$$L\phi = g \tag{2.18}$$

Les fonctions de green correspondants à l'opérateur différentiel L sont définies comme la solution de l'équation inhomogène du point source suivante :

$$LG(\boldsymbol{r}, \boldsymbol{r}') = \delta(\boldsymbol{r}, \boldsymbol{r}') \tag{2.19}$$

où $\boldsymbol{r}$ et $\boldsymbol{r}'$ sont les vecteurs de position du point champ (x, y, z) et le point source (x', y', z') respectivement et $\delta(\boldsymbol{r}, \boldsymbol{r}')$ et la fonction delta de Dirac qui satisfait la propriété de l'équation 2.1.

$$\int \delta(\boldsymbol{r}, \boldsymbol{r}')g(\boldsymbol{r}')dv' = g(\boldsymbol{r}) \tag{2.20}$$

La fonction de Green a les propriétés suivantes [11, 48] :

(a) G satisfait l'équation $LG = 0$ sauf au point source $\boldsymbol{r}'$, c'est-à-dire,

$$LG(\boldsymbol{r}, \boldsymbol{r}') = \delta(\boldsymbol{r}, \boldsymbol{r}')$$

(b) G est symétrique dans le sens où

$$LG(\boldsymbol{r}, \boldsymbol{r}') = LG(\boldsymbol{r}', \boldsymbol{r}) \tag{2.21}$$

(c) G vérifie la valeur aux limites imposée à f sur B, c'est-à-dire

$$G = f \qquad \text{sur} \quad B \tag{2.22}$$

(d) La dérivée directionnelle $\frac{\partial G}{\partial n}$ a une discontinuité à laquelle est spécifié par l'équation

$$\lim_{\varepsilon \to 0} \oint_S \frac{\partial G}{\partial n} dS = 1 \tag{2.23}$$

où n est la normale vers l'extérieur à la sphère de rayonε.

Le tableau (2.1) présente des exemples typiques des fonctions de green généralement utilisées des les problèmes électromagnétiques ([11]).

TABLE 2.1 – Fonctions de Green dans l'espace libre en 3D.

Opérateur	Région de solution	Fonction de Green
équation de Laplace	$\nabla^2 G = \delta(\boldsymbol{r}, \boldsymbol{r}')$	$-\frac{1}{4\pi(\boldsymbol{r},\boldsymbol{r}')}$
équation de l'onde (Helmholtz's)	$\nabla^2 G + k^2 G = \delta(\boldsymbol{r}, \boldsymbol{r}')$	$-\frac{exp(jk\lvert\boldsymbol{r},\boldsymbol{r}'\rvert)}{4\pi\lvert\boldsymbol{r},\boldsymbol{r}'\rvert}$
équation de l'onde (Helmholtz's modifiée)	$\nabla^2 G - k^2 G = \delta(\boldsymbol{r}, \boldsymbol{r}')$	$-\frac{exp(-k\lvert\boldsymbol{r},\boldsymbol{r}'\rvert)}{4\pi\lvert\boldsymbol{r},\boldsymbol{r}'\rvert}$

2.6 Les méthodes classiques pour les problèmes aux limites

Nous allons dans la suite considérer le problème aux limites régi par l'équation différentielle

$$L\phi = f \tag{2.24}$$

Dans cette section, nous allons présenter très brièvement deux méthodes dites classiques pour la solution des problèmes aux limites. La première est la méthode variationnelle de Ritz et l'autre est la méthode de Galerkin. Ces deux méthodes sont à la base de la méthode des éléments finis [49].

2.6.1 La méthode de Rayleigh-Ritz

La méthode de Ritz aussi connue comme la méthode de Rayleigh-Ritz est une méthode variationnelle dans laquelle le problème aux limites est exprimé en terme d'expression variationnelle nommée fonctionnel. La solution approximative est obtenue en minimisant le fonctionnel. La méthode a été présentée pour la première fois par Rayleigh en 1877 et étendue par Ritz en 1909 [11, 48–50].

Définissons un produit interne comme suit :

$$< \phi, \phi >= \int_{\Omega} \phi\phi^* d\Omega \tag{2.25}$$

le caractère * représente le conjugué complexe. Si l'opérateur L est auto-adjoint, c'est-à-dire,

$$< L\phi, \psi >=< \phi, L\psi > \tag{2.26}$$

et s'il est positif défini, à savoir,

$$< L\phi, \phi > \begin{cases} > 0, & \phi \neq 0 \\ = 0, & \phi = 0 \end{cases} \tag{2.27}$$

La solution de l'équation (2.24) équivaut à trouver le minimum du fonctionnel quadratique de l'équation (2.28) par rapport à $\tilde{\phi}$ qui représente une fonction de test [48, 49].

$$F(\tilde{\phi}) = \frac{1}{2} < L\tilde{\phi}, \tilde{\phi} > -\frac{1}{2} < \tilde{\phi}, f > -\frac{1}{2} < f, \tilde{\phi} > \tag{2.28}$$

Une fois le fonctionnel trouvé, la solution peut être obtenue par la procédure décrite ci-dessous [49].

Nous allons dans cette étude considérer le problème à valeur réelle. Supposons que $\tilde{\phi}$ dans (2.28) peut être approximée par le développement

$$\tilde{\phi} = \sum_{j=1}^{N} c_j v_j = \{c\}^T\{v\} = \{v\}^T\{c\} \tag{2.29}$$

où les v_j sont les fonctions de base ou de développement définies sur tous le domaine et c_j sont des coefficients constants à déterminer. $\{\cdot\}$ dénote un vecteur colonne et l'indice T dénote le transposé du vecteur. En substituant (2.29) dans (2.28), nous obtenons

$$F = \frac{1}{2}\{c\}^T \int_\Omega \{v\} L \{v\}^T d\Omega \{c\} - \{c\}^T \int_\Omega \{v\} f d\Omega \tag{2.30}$$

Pour minimiser $F(\tilde{\phi})$, on met ses dérivées partielles par rapport à c_j à zéro. Nous obtenons alors un ensemble d'équations algébriques linéaires (2.31)

$$\begin{aligned} \frac{\partial F}{\partial c_i} &= \frac{1}{2}\int_\Omega v_i L\{v\}^T d\Omega\{c\} + \frac{1}{2}\{c\}^T \int_\Omega \{v\} L v_i d\Omega - \int_\Omega v_i f d\Omega \\ &= \frac{1}{2}\sum_{j=1}^{N} c_j \int_\Omega (v_i L v_j + v_j L v_i) d\Omega - \int_\Omega v_i f d\Omega \\ &= 0 \qquad i = 1, 2, 3\cdots, N \end{aligned} \tag{2.31}$$

Le système (2.31) peut être écrit sous la forme :

$$[S]\{c\} = \{b\} \tag{2.32}$$

Les éléments de $[S]$ sont donnés par :

$$S_{ij} = \frac{1}{2}\int_{\Omega}(v_i L v_j + v_j L v_i)d\Omega \tag{2.33}$$

et les éléments de $\{b\}$ par :

$$b_i = \int_{\Omega} v_i f d\Omega \tag{2.34}$$

Il est évident que la matrice $[S]$ est symétrique. En se référant à la propriété d'auto-adjoint de l'opérateur L, S_{ij} peut s'écrire comme suit :

$$S_{ij} = \int_{\Omega} v_i L v_j d\Omega \tag{2.35}$$

Une solution approchée de l'équation (2.24) est donnée par l'équation (2.29), avec les c_i sont obtenus par la solution de le système linéaire de l'équation (2.32).

2.6.2 La méthode de Galerkin

La méthode de Rayleigh-Ritz est applicable dans le cas où un fonctionnel existe. Dans les cas où une telle fonction ne peut être trouvée, Il est alors possible d'utiliser l'une des techniques désignées collectivement comme la méthode des résidus pondérés. Cette famille de méthodes est plus générale et a un champ d'application plus large que la méthode de Rayleigh-Ritz du fait qu'elle n'est pas limitée à une classe de problèmes variationnels [11]. La méthode de Galerkin appartient à cette famille. Le principe est basé sur la recherche de solution en pondérant le résidus de l'équation différentielle.

Admettons que $\tilde{\phi}$ est une solution approchée de l'équation (2.24). En substituant $\tilde{\phi}$ pour ϕ dans l'équation (2.24), nous allons avoir un résidus non nul [49]

$$r = L\tilde{\phi} - f \neq 0 \tag{2.36}$$

La meilleur solution pour $\tilde{\phi}$ serait celle qui minimise le résidus r sur tous le domain Ω. Dans cette famille, on va choisir des fonctions de pondération w_m de telle manière à ce que

l'intégral des résidus pondérés de l'approximation soit zéro, ç'est-à-dire,

$$R_i = \int_\Omega w_i r d\Omega = 0 \tag{2.37}$$

Dans la méthode de Galerkin, les fonctions de pondération sont choisies pour être les mêmes que les fonctions de base pour la solution approchée. Dans le cas où l'opérateur L dans l'équation (2.24) est auto-adjoint, le système obtenu est exactement le même que celui de la méthode de Rayleigh-Ritz.

2.7 La méthode des moments (MoM)

La méthode des moments est une procédure générale pour résoudre l'équation (2.1). La méthode doit son nom au processus de calcul des moments en multipliant avec des fonctions de pondération appropriées et en intégrant ensuite. La méthode a été appliquée avec succès à une grande variété de problèmes électromagnétiques pratiques tels que les rayonnements dus aux éléments et aux réseaux à fils fins, les problèmes de diffusion, l'analyse des micro-rubans et des structures à pertes, la propagation sur une terre non homogène, etc [11].

En électromagnétisme, elle s'applique typiquement à la formulation intégrale du champ électrique (Electric Field Integral Equation, EFIE) pour laquelle les inconnues sont la distribution de courant circulant sur les conducteurs ou, dans le cas de structures planaires multicouches, sur les rubans placés aux interfaces. Le fondement de la MoM consiste à proposer une solution sous la forme d'une somme de fonctions connues auxquelles sont associés des coefficients inconnus. Il s'agit ensuite d'appliquer une procédure de minimisation de l'erreur résiduelle pour générer un système matriciel et déterminer les coefficients inconnus [44].

La procédure pour appliquer la MoM à la résolution de l'équation 2.1 implique généralement quatre étapes [11] :

1. dérivation de l'équation intégrale appropriée,
2. conversion (discrétisation) de l'IE en une équation matricielle à base (ou expansions) fonctions et fonctions de pondération (ou de test),
3. l'évaluation des éléments de la matrice, et
4. résoudre l'équation matricielle et obtenir les paramètres souhaités.

2.7.1 Principe de la méthode

On considère l'équation non homogène [6]

$$L\phi = f \tag{2.38}$$

Supposons que l'opérateur L est régulier, alors l'équation 2.38 a une solution unique donnée par :

$$\phi = L^{-1}f \tag{2.39}$$

La méthode des moments permet d'obtenir une approximation de L^{-1}, et par conséquent de phi.

Soit $\phi_1, \phi_2, \ldots$ une base de fonctions définies dans le domaine d'existence de l'opérateur L. On a alors :

$$\phi = \sum_n \alpha_n \phi_n \tag{2.40}$$

où α_n sont des constantes. Les ϕ_n sont appelées les fonctions de développement ou fonctions de base. Pour une solution exacte, la sommation est faite à l'infini. Pour une approximation, n est fini. En remplaçant (2.40) dans (2.39) et en utilisant la linéarité de l'opérateur L, on obtient

$$\sum_n \alpha_n L(\phi_n) = f \tag{2.41}$$

On suppose qu'un produit interne $< \phi, f >$ adéquat est définit pour le problème à résoudre. On définit un ensemble de fonctions de pondération appelées aussi fonctions de tests $w_1, w_2, w_3, \ldots$ dans le domaine de définition de L. Prenons le produit interne de l'équation (2.41) avec les éléments w_m, on obtient alors [6]

$$\sum_n \alpha_n < w_m, L(\phi_n) > = < w_m, f > \qquad m = 1, 2, 3, \ldots \tag{2.42}$$

Le système d'équations de (2.42) peut être écrit sous format matriciel

$$[l_{mn}][\alpha_n] = [f_m] \tag{2.43}$$

où

$$[l_{mn}] = \begin{bmatrix} < w_1, L(\phi_1) > & < w_1, L(\phi_2) > & \cdots & < w_1, L(\phi_n) > \\ < w_2, L(\phi_1) > & < w_2, L(\phi_2) > & \cdots & < w_2, L(\phi_n) > \\ \vdots & \ddots & \ddots & \vdots \\ < w_m, L(\phi_1) > & < w_m, L(\phi_2) > & \cdots & < w_m, L(\phi_n) \end{bmatrix} \tag{2.44}$$

$$[\alpha_n] = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{bmatrix} \qquad [f_n] = \begin{bmatrix} < w_1, f > \\ < w_2, f > \\ \vdots \\ < w_m, f > \end{bmatrix} \tag{2.45}$$

Si la matrice $[l]$ est non singulière, son inverse $[l^{-1}]$ existe. Les valeurs α_n sont données par

$$[\alpha_n] = [l^{-1}_{mn}][f_m] \tag{2.46}$$

La solution pour f est donnée par (2.40). Si on note le vecteur de fonctions

$$[\hat{\phi}_n] = [\phi_1 \quad \phi_2 \quad \phi_3 \quad \ldots] \tag{2.47}$$

L'équation (2.40) s'écrit :

$$\phi = [\hat{\phi}_n][\alpha_n] = [\hat{\phi}_n][l^{-1}_{mn}][f_m] \tag{2.48}$$

Le cas particulier où $w_n = \phi_n$ est connu comme la méthode de Galerkin [51].

Il est clair que l'une des tâches principales pour la solution d'un problème donné est le choix des fonctions ϕ_n et w_n. Les ϕ_n doivent être linéairement indépendantes et choisies de façon à ce que ϕ peut être approximée avec une précision acceptable par une certaine combinaison des ϕ_n. Les w_n doivent aussi être linéairement indépendantes et choisies de sorte que les produits $< w_n, f >$ dépendent peux des propriétés de f. D'autres facteur peuvent affecter le choix de ϕ_n et w_n dont la précision de la solution désirée, la simplicité de l'évaluation des éléments de la matrice, la taille de la matrice pouvant être inversée et l'obtention de la matrice $[l]$ qui soit bien conditionnée.

2.8 La méthode des différences finies dans le domaine temporelle (FDTD)

La méthode FDTD est une implémentation particulière d'une classe générale de méthodes connues sous le nom de techniques de différences finies. La méthode FDTD est largement utilisé dans le CEM lorsque des différences finies sont employées. Les méthodes de différences finies sont des méthodes numériques dans lesquelles les dérivées sont directement approchées par des quotients de différences finies. La classe générale de ces méthodes est l'approche numérique la plus intuitive et a été la première à être largement développée par la communauté de calcul scientifique. À ce jour, Elle reste probablement la technique nu-

mérique la plus universellement applicable et la plus largement utilisée dans le domaine des calculs scientifiques [10].

2.8.1 Les différences finies - un aperçu

Comme évoqué auparavant, la méthode des différences finies est un outil important du calcul scientifique pour la résolution d'équations aux dérivées partielles (Partial Differential Equation PDE), dont celles de Maxwell sont bien sûr un exemple fondamental. Traditionnellement, pour la solution numérique des PDE's, la classification est faite selon le type des problèmes, à savoir des problèmes à valeurs initiales ou des problèmes aux limites. Cependant, il existe des PDE's qui nécessitent les deux conditions [10].

Les étapes de base de toute méthode de différence finie peuvent être résumées comme suit :

— Diviser la région de la solution en une grille de nœuds.

— Approximer les dérivées dans l'équation différentielle partielle donnée par les différences finies impliquant la valeur de la solution à différents nœuds.

— Résoudre les équations aux différences finies pour la valeur de la solution à chaque nœud soumis aux conditions aux limites et/ou initiales. Si l'opération est effectuée dans le domaine temporel, cela revient à trouver les valeurs à la prochaine étape. Ce processus est appelé « marche temporelle », « intégration temporelle », ou plus spécifiquement dans le contexte de la FDTD, « saute-moutons ».

Un point central de toute méthode des différences finies est l'approximation des dérivées. A partir de la définition de la dérivée d'une fonction, différentes approximations numériques peuvent être proposées. Cependant, ceux-ci sont généralement dérivés du développement en série de Taylor qui permettent une estimation de l'erreur de l'approximation. Selon que les nœuds « suivant », « précédent » ou « central » sont utilisés, on obtient la différenciation avant, arrière ou centrale définies comme suit :

La formule de différence avant de la dérivée première est donnée par l'équation (2.49).

$$\frac{dU(x)}{dx} = \frac{U(x+\Delta x) - U(x)}{\Delta x} - \frac{(\Delta x)}{2}\frac{d^2U}{dx^2} + \mathcal{O}(\Delta x)^2 \tag{2.49}$$

La formule de différence arrière de la dérivée première est donnée par l'équation (2.49).

$$\frac{dU(x)}{dx} = \frac{U(x) - U(x-\Delta x)}{\Delta x} + \frac{(\Delta x)}{2}\frac{d^2U}{dx^2} + \mathcal{O}(\Delta x)^2 \tag{2.50}$$

La formule de différence centrale de la dérivée première est donnée par l'équation (2.51).

$$\frac{dU(x)}{dx} = \frac{U(x+\Delta x\acute{e}) - U(x-\Delta x)}{\Delta x} - \frac{(\Delta x)^2}{6}\frac{d^3U}{dx^3} + \mathcal{O}(\Delta x)^4 \tag{2.51}$$

2.8.2 Principe de la méthode FDTD

L'algorithme de base de FDTD a été proposé par K.S. Yee en 1966 [7] . En 1975, Taflove et Brodwin ont réussi à obtenir le bon critère de stabilité numérique pour l'algorithme de Yee et ont été les premiers à résoudre les problèmes de diffusion électromagnétique en régime permanent sinusoïdal, en deux et en trois dimensions [52]. Mur [53] a publié la première implémentation numérique stable de la condition aux limites absorbante (Abosorbante Boundary Condition ABC) en 1981. En 1994, Berenger a introduit une des meilleurs implémentation de l'ABC, la couche absorbante parfaitement adaptée (Perfectly matched layer PML) [54].

2.8.3 Algorithme de Yee

L'algorithme de Yee [7] peut être résumé comme suit [55] :

— Remplacer tous les dérivés dans les lois d'Ampère et de Faraday par des différences finies. Discrétiser l'espace et le temps afin que les champs électriques et magnétiques soient décalés dans l'espace et dans le temps.

— Résoudre les équations de différence résultantes pour obtenir des «formules de mise à jour» qui expriment les (inconnus) les valeurs des champs futurs en termes des (connus)les valeurs des champs passés.

— Évaluer les champs magnétiques à pas de temps dans le futur pour que leurs valeurs soient connus dans la pas courant (en fait, ils deviennent des champs passés).

— Évaluer les champs électriques à pas de temps dans le futur pour que leurs valeurs soient connus dans la pas courant (en fait, ils deviennent des champs passés).

— Répétez les deux étapes précédentes jusqu'à ce que les champs obtenus pendant la durée souhaitée.

En utilisant le système des unités international ou le système MKS (Mètre Kilogramme Seconde) et en assumant les paramètres du diélectrique μ, ε et σ constants et indépendants du temps, les équations de Maxwell peuvent être écrites en coordonnées rectangulaires

(x, y, z) comme suit [52, 56] :

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) \tag{2.52}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \tag{2.53}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right) \tag{2.54}$$

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma E_x\right) \tag{2.55}$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} - \sigma E_y\right) \tag{2.56}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma E_z\right) \tag{2.57}$$

La résolution des équations de Maxwell passe par la discrétisation des grandeurs électromagnétiques suivant le schéma de Yee [7] qui utilise les différences centrées pour exprimer les premières dérivées partielles.

Soit $f(x, y, z, t)$ une fonction qui représente une composante du champ électrique $\bar{\mathbf{E}}$ ou du champ magnétique $\bar{\mathbf{H}}$ qui sera notée $F^n(i, j, k)$ La fonction évaluée au nœud i, j, k et à l'instant n s'écrit :

$$F^n(i, j, k) = F(i\Delta x, j\Delta y, k\Delta z) \tag{2.58}$$

où $\Delta x = \Delta y = \Delta z$ correspondent aux incréments (pas) dans l'espace dans les directions x, y, z et Δt correspond à l'incrément temporel. Yee a utilisé des expressions de différences finies pour les dérivées spatiales et temporelles simples à programmer et précises au second ordre par rapport à $\Delta x, y, z$ et Δt, respectivement (2.59) et 2.60).

$$\frac{\partial F^n(i, j, k)}{\partial x} = \frac{F^n(i + \frac{1}{2}, j, k) - F^n(i - \frac{1}{2}, j, k)}{\Delta x} + \mathcal{O}[(\Delta x)^2] \tag{2.59}$$

$$\frac{\partial F^n(i, j, k)}{\partial t} = \frac{F^{n+\frac{1}{2}}(i, j, k) - F^{n-\frac{1}{2}}(i, j, k)}{\Delta t} + \mathcal{O}[(\Delta t)^2] \tag{2.60}$$

L'algorithme est obtenu par application du modèle des équations (2.59) et (2.60) au système des équations (2.52) à (2.57). On obtient alors un nouveau système d'équations discrétisées qui modélise la propagation du champ électromagnétique sur la grille de calcul. Dans un système homogène isotrope, ce système résultant s'écrit comme suit :

Pour atteindre la précision de l'équation (2.59), et de réaliser toutes les dérivées spatiales des

équations (2.52) - (2.57), Yee a positionné les composantes de $\bar{\mathbf{E}}$ et $\bar{\mathbf{H}}$ autour d'une cellule unitaire en treillis comme représenté sur la figure (2.1) [52].

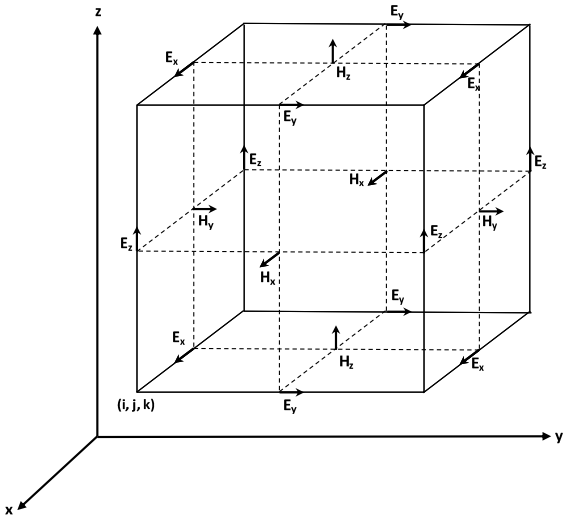


FIGURE 2.1 – Position des composants du champ autour d'une cellule élémentaire du treillis de Yee

Pour atteindre la précision de l'équation (2.60), on évalue $\bar{\mathbf{E}}$ et $\bar{\mathbf{H}}$ en alterné à mi-temps des pas de discrétisation. Le résultat est le système des équations aux différences finies suivantes pour le système (2.52) - (2.57). Commençons par considérer la composante E_x de l'équation (2.55) répétée ici pour plus de commodité :

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} - \sigma E_x\right) \tag{2.61}$$

En se référant à la figure (2.1), on considère une substitution de la différence centrale en temps et en espace des dérivées de l'équation (2.61), par exemple $E_x(i, j+\frac{1}{2}, k+\frac{1}{2}, n)$. on aura :

$$\frac{E_x\big|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - E_x\big|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{\Delta t} = \frac{1}{\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}} \cdot \left(\begin{array}{c} \frac{H_z\big|^n_{(i,j+1,k+\frac{1}{2})} - H_z\big|^n_{(i,j,k+\frac{1}{2})}}{\Delta y} - \frac{H_y\big|^n_{(i,j+\frac{1}{2},k+1)} - H_y\big|^n_{(i,j+\frac{1}{2},k)}}{\Delta z} \\ -\sigma_{(i,j+\frac{1}{2},k+\frac{1}{2})} E_x\big|^n_{(i,j+\frac{1}{2},k+\frac{1}{2})} \end{array} \right) \tag{2.62}$$

On note que tous les valeurs des champs sur le côté droit sont évaluées à l'instant n y compris le champs électrique E_x qui apparaît en raison de la présence du conductivité σ. Étant donné que la valeur de E_x à la valeur n n'est pas sensé être conservé (en effet, juste la va-

leur de E_x à l'instant $n-\frac{1}{2}$ est conservée), il faudrait un moyen pour estimer ce terme. Une solution consiste à utiliser l'approximation semi-implicite [56] :

$$E_x|^{n}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = \frac{E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} + E_x|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{2} \tag{2.63}$$

Il a été établi que l'approximation de l'équation (2.63) produit des résultats numériques stables et précis pour des valeurs de σ allant de zéro à l'infini.

Ici les valeurs E_x à l'instant n sont supposées être simplement la moyenne arithmétique des valeurs de E_x à l'instant $n-\frac{1}{2}$ et la valeur en cours de calcul à l'instant $n+\frac{1}{2}$. En substituant (2.63) dans (2.62) et en multipliant les deux cotés par Δt, on obtient

$$\begin{aligned} E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - E_x|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = \\ \frac{\Delta t}{\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}} \cdot \left(\begin{array}{c} \frac{H_z|^{n}_{(i,j+1,k+\frac{1}{2})} - H_z|^{n}_{(i,j,k+\frac{1}{2})}}{\Delta y} - \frac{H_y|^{n}_{(i,j+\frac{1}{2},k+1)} - H_y|^{n}_{(i,j+\frac{1}{2},k)}}{\Delta z} \\ -\sigma_{(i,j+\frac{1}{2},k+\frac{1}{2})} \left(\frac{E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} + E_x|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{2} \right) \end{array} \right) \end{aligned} \tag{2.64}$$

En regroupant les termes relatifs à E_x à l'instant $n+\frac{1}{2}$, nous aurons finalement

$$\begin{aligned} E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = \left(\frac{1 - \frac{\sigma_{(i,j+\frac{1}{2},k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}{1 + \frac{\sigma_{(i,j+\frac{1}{2},k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}}} \right) E_x|^{n-\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})} \\ + \left(\frac{\frac{\Delta t}{\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}}}{1 + \frac{\sigma_{(i,j+\frac{1}{2},k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i,j+\frac{1}{2},k+\frac{1}{2})}}} \right) \cdot \left(\begin{array}{c} \frac{H_z|^{n}_{(i,j+1,k+\frac{1}{2})} - H_z|^{n}_{(i,j,k+\frac{1}{2})}}{\Delta y} \\ - \frac{H_y|^{n}_{(i,j+\frac{1}{2},k+1)} - H_y|^{n}_{(i,j+\frac{1}{2},k)}}{\Delta z} \end{array} \right) \end{aligned} \tag{2.65}$$

De la même manière, les expressions des différences finies basées sur l'algorithme de Yee pour les composantes E_y et E_z du champ électrique données par les équations (2.56) et

(2.57) peuvent être obtenues (Équations 2.66 et 2.67).

$$E_y\big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})} = \left(\frac{1-\frac{\sigma_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}{1+\frac{\sigma_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}\right) E_y\big|^{n-\frac{1}{2}}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})} + \left(\frac{\frac{\Delta t}{\varepsilon_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}{1+\frac{\sigma_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}}\right) \cdot \left(\begin{array}{c} \frac{H_x|^n_{(i-\frac{1}{2},j+1,k+1)} - H_x|^n_{(i-\frac{1}{2},j+1,k)}}{\Delta z} \\ -\frac{H_z|^n_{(i,j+1,k+\frac{1}{2})} - H_z|^n_{(i-1,j+1,k+\frac{1}{2})}}{\Delta x} \end{array}\right) \tag{2.66}$$

$$E_z\big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)} = \left(\frac{1-\frac{\sigma_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}{1+\frac{\sigma_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}\right) E_z\big|^{n-\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)} + \left(\frac{\frac{\Delta t}{\varepsilon_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}{1+\frac{\sigma_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}\Delta t}{2\varepsilon_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}}\right) \cdot \left(\begin{array}{c} \frac{H_y|^n_{(i,j+\frac{1}{2},k+1)} - H_y|^n_{(i-1,j+\frac{1}{2},k+1)}}{\Delta x} \\ -\frac{H_x|^n_{(i-\frac{1}{2},j+1,k+1)} - H_x|^n_{(i-\frac{1}{2},j,k+1)}}{\Delta y} \end{array}\right) \tag{2.67}$$

Par analogie, on peut obtenir les équations aux dérivées partielles pour les équations (2.52)-(2.54) des composantes du champs magnétiques H_x, H_y et H_z. Dans le cas où on considère un terme relatif aux pertes magnétiques noté $\sigma^* H$ avec σ^* qui représente le coefficient de pertes magnétiques équivalentes analogue aux pertes électriques, les équations (2.52)-(2.54) s'écriront [56] :

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} - \sigma^* H_x\right) \tag{2.68}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} - \sigma^* H_y\right) \tag{2.69}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial x} - \frac{\partial E_y}{\partial x} - \sigma^* H_z\right) \tag{2.70}$$

La présence du terme $\sigma^* H$ à droite des équations implique l'utilisation de l'approximation de la valeur de H par la même formule semi-implicite analogue à celle de l'équation (2.63). cela se traduit par trois équations ayant des formes similaires à ceux calculées pour E.

En se référant à la figure (2.1), on aura la formule de calcul itérative en fonction du temps

pour la composante H_x :

$$H_x|^{n+1}_{(i-\frac{1}{2},j+1,k+1)} = \left(\frac{1 - \frac{\sigma^*_{(i-\frac{1}{2},j+1,k+1)} \Delta t}{2\mu_{(i-\frac{1}{2},j+1,k+1)}}}{1 + \frac{\sigma^*_{(i-\frac{1}{2},j+1,k+1)} \Delta t}{2\mu_{(i-\frac{1}{2},j+1,k+1)}}} \right) H_x|^{n}_{(i-\frac{1}{2},j+1,k+1)} + \left(\frac{\frac{\Delta t}{\mu_{(i-\frac{1}{2},j+1,k+1)}}}{1 + \frac{\sigma^*_{(i-\frac{1}{2},j+1,k+1)} \Delta t}{2\mu_{(i-\frac{1}{2},j+1,k+1)}}} \right) \cdot \begin{pmatrix} \frac{E_y|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+1,k+\frac{3}{2})} - E_y|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}{\Delta z} \\ - \frac{E_z|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{3}{2},k+1)} - E_z|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\Delta y} \end{pmatrix} \tag{2.71}$$

De la même manière, l'équation (2.72) donne l'expression en fonction du temps de la composante H_y.

$$H_y|^{n+1}_{(i,j+\frac{1}{2},k+1)} = \left(\frac{1 - \frac{\sigma^*_{(i,j+\frac{1}{2},k+1)} \Delta t}{2\mu_{(i,j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^*_{(i,j+\frac{1}{2},k+1)} \Delta t}{2\mu_{(i,j+\frac{1}{2},k+1)}}} \right) H_y|^{n}_{((i,j+\frac{1}{2},k+1))} + \left(\frac{\frac{\Delta t}{\mu_{(i,j+\frac{1}{2},k+1)}}}{1 + \frac{\sigma^*_{(i,j+\frac{1}{2},k+1)} \Delta t}{2\mu_{(i,j+\frac{1}{2},k+1)}}} \right) \cdot \begin{pmatrix} \frac{E_z|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2},k+1)} - E_z|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2},k+1)}}{\Delta x} \\ - \frac{E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{3}{2})} - E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{\Delta z} \end{pmatrix} \tag{2.72}$$

Finalement, l'équation (2.73) donne l'expression en fonction du temps de la composante H_z.

$$H_z|^{n+1}_{(i,j+1,k+\frac{1}{2})} = \left(\frac{1 - \frac{\sigma^*_{(i,j+1,k+\frac{1}{2})} \Delta t}{2\mu_{(i,j+1,k+\frac{1}{2})}}}{1 + \frac{\sigma^*_{((i,j+1,k+\frac{1}{2})} \Delta t}{2\mu_{(i,j+1,k+\frac{1}{2})}}} \right) H_z|^{n}_{(i,j+1,k+\frac{1}{2})} + \left(\frac{\frac{\Delta t}{\mu_{(i,j+1,k+\frac{1}{2})}}}{1 + \frac{\sigma^*_{(i,j+1,k+\frac{1}{2})} \Delta t}{2\mu_{(i,j+1,k+\frac{1}{2})}}} \right) \cdot \begin{pmatrix} \frac{E_x|^{n+\frac{1}{2}}_{(i,j+\frac{3}{2},k+\frac{1}{2})} - E_x|^{n+\frac{1}{2}}_{(i,j+\frac{1}{2},k+\frac{1}{2})}}{\Delta y} \\ - \frac{E_y|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+1,k+\frac{1}{2})} - E_y|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+1,k+\frac{1}{2})}}{\Delta x} \end{pmatrix} \tag{2.73}$$

Avec les systèmes des expressions de différences finies de (2.65-2.67) et (2.71-2.73), la nouvelle valeur d'une composante du vecteur de champ électromagnétique à tout point du treillis ne dépend que de sa valeur précédente, les valeurs précédentes des composantes de vecteur de l'autre champs en des points adjacents, et les sources de courant électrique et magnétique connues (s'ils existent). Par conséquent, à n'importe quel pas de temps donné, le calcul d'un vecteur de champ peut se faire soit séquentiellement, c'est à dire, point à la

fois, soit, en parallèle si p processeurs sont utilisés simultanément, c'est à dire p points à la fois [52, 56].

2.8.4 Précision et stabilité

Pour assurer l'exactitude des résultats calculés, l'incrément spatial doit être petit par rapport à la longueur d'onde (généralement $\leq \lambda/10$). Cela revient à avoir 10 cellules ou plus par longueur d'onde. Pour assurer la stabilité du schéma de différences finies des équations (2.65)-(2.73), l'incrément de temps Δt doit satisfaire la condition de stabilité suivante connu comme la condition de Courant-Friedrichs-Lewy (CFL) [11, 56, 57] :

$$u_{max}\Delta t \leq \frac{1}{\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}}} \tag{2.74}$$

où u_{max} est la vitesse maximale de phase de l'onde dans le modèle. Dans le cas où on utilise une cellule cubique avec $\delta = \Delta x = \Delta y = \Delta z$, l'équation (2.74) devient :

$$\Delta t \leq \frac{\delta}{u_{max}\sqrt{3}} \tag{2.75}$$

Nous pouvons conclure qu'il y a une limite supérieur pour le choix de l'incrément temporelle dans l'algorithme de Yee après que les incréments spatiaux aient été choisis. Cette condition peut conduire à des incréments temporels inutilement petits pour certains problèmes.

2.8.5 Les conditions absorbante - PML

Une problématique fondamentale qui doit être traitée avec l'approche FDTD est que de nombreuses géométries à étudier sont définies dans des régions dite "ouvertes" où le domaine spatial du champ calculé est illimité dans une ou plusieurs directions de coordonnées. Il est clair qu'aucun ordinateur ne peut stocker une quantité illimitée de données, et par conséquent, le domaine de calcul de champ doit être limité en taille. Le domaine de calcul doit être suffisamment grand pour englober la structure étudiée, et une condition aux limites appropriée sur le périmètre extérieur du domaine doit être utilisée pour simuler son extension à l'infini [56].

Dans le processus, la condition aux limites extérieures doit supprimer les réflexions parasites des ondes numériques sortantes à un niveau acceptable, permettant à la solution

FDTD de rester valable pour toutes les étapes temporelles, surtout après que les ondes numériques réfléchies reviennent au voisinage de la structure modélisée . Selon leur base théorique, les conditions aux limites extérieures de ce type ont été appelées soit conditions aux limites de rayonnement (Radiation Boundary Conditions (RBC) ou conditions aux limites absorbantes (Absorbing Boundary Conditions ABC). C'est cette dernière qui sera utilisée pour décrire toutes les conditions aux limites extérieures utilisées pour simuler l'extension d'un domaine de calcul FDTD à l'infini [56]. La précision de l'ABC conditionne celle de la méthode FDTD. Ainsi, le besoin des couches absorbantes précises a donné lieu à différents types d'ABC's [56, Chapter 6]. Dans cette section, nous allons considérer seulement l'ABC de type PML de Bérenger [54, 56, 58, 59] étant donnée que c'est une méthode largement acceptée dans la communauté du CEM.

Dans la technique de troncature PML, une couche artificielle de matériau absorbant est placée autour de la limite extérieure du domaine de calcul. Le but est de s'assurer qu'une onde plane incidente depuis l'espace libre FDTD vers la région PML avec un angle arbitraire est complètement absorbée dans la couche sans réflexion. Cela revient à dire qu'il y a une transmission complète de l'onde plane incidente à l'interface entre l'espace libre et la région PML (Figure 2.2). Ainsi, la région FDTD et la région PML sont parfaitement adaptées [11]. Pour illustrer la technique PML, considérons les équations de Maxwell en deux dimensions

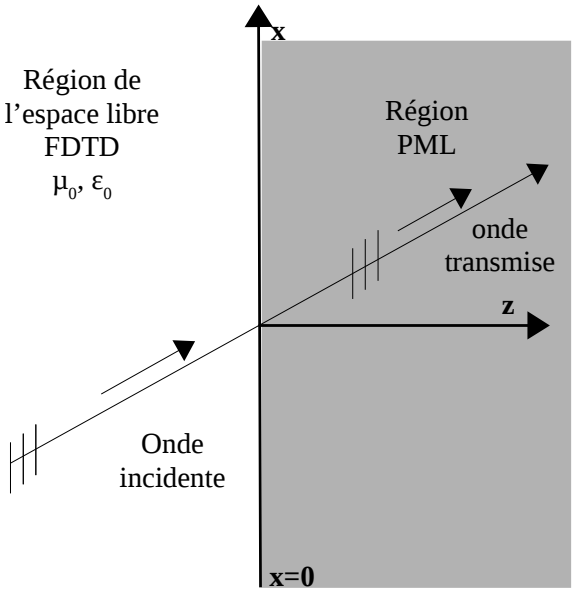


FIGURE 2.2 – Transmission sans réflexion d'une onde plane à une interface PML/espace libre.

(2D) pour le mode Transverse électrique (TE) avec les composantes E_x, E_y et H_z et aucune variation le long de l'axe z. Considérons les équations (2.54 - 2.56) et mettant $E_z = 0 = \frac{\partial}{\partial z}$,

nous aurons :

$$\varepsilon_0 \frac{\partial E_x}{\partial t} + \sigma E_x = \frac{\partial H_z}{\partial y} \tag{2.76}$$

$$\varepsilon_0 \frac{\partial E_y}{\partial t} + \sigma E_y = -\frac{\partial H_z}{\partial x} \tag{2.77}$$

$$\mu_0 \frac{\partial H_z}{\partial t} + \sigma^* H_z = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \tag{2.78}$$

où la PML, en tant que milieu avec perte, est caractérisée par une conductivité électrique σ et une conductivité magnétique σ^*. Les conductivités sont liées par la relation suivante :

$$\frac{\sigma}{\varepsilon_0} = \frac{\sigma^*}{\mu_0} \tag{2.79}$$

Cette relation assure un niveau d'atténuation nécessaire et force l'impédance de la PML à être égale à celle de l'espace libre. Ainsi, une transmission sans réflexion d'une d'onde plane à travers l'interface est faisable. Pour les angles d'incidence obliques, la conductivité de la PML doit présenter une certaine caractéristique d'anisotropie pour assurer une transmission sans réflexion. Pour ce faire, la composante H_z doit être scindée en deux sous-composants, H_{zx} et H_{zy}, avec la possibilité d'attribuer des pertes aux composants individuels du champ divisé. cette technique est la pierre angulaire de la technique PML. Le résultat est la présence de quatre composantes E_x, E_y, H_{zx} et H_{zy} et quatre équations de champ couplées (plutôt que les trois habituelles) [11, 56].

$$\varepsilon_0 \frac{\partial E_x}{\partial t} + \sigma_y E_x = \frac{\partial (H_{zx} + H_{zy})}{\partial y} \tag{2.80}$$

$$\varepsilon_0 \frac{\partial E_y}{\partial t} + \sigma_x E_y = -\frac{\partial (H_{zx} + H_{zy})}{\partial x} \tag{2.81}$$

$$\mu_0 \frac{\partial H_{zx}}{\partial t} + \sigma_x^* H_{zx} = -\frac{\partial E_y}{\partial x} \tag{2.82}$$

$$\mu_0 \frac{\partial H_{zy}}{\partial t} + \sigma_y^* H_{zy} = \frac{\partial E_x}{\partial y} \tag{2.83}$$

Les équations (2.80)-(2.83) peuvent être discrétisées pour donner les équations aux différences en fonction du temps Δt de la région PML. Le calcul standard de Yee ne peut pas être utilisé ici du fait de la rapide atténuation de l'onde incidente causée par la couche PML. En contrepartie, On utilise des équations aux dérivées exponentielles pour exclure toute possibilité d'instabilité de diffusion. Dans les notations FDTD habituelles, les quatre équations

temporelles résultantes pour la région PML sont [11, 54] :

$$E_x\big|^{n+1}_{(i+\frac{1}{2},j)} = e^{-\sigma_y(j)\frac{\Delta t}{\varepsilon_0}} E_x\big|^{n}_{(i+\frac{1}{2},j)} + \frac{1-e^{-\sigma_y(j)\frac{\Delta t}{\varepsilon_0}}}{\sigma_y(j)\delta} \cdot \left[H_{zx}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} + H_{zy}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} - H_{zx}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j-\frac{1}{2})} - H_{zy}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j-\frac{1}{2})} \right] \quad (2.84)$$

$$E_y\big|^{n+1}_{(i,j+\frac{1}{2})} = e^{-\sigma_x(i)\frac{\Delta t}{\varepsilon_0}} E_y\big|^{n}_{(i,j+\frac{1}{2})} + \frac{1-e^{-\sigma_x(i)\frac{\Delta t}{\varepsilon_0}}}{\sigma_x(i)\delta} \cdot \left[H_{zx}\big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2})} + H_{zy}\big|^{n+\frac{1}{2}}_{(i-\frac{1}{2},j+\frac{1}{2})} - H_{zx}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} - H_{zy}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} \right] \quad (2.85)$$

$$H_{zx}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} = e^{-\sigma_x^*(i+\frac{1}{2})\frac{\Delta t}{\mu_0}} H_{zx}\big|^{n-\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} + \frac{1-e^{-\sigma_x^*(i+\frac{1}{2})\frac{\Delta t}{\mu_0}}}{\sigma_x^*(i+\frac{1}{2})\delta} \cdot \left[E_y\big|^{n}_{(i,j+\frac{1}{2})} - E_y\big|^{n}_{(i+1,j+\frac{1}{2})} \right] \quad (2.86)$$

$$H_{zy}\big|^{n+\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} = e^{-\sigma_y^*(i+\frac{1}{2})\frac{\Delta t}{\mu_0}} H_{zy}\big|^{n-\frac{1}{2}}_{(i+\frac{1}{2},j+\frac{1}{2})} + \frac{1-e^{-\sigma_y^*(i+\frac{1}{2})\frac{\Delta t}{\mu_0}}}{\sigma_y^*(i+\frac{1}{2})\delta} \cdot \left[E_x\big|^{n}_{(i+\frac{1}{2},j+1)} - E_x\big|^{n}_{(i+\frac{1}{2},j)} \right] \quad (2.87)$$

Ces équations peuvent être directement implémentées dans une simulation FDTD pour modéliser un milieu PML. Il reste le choix de la profondeur de la couche PML et sa conductivité. En théorie, la PML aurait δ de profondeur et avoir une conductivité quasi-infinie. Il a été montré, cependant, que l'augmentation graduelle de la conductivité en fonction de la profondeur minimise les réflexions ; ç'est à dire la «superposition» du médium et la dépendance de σ en fonction de i et j.

2.9 La méthode des éléments finis (FEM)

La méthode des éléments finis est une technique de calcul permettant d'obtenir des solutions approximatives aux équations aux dérivées partielles. Plutôt que d'approcher directement l'équation aux dérivées partielles comme dans le cas des méthodes à différences finies, la méthode des éléments finis utilise un problème variationnel qui implique une intégrale

de l'équation différentielle sur le domaine du problème. Ce domaine est divisé en un certain nombre de sous-domaines appelés éléments finis et la solution de l'équation différentielle partielle est approchée par une fonction polynomiale plus simple sur chaque élément. Ces polynômes doivent être assemblés pour que la solution approximative ait un degré de régularité approprié sur tout le domaine. Une fois cela fait, l'intégrale variationnelle est évaluée comme la somme des contributions de chaque élément fini. Le résultat est un système algébrique pour la solution approximative ayant une taille finie plutôt que l'équation différentielle partielle en dimension infinie originale. Ainsi, comme les méthodes de différences finies, le processus des éléments finis réalise une discrétisation de l'équation aux dérivées partielles, mais contrairement aux méthodes de différences finies, la solution approximative est connue dans le domaine comme une fonction polynomiale par morceaux et pas seulement sur un ensemble de points.

2.9.1 Les étapes élémentaires de l'analyse par FEM

L'analyse par éléments finis de tout problème implique essentiellement quatre étapes [11, 49]

- Discrétiser la région de la solution en un nombre fini de sous-régions ou d'éléments,
- Déduire des équations pour un élément typique (choix des fonctions d'interpolation),
- Assembler tous les éléments dans la région de solution,
- Résoudre le système d'équations obtenues.

2.9.1.1 Discrétisation du domaine

La discrétisation du domaine, noté Ω, est la première étape, et peut-être la plus importante, de toute analyse par éléments finis car la manière dont le domaine est discrétisé affectera les besoins en terme de stockage, le temps de calcul, et la précision des résultats numériques. Dans cette étape, le domaine entier Ω est subdivisé en un certain nombre de petits sous-domaines, notés $\Omega^e (e = 1, 2, \ldots, M)$, avec M est le nombre total de sous-domaines. Ces sous-domaines sont généralement appelés éléments. Pour un domaine unidimensionnel (ligne droite ou courbe), les éléments sont souvent des petits segments de ligne interconnectés pour former (au moins approximativement) la ligne d'origine (Figure 2.3a). Pour un domaine bidimensionnel, les éléments sont généralement de petits triangles et rectangles (Figures 2.3b,2.3c). Les éléments rectangulaires sont, évidement, les mieux adaptés

pour discrétiser des régions rectangulaires, tandis que les triangulaires peuvent être utilisés pour des régions irrégulières. Dans une solution tridimensionnelle, le domaine peut être subdivisé en tétraèdres, prismes triangulaires ou en briques rectangulaires (Figures 2.3d,2.3e,2.3f). Parmi ceux-ci, les tétraèdres sont les plus simples et les mieux adaptés aux domaines à volume arbitraire [11, 49].

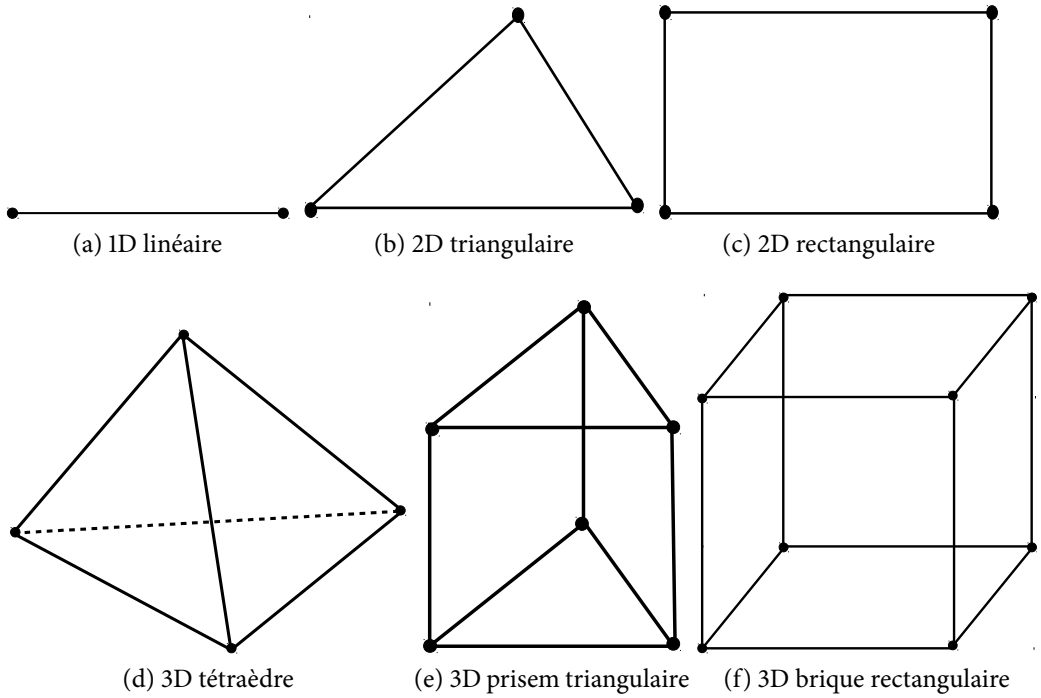


FIGURE 2.3 – Éléments finis de base

Dans la plupart des solutions d'éléments finis, le problème est formulé en termes de fonction inconnue ϕ aux nœuds associés à l'élément. Par exemple, un élément ligne linéaire a deux nœuds, un à chaque extrémité. Un élément triangulaire linéaire a trois nœuds situés à ses trois sommets, alors qu'un tétraèdre linéaire possède quatre nœuds situés aux quatre coins. L'implémentation de la FEM nécessite la description de chaque nœud. Une description complète d'un nœud comprend les valeurs des coordonnées, le numéro local et le numéro global associé. Le numéro local d'un nœud indique sa position dans l'élément, alors que le numéro global spécifie sa position dans l'ensemble du système. Bien que la spécification des valeurs de coordonnées soit un travail plutôt simple, la numérotation des nœuds et des éléments nécessite une certaine stratégie [49].

La discrétisation du domaine est considérée comme une tâche de prétraitement car elle peut être séparée des autres étapes.

2.9.1.2 Choix des fonctions d'interpolation

La deuxième étape d'une analyse par éléments finis est la sélection d'une fonction d'interpolation qui fournit une approximation de la solution inconnue dans un élément. L'interpolation est généralement sélectionnée pour être un polynôme du premier ordre (linéaire), deuxième ordre (quadratique) ou d'un ordre supérieur. Les polynômes d'ordre supérieur, bien que très précis, aboutissent généralement à une formulation plus compliquée que les polynômes d'ordre inférieur. Par conséquent, l'interpolation linéaire simple est encore largement utilisée. Une fois l'ordre du polynôme sélectionné, nous pouvons dériver une expression pour la solution inconnue dans un élément e sous la forme suivante [49] :

$$\hat{\phi}^e = \sum_{j=1}^{n} N_j^e \phi_j^e = \{N^e\}^T \{\phi^e\} = \{\phi^e\}^T \{N^e\} \tag{2.88}$$

où n est le nombre de noeuds de l'élément, ϕ_j^e est la valeur de ϕ au noeud j de l'élément et N_j^e est la fonction d'interpolation du nœud j qui est également appelée fonction de base ou fonction de forme. Une propriété importante de la fonction de base N_j^e est qu'elle est non nulle seulement à l'intérieur de l'élément considéré.

2.9.2 Formulation du système d'équations

La troisième étape dans l'analyse par éléments finis consiste à formuler le système d'équations. Les deux méthodes celle variationnel de Ritz et celle de Galerkin peuvent être utilisées dans cet but.

2.9.2.1 Formulation par la méthode de Ritz

Considérons le problème aux limites défini par l'équation (2.24). Le fonctionnel F donné par l'équation (2.28) peut être exprimé par :

$$F(\tilde{\phi}) = \sum_{e=1}^{M} F^e(\tilde{\phi}^e) \tag{2.89}$$

avec M est le nombre des éléments dans le domaine globale et

$$F^e(\tilde{\phi}^e) = \frac{1}{2} \int_{\Omega^e} \tilde{\phi}^e L \tilde{\phi}^e d\Omega - \int_{\Omega^e} f \tilde{\phi}^e d\Omega \tag{2.90}$$

Remplaçons (2.88) dans (2.90), nous obtenons

$$F^e = \frac{1}{2}\{\phi^e\}^T \int_{\Omega^e} \{N^e\} L \{N^e\}^T d\Omega \{\phi^e\} - \{\phi^e\}^T \int_{\Omega^e} f\{N^e\} d\Omega \tag{2.91}$$

ou sous format matricielle

$$F^e = \frac{1}{2}\{\phi^e\}^T [K^e] \{\phi^e\} - \{\phi^e\}^T \{b^e\} \tag{2.92}$$

$[K^e]$ est une matrice $n \times n$ et $\{b^e\}$ un vecteur colonne $n \times 1$ avec leurs éléments donnés par

$$K^e_{ij} = \int_{\Omega^e} N^e_i L N^e_j d\Omega \tag{2.93}$$

et

$$b^e_i = \int_{\Omega^e} f N^e_i d\Omega \tag{2.94}$$

En introduisant la formule du fonctionnel F^e de l'équation (2.92) dans l'équation (2.89) et en effectuant la sommation, nous arrivons à l'équation (2.95),

$$F = \frac{1}{2}\{\phi\}^T [K] \{\phi\} - \{\phi\}^T \{b\} \tag{2.95}$$

$[K]$ est une matrice $N \times N$ symétrique avec N est le nombre total des inconnus qui est égale au nombre total des nœuds du domaine, $\{\phi\}$ est un vecteur $N \times 1$ et dont les éléments sont les coefficients des fonctions de base à déterminer et $\{b\}$ est un vecteur $N \times 1$ connu.

Le système d'équations est enfin obtenu en mettant les dérivées partielles de F par rapport à ϕ_i égales à zéro :

$$[K]\{\phi\} = \{b\} \tag{2.96}$$

2.9.2.2 Formulation par la méthode de Galerkin

Le système d'équations peut être formulé par la méthode de Galerkin. Pour le problème de l'équation (2.24), le résidu pondéré du $e^{ème}$ élément est donnée par [49]

$$R^e_i = \int_{\Omega^e} N^e_i (L\tilde{\phi}^e - f) d\Omega \qquad i = 1, 2, 3, \cdots, n \tag{2.97}$$

En substituant (2.88) dans (2.97), On obtient

$$R^e_i = \int_{\Omega^e} N^e_i L \{N^e\}^T d\Omega \{\phi^e\} - \int_{\Omega^e} f N^e_i d\Omega \qquad i = 1, 2, 3, \cdots, n \tag{2.98}$$

qui sous forme matricielle, peut s'écrire comme

$$\{R^e\} = [K^e]\{\phi^e\} - \{b^e\} \tag{2.99}$$

$\{R^e\} = [R_1^e, R_2^e, \cdots, R_n^e]^T$, K_{ij}^e et b_i^e ont la même forme que ceux de la méthode de Ritz (2.93) et (2.94). Étant donnée la fonction de développement et donc celle de pondération associée à un nœud, fait intervenir tous les éléments directement connectés au nœud considéré, alors, le résidu pondéré R_i associé au nœud i est une sommation sur les éléments directement connectés au nœud i. Il est alors possible de développer l'équation (2.99) en se servant de la relation entre les indices locaux et globaux et en additionnant sur chaque élément pour obtenir [49]

$$\{R\} = \sum_{e=1}^{M} \{\bar{R}^e\} = \sum_{e=1}^{M} \left([\bar{K}^e]\{\bar{\phi}^e\} - \{\bar{b}^e\} \right) \tag{2.100}$$

Avec $\{R\} = [R_1, R_2, \cdots, R_N]^T$ et N est le nombre total des nœuds. Le système d'équations est obtenu en mettant le résidus $\{R\}$ égale à zéro et on obtient finalement un système avec la même forme que (2.96).

Avant de résoudre le système de l'équation (2.96) pour une solution donnée, nous devons appliquer les conditions aux limites requises.

2.9.3 Solution du système d'équations

La solution du système d'équations est l'étape finale de l'analyse par éléments finis. Le système résultant possède une des deux formes suivantes [49] :

$$[K]\{\phi\} = \{b\} \tag{2.101}$$

ou

$$[A]\{\phi\} = \lambda\{B\}\{\phi\} \tag{2.102}$$

L'équation (2.101) est de type déterministe résultant soit d'une équation aux différences non homogène ou des conditions aux limites non homogènes soit les deux. En électromagnétisme, les systèmes déterministes sont généralement associés aux problèmes de dispersion, radiation et d'autres problèmes où il existe une source ou une excitation. L'équation 2.102, par contre est de type valeur propre produite par des systèmes gouvernés par des équations aux différences homogènes et des conditions aux limites homogènes. En électromagné-

tisme, les systèmes à valeur propre sont généralement associés aux problèmes sans source tel que la propagation des ondes dans les guides d'ondes et les cavités à résonance [49].

2.9.4 Aspects calculatoires

Dans la formulation FEM, les matrices sont généralement de grande taille, mais elles sont creuses et symétriques. Par conséquent, une solution efficace de l'équation matricielle est très importante, car cet aspect domine généralement les besoins globaux en ressources de calcul. Les points importants à prendre en considération sont les schémas de stockage, les solveurs (directs ou itératifs) et les pré-conditionneurs utilisés (dans le cas itératif) [12].

Les matrices produites par la méthode des éléments finis sont creuses, avec seulement un très petit pourcentage d'éléments non nuls. En ne stockant que les éléments non nuls, le besoin en stockage est réduit de $\mathcal{O}(N^2)$ à $\mathcal{O}(N)$. Les techniques les plus répandues pour stocker les matrices creuses sont celles qui reposent sur un format de stockage basé soit sur une ligne compressée, soit sur une colonne compressé. Dans ces approches, les valeurs des éléments non nuls d'une matrice fragmentée sont stockées dans un vecteur à virgule flottante. Un vecteur entier est utilisé pour stocker les index de ligne ou de colonne des éléments non nuls, et un autre vecteur entier est introduit pour stocker l'emplacement du premier élément non nul de chaque ligne dans le vecteur compressé. Pour une matrice symétrique, seules les éléments non nuls du triangle supérieur ou inférieur (y compris ceux de la diagonale) doivent être stockés.

Le choix d'un solveur matriciel peut avoir un impact significatif sur l'efficacité de calcul, et il est donc important de choisir un solveur qui peut mieux exploiter les propriétés de la matrice d'éléments finis. Il existe deux types de solveurs matriciels. Le premier, connu sous le nom de solveur direct, est basé sur l'élimination de Gauss ou la décomposition LU. Ces solveurs sont couramment utilisés pour les matrices denses, bien qu'ils soient également applicables aux matrices creuses stockées dans un format bande, ou même un format entièrement creux dans le cas des méthodes frontales et multi-frontales [60, 61]. L'alternative aux solveurs directes est celle des solveurs itératifs, qui nécessitent beaucoup moins de mémoire car ils sont basés sur le calcul successif de produits matrice-vecteur selon un algorithme itératif destiné à converger vers la solution [62]. Le principal inconvénient des techniques itératives est qu'elles peuvent nécessiter un grand nombre d'itérations pour converger, principalement en raison des emplacements des valeurs propres de la matrice dans le plan complexe. Cependant, si les valeurs propres sont toutes regroupées autour de (1,0), la convergence est généralement rapide. Pour améliorer la convergence d'un solveur ité-

ratif, un préconditionneur est habituellement utilisé pour rapprocher les valeurs propres de (1,0), réduisant ainsi le nombre d'itérations. Un préconditionneur peut être construit selon une bonne compréhension de la nature physique du problème ou de la structure de la matrice d'origine [12].

Il existe beaucoup de solveurs directs [63, 64] et itératifs très robustes est efficaces qui traitent des matrices creuses. Par exemple, MKL pour Intel Math Kernel Library [65] et WSMP pour IBM Watson Sparse Matrix Package [66, 67] possèdent des solveurs directs pour les matrices symétriques et asymétriques pouvant être utilisés comme des programmes séquentiel et sur des systèmes à mémoire partagée ou à mémoire distribuée. Pour les matrices asymétriques, UMFPACK pour Unsemetric MultiFrontal Package intégrée dans la suite SuiteSparse [63, 68–70] , MUMPS pour a parallel sparse direct solver [71] et SuperLU pour Supernodal LU [72, 73] fournissent une solution parallèle évolutive sur les systèmes de calcul à mémoire distribuée et éventuellement à mémoire partagée ou parfois sur des systèmes de calcul à base de processeurs graphiques. On peut aussi citer PaSTiX pour Parallel Sparse matriX package [74–76], une bibliothèque scientifique qui fournit un solveur parallèle haute performance pour de très grands systèmes linéaires creux.

pour les solveurs itératifs, PETSc pour Portable, Extensible Toolkit for Scientific Computation [77–79] et SPARSKIT [80] fournissent une variété d'algorithmes de sous-espace de Krylov, tels que ceux basés sur les méthodes stabilisées du gradient bi-conjugué ou en anglais biconjugate gradient stabilized method (BiCGStab) et les méthodes du résidu minimal généralisé ou en anglais generalized minimal residual (GMRES), et une variété de préconditionneurs, tels que les préconditionneurs LU incomplets et les préconditionneurs à relaxation successive symétrique ou en anglais symmetric successive over-relaxation (SSOR) pour accélérer la convergence itérative [3]. Par ailleurs, Il est impossible de lister tous les outils utilisés dans le domaine qu'ils soient open source ou propriétaires.

2.10 Calcul parallèle en électromagnétisme - biographie et état de l'art

Pour une simulation électromagnétique associée à une configuration à grande échelle, Le nombre requis de points de maillage pour répondre à une précision donnée est souvent hors de portée d'un système informatique classique. L'efficacité numérique de la CEM peut maintenant être considérablement améliorée en utilisant des systèmes multiprocesseurs évolutifs [81].

Le calcul parallèle sur des clusters et/ou des ordinateurs avec des processeurs multicœurs reste la méthode de choix pour répondre aux défis techniques et scientifiques modernes qui découlent des applications extrêmement complexes de la vie réelle [3]. Le sujet du calcul parallèle en CEM est très large pour être couvert ici, nous allons donc donner seulement une vue synoptique.

La méthode MoM est la technique la plus connue parmi les méthodes basées sur l'équation intégrale (IE). En raison des nombreux avantages de la méthode, l'implémentation parallèle de la méthode elle-même, des méthodes hybrides et les algorithmes rapides liés à la MoM, ont tous fait l'objet de recherches importantes au cours des dernières décennies et certainement les années à venir [3].

L'algorithme de la MoM a été parallélisé au cours des années 1990 [82–84]. Patterson *et al.* [82] a implémenté et exécuté le code électromagnétique ou en anglais numerical electromagnetics code (NEC) dans un environnement parallèle développé au laboratoire national de Lawrence Livermore aux USA en 1990. En 1991, Cwik *et al.* [83] ont utilisés la MoM pour résoudre des problèmes de dispersion en utilisant la programmation parallèle. Plus tard, et en 1994, Cwik *et al.* [85] en collaboration avec l'auteur de PLAPACK [86] a développé une implémentation parallèle de la méthode des moments utilisant les fonctions de base RWG [87] en. En 1998, une implémentation parallèle de la MoM basée sur les fonctions de base RWG utilisant la bibliothèque ScaLAPACK [88] pour la résolution du système linéaire résultant a été portée sur un super calculateur Cray T3E [89].

Depuis le milieu des années 1990, des recherches ont été menées sur des implémentations parallèles de codes MoM répondants aux normes d'utilisation dans un milieu de production. L'une des implémentations parallèles fréquemment étudiées des codes existants largement utilisé est le NEC. Implémenté avec succès en 2003, NEC parallèle est portable sur n'importe quelle plate-forme supportant un environnements parallèles de passage de messages tels MPI [33] et la PVM [90]. Le code pourrait même être exécuté sur des clusters hétérogènes tournant sur des systèmes d'exploitation différents [3].

Le grand problème ou le goulot d'étranglement des ces implémentations parallèles est dû partiellement de la mémoire de travail requise. Une des solutions envisagée pour palier à ce problème est la parallélisation des méthodes hybrides comme la MoM-UTD pour uniform geometrical theory of diffraction [91], MoM-PO pour la méthode de physique optique (physical optics, PO) [92].

Les algorithmes rapides peuvent être aussi utilisés pour réduire l'empreinte mémoire et le temps d'exécution globale. Parmi les algorithmes les plus utilisés, le gradient conju-

gué ou conjugate gradient-fast Fourier transform (CG-FFT) [93], la méthode de l'intégral adaptatif (Adaptive Integral Method, AIM) [94], la FMM [95] et la méthode de la FFT pré-corrigée (precorrected FFT) [96]. La FMM, indiscutablement la plus populaire parmi ces méthodes, a encore été améliorée grâce à l'algorithme MLFMA en réduisant encore le temps de calcul du produit matrice-vecteur [97, 98]. En dépit du fait qu'il soit difficile à implémenter, le MLFMA est devenu l'algorithme par défaut pour les problèmes de diffusion à grande échelle survenant en électromagnétisme [3]. Par conséquent, la parallélisation du MLFMA sur des architectures à mémoires distribuées (clusters) [99–106] et sur des machines à mémoire partagée [107] a été toujours un sujet de recherche très actif.

Une extension basée sur la FFT de la FMM conventionnelle connue sous le nom FMM-FFT a permis une diminution du temps de la multiplication matrice-vecteur de l'algorithme conventionnel, tout en conservant la propriété d'une bonne parallélisation de FMM à un seul niveau (puisqu'elle n'utilise pas la structure arborescente de MLFMA). Waltz *et al.* a démontrer que l'implémentation parallèle de l'algorithme FMM-FFT est assez attrayant (comparé à l'algorithme MLFMA) dans le contexte des machines à mémoire distribuée massivement parallèles [3, 108].

Cependant, l'utilisation de méthodes hybrides et d'algorithmes rapides sacrifie la précision pour s'accommoder à la résolution des problèmes à grande échelle. Une solution pour surmonter le problème de l'occupation mémoire de la MoM tout en conservant sa précision consiste à utiliser un solveur dite out-of-core utilisant la mémoire secondaire du disque pour le stockage.

En effet, puisque la matrice générée par la MoM est une matrice complètement dense, la méthode de décomposition LU utilisée pour résoudre l'équation matricielle nécessite un calcul intensif comparée au processus lecture/écriture des éléments de la matrice à partir du disque dur [109]. Par conséquent, il est logique d'introduire un solveur out-of-core pour s'attaquer à de tels systèmes. En outre, il est possible d'optimiser l'implémentation parallèle pour fonctionner sur un cluster et ainsi résoudre des problèmes à grande échelle en un temps moindre [3].

Les recherches ont continués au cours des dernières années pour améliorer l'implémentation parallèle de la méthode des moments et ces dérivés sur toutes les plateformes de calcul parallèles y compris les architectures à base de processeurs graphiques (GPU). Lezar et Davidson [110] ont implémentés la MoM sur une architecture GPU en utilisant CUDA pour l'analyse de la dispersion d'une plaque parfaitement conductrice ou en anglais perfect electric conductor (PEC). Chen *et al.* [111] a fait de même pour l'analyse d'un réseau d'antenne micro-ruban en utilisant la MoM. Dans les références [112–116], plusieurs implémentations

parallèles de la méthodes des moments pour différentes architectures sont rapportées.

Outre la méthode des moments, la méthode des éléments finis (FEM) est une autre technique dans le domaine fréquentiel largement utilisée et qui présente un moyen efficace d'analyse des problèmes électromagnétiques. L'avantage principal de la FEM est qu'elle permet la modélisation efficace des géométries très irrégulières ainsi que des milieux matériels perméables et non homogènes. Mackerle [117] a rapporté les travaux sur les implémentations parallèles de la FEM sur les super ordinateurs, les grappes de calcul et les stations de travail dans la période 1985-1995. Il a été rapporté qu'une implémentation sur des machines massivement parallèles de l'algorithme séquentiel classique ne présente pas nécessairement une adaptation à l'architecture multiprocesseurs et peut conduire à une très mauvaise utilisation des ressources de la plateforme matérielle [118]. Dans la référence [118], l'auteur présente une méthode de décomposition de domaine basée sur l'algorithme FETI pour (Finit Element Tearing and Interconnect) pour la résolution des grandes matrices creuses associées à la solution issue de la FEM de l'équation d'onde vectorielle. L'algorithme FETI est basé sur la méthode des multiplicateurs de Lagrange et conduit à un système d'ordre réduit, qui est résolu en utilisant la méthode du gradient biconjugué (BiCGM). Il est montré que cette méthode est hautement évolutive et qu'elle est plus efficace sur les plateformes parallèles lors de la résolution de grandes matrices que les méthodes itératives traditionnelles telles qu'un algorithme de gradient conjugué pré-conditionné spécialement lorsque une PML est utilisé pour les conditions aux limites.

Plusieurs implémentations parallèles de la méthode des éléments finis (FEM), y compris les technique d'accélération en utilisant les GPU's [119–124], ont été utilisées avec succès pour traiter le couplage mutuel entre différentes structures.

Plus récemment, la capacité de la FEM utilisée pour résoudre des problèmes très complexes et multi-échelles a été considérablement améliorée en introduisant une nouvelle méthode de décomposition de domaine (Domain Decomposition Method, DDM) [125–128]. Celle-ci divise le problème original potentiellement gros et compliqué en un grand nombre de sous-domaines plus petits mais gérables, et les traitent individuellement. Au cours de son implémentation, des conditions de transmission appropriées sont appliquées entre les sous-domaines adjacents pour que le problème soit bien posé [129].

En fait, la caractéristique de la DDM la rend intrinsèquement parallélisable sur une plateforme de super ordinateurs [130]. En outre, la DDM fournit également une technique de préconditionnement efficace pour surmonter l'inconvénient de la FEM classique lors de la résolution d'un large système linéaire mal conditionné. Pour cela, des algorithmes parallèles basés sur différentes versions de DDM, incluant FETI et Schwarz, ont été proposés

ces dernières années [131, 132]. L'ancienne version de la DDM nécessite un système globalement interconnecté représentant la relation entre les arêtes des coins partagées par les sous-domaines. Par conséquent, même avec une convergence plus lente, le DDM de type Schwarz est plus puissante en prenant en compte les architectures à mémoire distribuée, où le coût de l'échange et de la synchronisation des données est élevé [133].

Dans la référence [133], l'auteur propose un schéma parallèle à deux niveaux basé sur une infrastructure avancée JAUMIN (pour J parallel adaptive unstructured mesh applications infrastructure) [134] et DDM pour simuler des problèmes d'E3 de grande taille. Cet outil développé en interne construit d'abord un planificateur de tâches de tel manière à ce que chaque processus soit attaché à un nœud de calcul, ensuite, ce processus va créer plusieurs processus (threads) qui sont mappés aux multiples cœurs du processeur. Ainsi, les simulations indépendantes des sous-domaines sont parallélisées de manière à exploiter l'hiérarchie de l'architecture matérielle et fournir une distribution de charge maximale.

Les recherches pour l'implémentation parallèle des méthodes numériques dans le domaine temporel ont présentés un intérêt certain depuis de nombreuses années. Parmi les méthodes les plus populaires, on peut citer la FDTD. La FDTD est par nature essentiellement basée sur une parallélisation de type données. Pala *et al.* [135] ont été les premiers à proposer une implémentation parallèle de la FDTD sur des machines massivement parallèles de type CM-1 et CM-2. La parallélisation de la méthode FDTD sur une architecture à mémoire distribuée a été réalisée en 1994 par Varadarajan et Mittra [136] qui ont suggéré des règles et des tolérances pour la mise en œuvre sur un cluster. Ils ont utilisé le protocole de transmission de messages de la machine virtuelle parallèle ou en anglais parallel virtual machine (PVM) et TCP/IP sur un cluster de huit ordinateurs. Les travaux se sont succédé pour donner des implémentations sur différentes architectures. Ainsi, Liu *et al.* [137] ont implémenter FDTD parallèle sur un supercalculateur de type CM-5 [137]. Tinniswood *et al.* [138] a utilisé une grappe de calcul de plus de 128 nœuds pour exécuter une implémentation parallèle de la FDTD. Les mesures de performances reportés dans ces travaux varient significativement et de ce fait ne représentent pas une mesure fiable de la qualité de l'algorithme parallèle à cause des limitations de la technologie et des systèmes utilisés [3].

Avec le développement de la technologie de passage de message MPI, plusieurs implémentations parallèles utilisant MPI ont vu le jours. Guiffaut et Mahdjoubi [139] ont utilisés une topologie en 2D des processus MPI pour implémenter la FDTD en parallèle. Pour minimiser le surcoût lié aux communications inter-processus, il ont utilisés le regroupement des zones de données non contigus qui est une fonctionnalité de MPI. Pour les régions aux limites (PML), l'auteur a utilisé une approche basée sur l'algorithme GUEHPMLs (Ge-

neralized un-split E-H PMLs) [140] qui permet une meilleur distribution de charge entre les processeurs et augmente ainsi l'accélération globale. Andersson [141] a utilisé une topologie cartésienne en 3D pour ces recherches. Zhang *et al.* [142] a fait de même dans ces recherches. Par la suite, une série de travaux traitant la topologie MPI en 3D optimale et différents modèles de communication MPI ont été étudiés. En 2005, Zhang *et al.* [143] ont étudiés l'influence des schémas de la topologie virtuelle MPI sur les performances du code parallèle de la FDTD conforme. En conclusion, ils présentent des règles générales pour obtenir la meilleur efficacité du code FDTD parallèle en optimisant la topologie virtuelle MPI. Les travaux sur l'implémentation parallèle de la FDTD sont toujours d'actualité [144–147] et il est pratiquement impossible de donner une biographie complète des travaux dans le domaine du calcul haute performance en électromagnétisme.

2.11 Conclusion

Dans ce chapitre, nous avons présenté un aperçu des méthodes numériques dans le domaine du calcul en électromagnétisme. Nous nous sommes intéressés aux trois méthodes les plus populaires et les plus utilisées dans le domaine, à savoir la méthode MoM, la FEM et la FDTD. Une biographie relativement riche sur les travaux de parallélisation de ces techniques et dérivées sur les différentes architectures parallèles est donnée. Nous avons aussi discuter de quelques aspects calculatoires rencontrés lors de la solution des systèmes issus de ces méthodes qui présentent des matrices denses (MoM et FDTD) et parfois creuses (FEM). plusieurs techniques de solutions sont évoquées ainsi que plusieurs bibliothèques de calcul très utilisés dans la solution de ces systèmes sont reportées.

Chapitre 3

Analyse de l'antenne circulaire multicouches et multiconducteurs alimentée par un câble coaxial

Sommaire

3.1 Introduction 62
3.2 Formulation mathématique 63
3.3 Résultats et discussions 66
3.4 Implémentation parallèle 67
3.5 Étude des performances réseaux 71
3.6 Conclusion 73

3.1 Introduction

Nous allons discuter dans ce chapitre du processus de parallélisation du calcul de l'analyse d'une antenne circulaire multicouche et multiconducteurs alimentée par un câble coaxiale.

L'inconvénient majeur des antennes micro-bandes est qu'elles rayonnent de façon efficace seulement dans une bande de fréquence très étroite, avec des largeurs de bande typiques de quelques centièmes [148]. Pour une antenne monocouche, augmenter l'épaisseur du substrat, afin d'augmenter la largeur de bande de l'impédance, produit des lobes secondaires de niveaux de cross-polarisation dus aux fortes énergies couplées aux ondes de surface. Les antennes microbandes, possédant des configurations à empilement, constituées d'un

ou de plusieurs plaques conductrices couplées parasitiquement à un « driven patch », surmontent l'inconvénient de la limitation de bande, et ceci par introduction des résonances dans la gamme de fréquences d'opération, atteignant des largeurs de bande de 10 – 20%. En outre, les antennes multicouches multiconducteurs ont permis des gains élevés et ont offert la possibilité d'obtenir une fréquence d'opération double (ou duale) [149].

3.2 Formulation mathématique

La géométrie de la structure multicouches considérée est illustrée en Figure 3.1. L'antenne

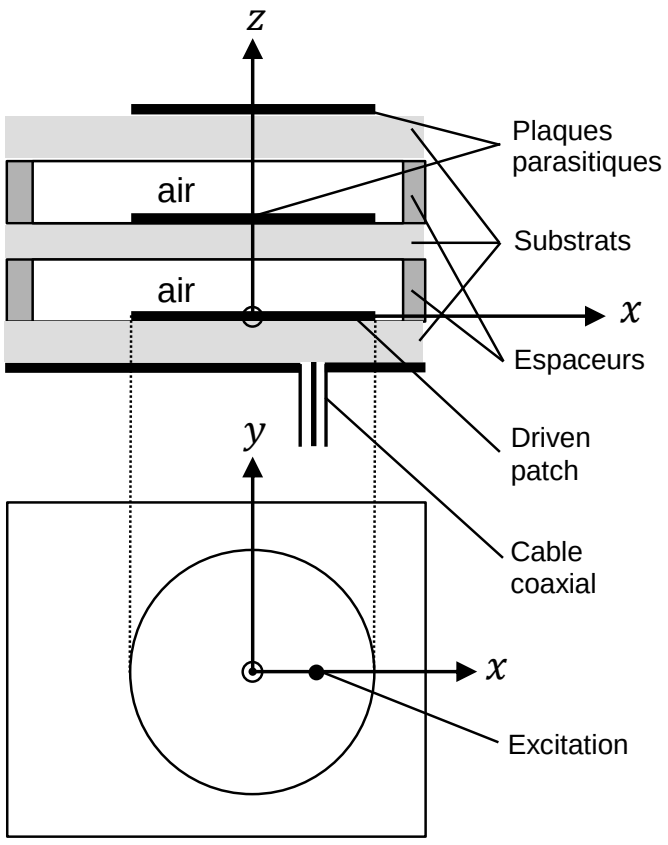


FIGURE 3.1 – Géométrie de l'antenne microbande circulaire à trois couches.

étudiée consiste en un empilement de deux plaques circulaires parasitiques et un driven patch, ayant toutes des rayons identiques $a = 1.65cm$, avec deux régions d'air. Le driven patch est déposé sur un substrat d'épaisseur $d_1 = 0.158cm$ et de permittivité $\varepsilon_{1r} = 2.33$ et

excitée en mode TM_{11} (de fréquence de résonance $f_{11} = 3.3072GHz$) par un câble coaxial de rayon $R = 0.5mm$ imprimée à une distance $\rho_0 = 1.33cm$ du centre du disque. La plaque parasitique 1 est imprimée sur un substrat d'épaisseur $d_3 = 0.0762cm$ avec $\varepsilon_{3r} = 2.45$. La plaque parasitique 2 est imprimée sur un substrat d'épaisseur $d_5 = 0.0508cm$, $\varepsilon_{5r} = 2.2$.

En utilisant la formulation de la fonction dyadique de Green en coordonnées cylindriques pour un milieu stratifié, nous obtenons des expressions pour les composantes transverses des champs électriques dus aux distributions des courants des disques et de la sonde. Les conditions aux frontières impliquent que les composantes transverses du champ électrique s'annulent sur les disques parfaitement conducteurs et les courants s'annulent au delà des disques. Nous obtenons finalement, les équations intégrales couplées suivantes (3.1, 3.2) [148] :

$$\begin{aligned}\mathbf{E}_i(\rho,\phi) = &\sum_{n=-\infty}^{\infty} e^{in\phi} \sum_{j=1}^{3} \int_0^{\infty} dk_\rho k_\rho \bar{\mathbf{H}}_n(k_\rho\rho)\cdot\bar{\mathbf{G}}_{ij}(k_\rho)\cdot\mathbf{K}_{jn}(k_\rho) \\ &+ \sum_{n=-\infty}^{\infty} e^{in\phi} \int_0^{\infty} dk_\rho k_\rho \bar{\mathbf{H}}_n(k_\rho\rho)\cdot\bar{\mathbf{G}}_{i1}(k_\rho)\cdot\mathbf{P}_n(k_\rho) = 0 \qquad \rho < a_i, i = 1,2,3 \end{aligned} \tag{3.1}$$

$$\boldsymbol{\kappa}_i(\rho,\phi) = \sum_{n=-\infty}^{\infty} e^{in\phi} \int_0^{\infty} dk_\rho k_\rho \bar{\mathbf{H}}_n(k_\rho\rho)\cdot\mathbf{K}_{in}(k_\rho) = 0 \qquad \rho > a_i, i = 1,2,3 \tag{3.2}$$

Avec :

(ρ,ϕ)	coordonnées polaires du point d'observation.
k_ρ	le nombre d'onde transverse.
$\mathbf{E}_i$	champ électrique sur le disque i.
$\bar{\mathbf{H}}_n(\cdot)$	noyau de la transformée de Hankel vectorielle.
$\bar{\mathbf{G}}_{ij}$	fonction spectrale dyadique de Green en représentation (TM,TE).
$\mathbf{K}_{in}$	$n^{ème}$ le mode de la densité du courant électrique surfacique sur le $i^{ème}$ disque.
$\boldsymbol{\kappa}_i$	la transformé de Hankel vectorielle de $\mathbf{K}_{in}$.

Les matrices $\bar{\mathbf{G}}_{i1}(k_\rho)$ comportent les effets du milieu stratifié lorsqu'on relie les courants de la sonde aux champs électriques transverses. Dans le dernier terme de (3.2), $\mathbf{P}_n(k_\rho)$ est associé au courant de la sonde et se donne par l'équation (3.3).

$$\mathbf{P}_n(k_\rho) = \begin{bmatrix} P_n(k_\rho) \\ 0 \end{bmatrix} = -\frac{I}{2\pi}\frac{k_\rho}{k_{1z}^2} J_n(k_\rho\rho_0) J_0(k_\rho R) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.3}$$

La méthode de Galerkin est utilisée pour résoudre les équations intégrales couplées (3.1)

et (3.2). Les courants inconnus sont développés en terme d'un système complet $\boldsymbol{\kappa}_{np}$ et un système orthogonal $\mathbf{f}_{nq}$ de fonctions de base issues du modèle de la cavité. La solution finale peut être ramenée à la forme algébrique de l'équation 3.4.

$$[Z_n] \cdot [I_n] = [V_n] \tag{3.4}$$

La matrice $[Z_n]$ est donnée par (3.5) :

$$[Z_n] = \begin{bmatrix} Z^{k^{(1)}\times k^{(1)}} & Z^{k^{(1)}\times F^{(1)}} & Z^{k^{(1)}\times k^{(2)}} & Z^{k^{(1)}\times F^{(2)}} & Z^{k^{(1)}\times k^{(3)}} & Z^{k^{(1)}\times F^{(3)}} \\ Z^{F^{(1)}\times k^{(1)}} & Z^{F^{(1)}\times F^{(1)}} & Z^{F^{(1)}\times k^{(2)}} & Z^{F^{(1)}\times F^{(2)}} & Z^{F^{(1)}\times k^{(3)}} & Z^{F^{(1)}\times F^{(3)}} \\ Z^{k^{(2)}\times k^{(1)}} & Z^{k^{(2)}\times F^{(1)}} & Z^{k^{(2)}\times k^{(2)}} & Z^{k^{(2)}\times F^{(2)}} & Z^{k^{(2)}\times k^{(3)}} & Z^{k^{(2)}\times F^{(3)}} \\ Z^{F^{(2)}\times k^{(1)}} & Z^{F^{(2)}\times F^{(1)}} & Z^{F^{(2)}\times k^{(2)}} & Z^{F^{(2)}\times F^{(2)}} & Z^{F^{(2)}\times k^{(3)}} & Z^{F^{(2)}\times F^{(3)}} \\ Z^{k^{(3)}\times k^{(1)}} & Z^{k^{(3)}\times F^{(1)}} & Z^{k^{(3)}\times k^{(2)}} & Z^{k^{(3)}\times F^{(2)}} & Z^{k^{(3)}\times k^{(3)}} & Z^{k^{(3)}\times F^{(3)}} \\ Z^{F^{(3)}\times k^{(1)}} & Z^{F^{(3)}\times F^{(1)}} & Z^{F^{(3)}\times k^{(2)}} & Z^{F^{(3)}\times F^{(2)}} & Z^{F^{(3)}\times k^{(3)}} & Z^{F^{(3)}\times F^{(3)}} \end{bmatrix} \tag{3.5}$$

Chaque élément des sous matrices de $[Z_n]$ est donné par :

$$Z_{ij}^{\gamma^{(p)}\chi^{(q)}} = \int_0^\infty dk_\rho k_\rho \boldsymbol{\gamma}_{ni}^{(p)\dagger}(k_\rho) \cdot \mathbf{G}_{ij}(k_\rho) \cdot \boldsymbol{\chi}_{nj}^{(q)}(k_\rho) \tag{3.6}$$

Les quantités $\boldsymbol{\gamma}_{ni}^{(p)}$ et $\boldsymbol{\chi}_{nj}^{(q)}$ représentent $\mathbf{K}_{ni}^{(p)}$ or $\mathbf{F}_{nj}^{(q)}$. Les éléments du vecteur excitation sont donnés par (3.7) :

$$d_{nij}^{\gamma^{(p)}} = -\int_0^\infty dk_\rho k_\rho \boldsymbol{\gamma}_{ni}^{(p)\dagger}(k_\rho) \cdot \mathbf{G}_{i1}(k_\rho) \cdot \mathbf{P}_n(k_\rho) \tag{3.7}$$

L'impédance d'entrée est donnée par (3.8).

$$Z_{in} = -\frac{1}{I^2} \iiint dV \mathbf{E}(\mathbf{r}) \cdot \mathbf{J}_{Probe}(\mathbf{r}) \tag{3.8}$$

$\mathbf{E}$ est le champ électrique total dû aux courants de la sonde coaxiale et des plaque rayonnantes. En utilisant le théorème de Parseval dans le domaine des transformées vectorielles de Hankel [150], on peut écrire l'équation 3.8 sous la forme variationnelle de l'équation 3.9

.

$$Z_{in} = -\frac{2\pi}{I^2} \sum_{n=-\infty}^{\infty} [V_n]^\dagger \cdot [I_n] + \omega\mu d_1 J_0(k_1 R) H_0^{(1)}(k_1 R) \tag{3.9}$$

Le coefficient de réflexion Γ est défini par

$$\Gamma = \frac{Z_{in} - 50}{Z_{in} + 50} \tag{3.10}$$

3.3 Résultats et discussions

Dans cette partie, nous allons discuter de l'implémentation sous MATLAB de la formulation mathématique. Tous les programmes sont développés dans une version pour être exécutée séquentiellement. Il est primordial dans cette phase d'optimiser au maximum les codes développés en exploitant la notion de vectorisation propre à MATLAB. En effet, il est toujours préférable de remplacer l'utilisation de boucles par des opérations vectorielles (ou matricielles) équivalentes. Ceci est d'autant plus vrai si le nombre d'opérations réalisées par la boucle est important. Cette approche est favorisée par le fait que la plupart des fonctions MATLAB sont écrites pour être compatibles avec des tableaux.

Nous avons développer un ensemble de fonctions réalisant les différents parties avec les paramètres suivants :

nfreq = 1800	fréquences dans l'intervalle [1GHz-10GHz].
nkrau1 = 501	Nombre de points d'échantillonnage de l'axe des imaginaires.
nkrau1 = 9501	Nombre de points d'échantillonnage de l'axe des réels.
$d_4(mm)$	Hauteur de la séparation en air supérieure ; huit valeurs [0.50, 1.00, 2.00, 3.20, 4.20, 5.2, 6.4, 7.4]

La figure (3.2) présente le coefficient de réflexion en fonction de la fréquence pour différentes valeurs de $d4$, où le fonctionnement en dual bande est clairement visible.

Le tableau (3.1) donne les valeurs de FL et FU, respectivement les fréquences de résonance basse et haute de l'antenne. Nous pouvons noter que les changements dans FL sont minimes quand il y a une augmentation de FU.

TABLE 3.1 – Fréquences de résonances supérieures FU et inférieures FL de l'antenne pour différentes valeurs de d_4.

d4(mm)	FL (GHz)	FU(GHz)
4.2	3.510	4.050
5.2	3.610	4.200
6.4	3.635	4.360
7.4	3.635	4.450

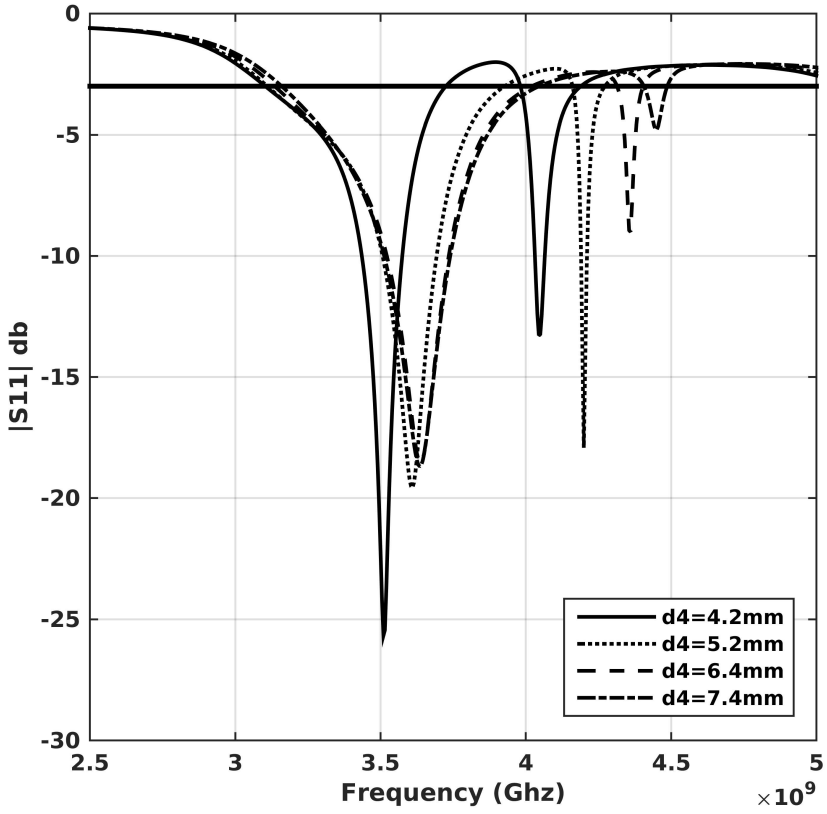


FIGURE 3.2 – Coefficient de réflexion S_{11} en dB de l'antenne microruban multicouche en fonction de la fréquence en GHz pour différentes valeurs de d_4

3.4 Implémentation parallèle

Avant de paralléliser un code, il est primordiale de l'optimiser au maximum pour une utilisation séquentielle pour pouvoir mesurer correctement les performances du code parallèle.

Un aspect important du calcul parallèle est le temps d'exécution parallèle qui regroupe le temps de calcul sur les processeurs et le temps d'échange ou de synchronisation des données. Le temps d'exécution parallèle devrait être plus petit que le temps d'exécution séquentiel sur un processeur pour que la parallélisation soit bénéfique. Pratiquement, Le temps d'exécution parallèle est mesuré entre l'instant du début d'exécution de l'application sur le premier processeur et l'instant de la fin de l'exécution de l'application sur tous les processeurs. Ce temps est influencé par la distribution du travail aux différents processeurs, le temps d'échange ou de synchronisation des informations, et les temps d'inactivité pendant lesquels un processeur est en attente d'un événement.

En général, le temps d'exécution parallèle est le plus petit lorsque la charge de travail est affectée de façon égale aux processeurs, ce que l'on appelle l'équilibrage de charge et lorsque le surcoût relatif aux temps pour l'échange d'informations, la synchronisation et les phases d'inactivité est faible.

Toutefois, trouver une stratégie et un schéma de répartition spécifique conduisant à un bon équilibre de charge et à un surcoût moindre est souvent difficile en raison de nombreuses interactions. Par exemple, la réduction de la surcharge due à l'échange d'informations peut entraîner un déséquilibre de charge, tandis qu'un bon équilibrage de la charge peut nécessiter plus de surcoût en terme d'échange d'informations et/ou de synchronisation.

La calcul de l'impédance pour une seule fréquence prend environ 12 secondes. Un profilage consistant en la mesure des temps d'exécution des différentes fonctions sous MATLAB a montré qu'une parallélisation pour une seule fréquence est inefficace. En effet, la quantité de données échangées entres les différents processus (ou worker dans le langage MATLAB) s'exécutant en parallèles est grande, ainsi, le temps nécessaire pour les communications inter-processus dépassera le gain réalisé en temps d'exécution. Par contre, le temps de calcul de l'impédance d'entrée pour 1800 fréquences également espacées dans l'intervalle [1-10] GHz est de 5 heures 42 minutes. Par conséquent, il est logique de réaliser la parallélisation en distribuant les valeurs du vecteur fréquence sur les différents processus.

Si on a N fréquences et P processus, chaque processus va calculer $\lceil N/P \rceil$ valeurs, avec $\lceil ./. \rceil$ qui représente la division entière. par exemple, pour $N = 1800$ et $P = 32$, chaque processus va calculer $\lceil 1800/32 \rceil = 56$ valeurs de l'impédance. Toutefois, puisque il va rester 8 fréquences non attribuées, il y aura alors 8 processus qui vont exécuter 57 valeurs.

Le modèle SPMD est utilisé pour la parallélisation. Les calculs ne nécessitent pas une grande quantité de RAM, il est alors préférable sinon équivalent de dupliquer les données d'initialisation complètes dans chaque worker. Les données comprennent les paramètres physiques et géométriques de l'antenne qui sont : la perméabilité complexe et la permittivité des couches, l'épaisseur des couches, les rayons des disques circulaires et leurs positions sur les couches et l'intervalle de fréquence. Chaque processus calcule sa partie du vecteur d'impédance d'entrée. Cette partie ne nécessite pas de passage de messages, sauf la diffusion du code aux différents workers, par conséquent, l'accélération de cette étape de calcul est presque optimale. Enfin, les résultats (le vecteur impédance d'entrée) de chaque nœud sont rassemblés dans le nœud principal. Le temps nécessaire pour effectuer cette partie est petit et n'a pas d'effet significatif sur le temps d'exécution total du fait qu'il nécessite une faible quantité de messages transmis.

Dans tout nos développements, les programmes série et parallèle sont mis en œuvre et vérifiés pour s'assurer qu'ils fournissent exactement les mêmes résultats numériques ; par le principe de vérification de la cohérence (en anglais consistency check principle). Un programme parallèle est cohérent s'il donne le même résultat que sa version séquentielle en dépit de l'ordre d'exécution des processus [151, 152].

La figure 3.3 montre le temps d'exécution du code en fonction du nombre de processus (worker).

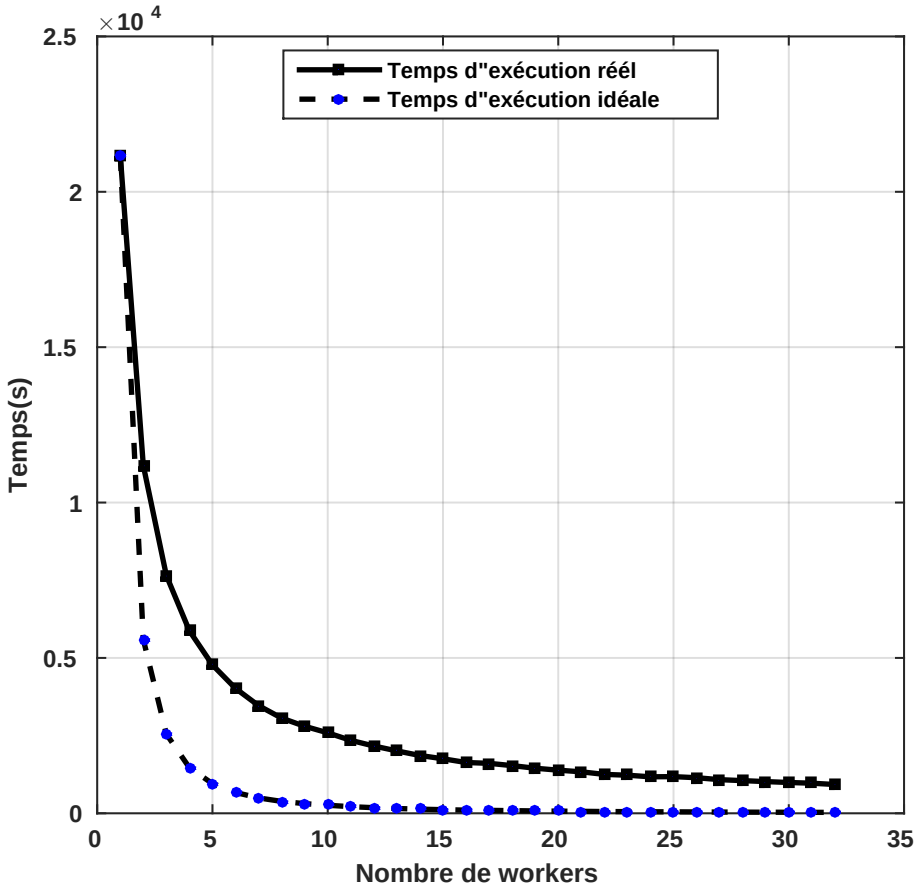


FIGURE 3.3 – Temps d'exécution vs nombre de workers.

La figure 3.4 présente l'accélération obtenue en fonction du nombre de processus (worker). La figure 3.5 présente l'efficacité obtenue en fonction du nombre de processus (worker).

Comme nous pouvons le voir sur la courbe représentant le temps d'exécution, nous gagnons un facteur proportionnel à celui de la courbe idéale sur le temps d'exécution quand

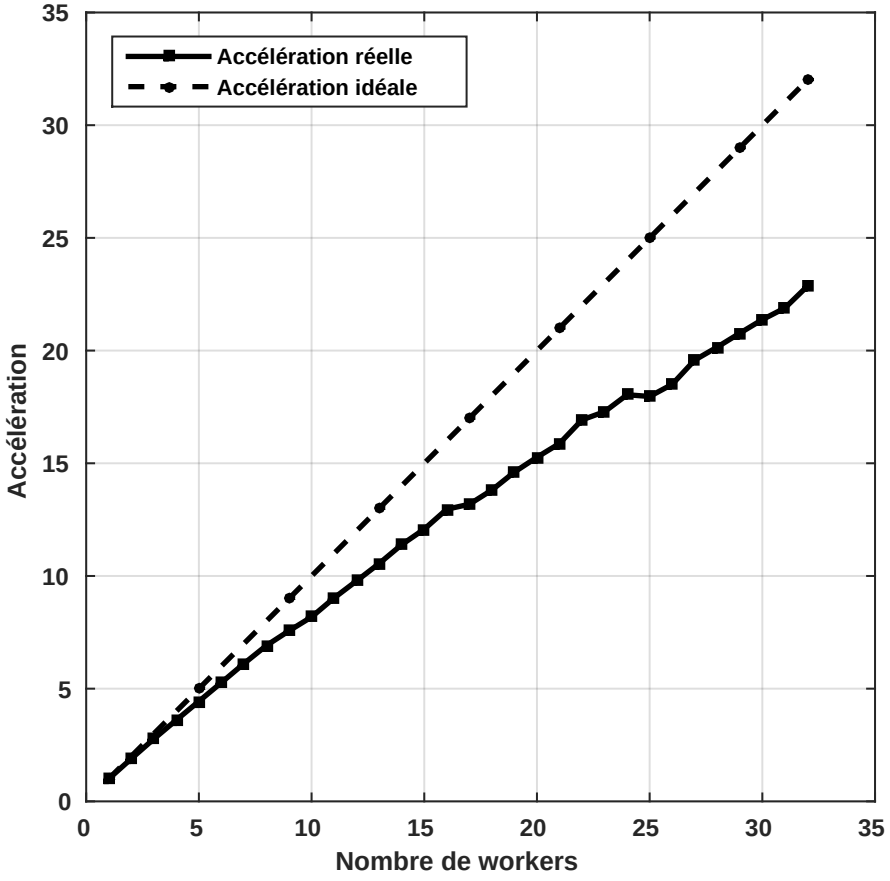


FIGURE 3.4 – Accélération vs nombre de workers.

nous continuons d'ajouter des workers jusqu'à une certaine valeur limite. À partir de 25 workers, on constate une augmentation très faible du gain.

Les courbe d'accélération et de l'efficacité montrent un gain maximale en terme d'accélération atteignant une valeur très proche de 23 pour une valeur idéale de 32 et une efficacité de 70% lorsque tous les workers sont utilisés. Par contre, nous ne pouvons pas déterminer le point de saturation de la courbe d'accélération parce que nous ne pouvons plus ajouter de nouveaux workers du fait que tous les workers ont été utilisés.

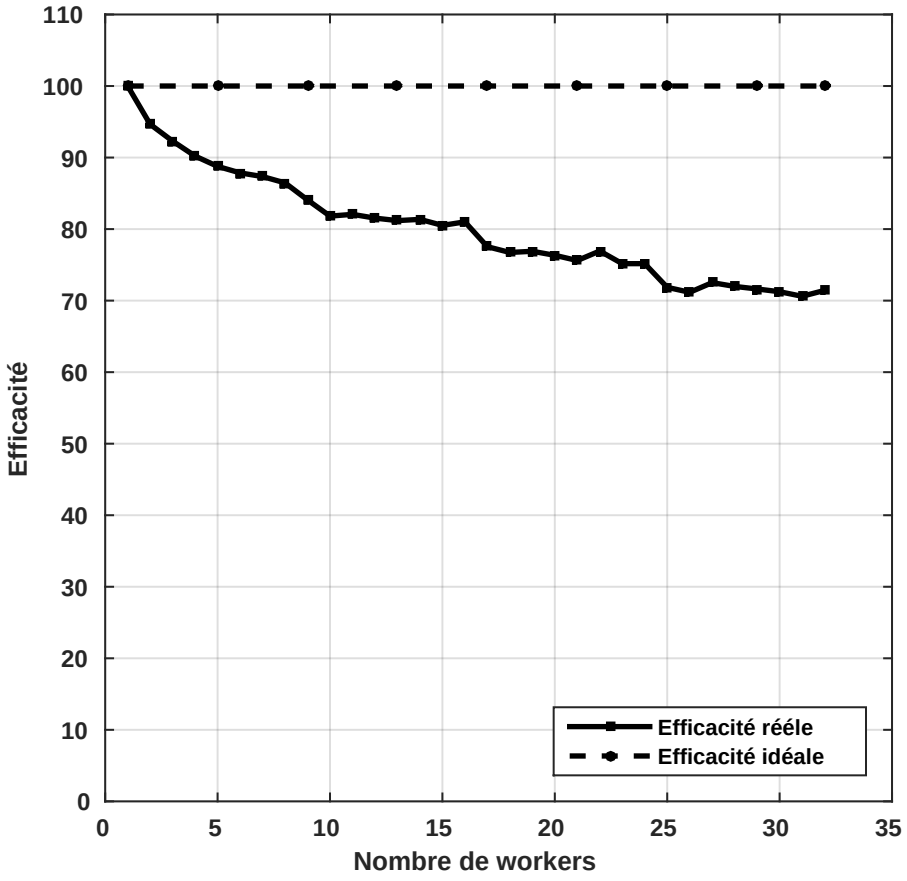


FIGURE 3.5 – Efficacité vs nombre de workers.

3.5 Étude des performances réseaux

Une application scientifique parallèle, s'exécutant sur une grappe de calcul, effectue des phases de calcul et de communication. Les phases de communications peuvent bloquer le processus dans le cas d'une communication synchrone ou bien être non-bloquante dans le cas d'une communication asynchrone.

Plusieurs tâches s'exécutant sur un nœud multiprocesseur ont la possibilité lors de phases de communication simultanées d'utiliser la ressource réseau. A titre d'exemple, une tâche peut envoyer des données pendant qu'une autre tâche sur le même nœud est en phase de réception. Il est alors intéressant d'étudier comment est partagée la ressource réseau entre les tâches. Plus particulièrement, quel est l'impact du partage des ressources, par exemple au niveau de la carte réseaux ou du commutateur, sur les performances des communications,

et, plus généralement, sur les applications [34].

Pour mener cette étude, nous avons adopter une approche très basique qui consiste à analyser et mesurer le coût de la concurrence est l'observation du comportement des communications lors d'expérimentations réelles. En effet, l'étude de l'utilisation des ressources réseaux matérielles ou le traçage logiciel des mécanismes de communication sont difficiles et impliquent un surcoût important sur la mesure obtenue, Nous avons alors mis en œuvre une expérience avec deux scénarios. Dans le premier scénario nous avons procéder aux calculs en ajoutant à chaque fois un worker dans un nœud différent. Si le nombre de workers demandé est supérieur au nombre des nœuds, on ré-alloue à partir du premier noeud et ainsi de suite (Figure 3.6).

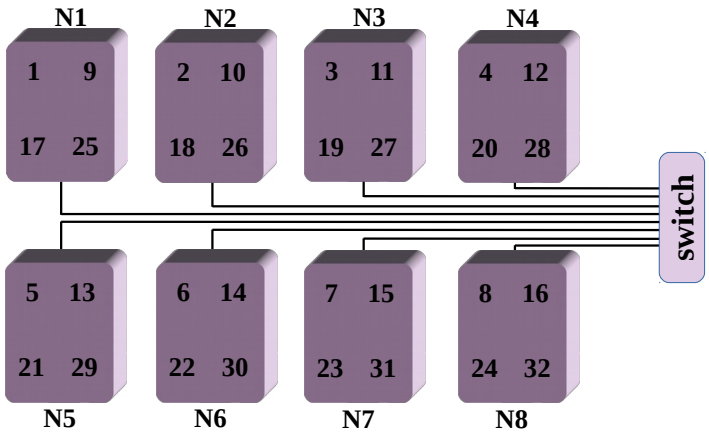


FIGURE 3.6 – Schéma d'ajout d'un nouveau worker pour le scénario 1.

Le deuxième scénario consiste à utiliser tous les cœurs du noeud lors de l'ajout d'un worker avant de passer à un autre noeud et ainsi de suite (Figure 3.7).

En combinant les deux types de communications et les deux types de directions possibles à savoir entrant et sortant, nous obtenons quatre types de conflits possibles, mais dont deux seulement ont un effet notable dans notre cas :

1. Conflit Entrant/Entrant : deux communications arrivent sur le même nœud exécutant deux workers différents. En arrivant simultanément au nœud récepteur, il se produit une répartition des accès aux ressources du nœud ou du commutateur.
2. Conflit Sortant/Sortant : ce conflit est opposé au conflit précédent. En effet, deux communications sortent simultanément d'un même nœud. Ainsi, dès le début des communications, elles sont en concurrence sur les ressources du nœud.

La figure 3.8 présente le temps d'exécution en fonction du nombre de workers des deux

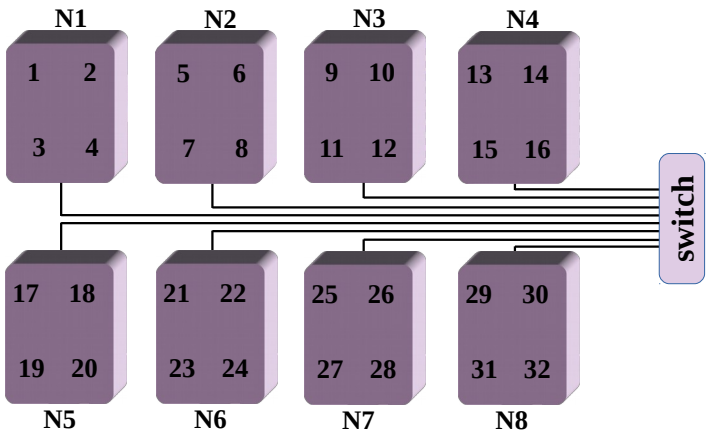


FIGURE 3.7 – Schéma d'ajout d'un nouveau worker pour le scénario 2.

scénarios. La différence entre les deux courbes est représentée par des barres verticales entre les deux courbes.

En se basant sur les types de conflits précédents, nous présentons les résultats obtenus suivant le protocole expérimental. Les valeurs des temps présentées dans les courbes sont les médianes des temps de chaque expérience répétée deux fois. Nous remarquons d'après la figure 3.8 que le temps d'exécution du scénario1 est toujours plus petit que celui du scénario2. On peut remarquer aussi que la différence est grande lorsque dans le deuxième scénario, tous les ressources du nœuds sont utilisés (les quatre cœurs). En effet, on peut expliquer ce phénomène par les conflit Sortant/Sortant lors de l'envoi de messages par les workers d'un même noeud, ce qui a comme effet l'augmentation des collisions dans le protocole Ethernet du réseau Gigabit. En plus, lorsque tous les cœurs du noeud sont utilisés, le système d'exploitation a tendance à interrompre plus souvent les processus de calcul pour ces tâches internes.

Il est par ailleurs, difficile de prévoir exactement les différences compte tenu de la complexité de la pile réseaux Ethernet et le système de préemption du système d'exploitation de type Linux.

3.6 Conclusion

La méthode des moments dans le domaine spectral a été appliquée pour l'étude d'une antenne circulaire multicouches et multiconducteurs. Le calcul de l'impédance d'entrée pour une large bande de fréquence peut devenir intensif et il est alors nécessaire d'utiliser le cal-

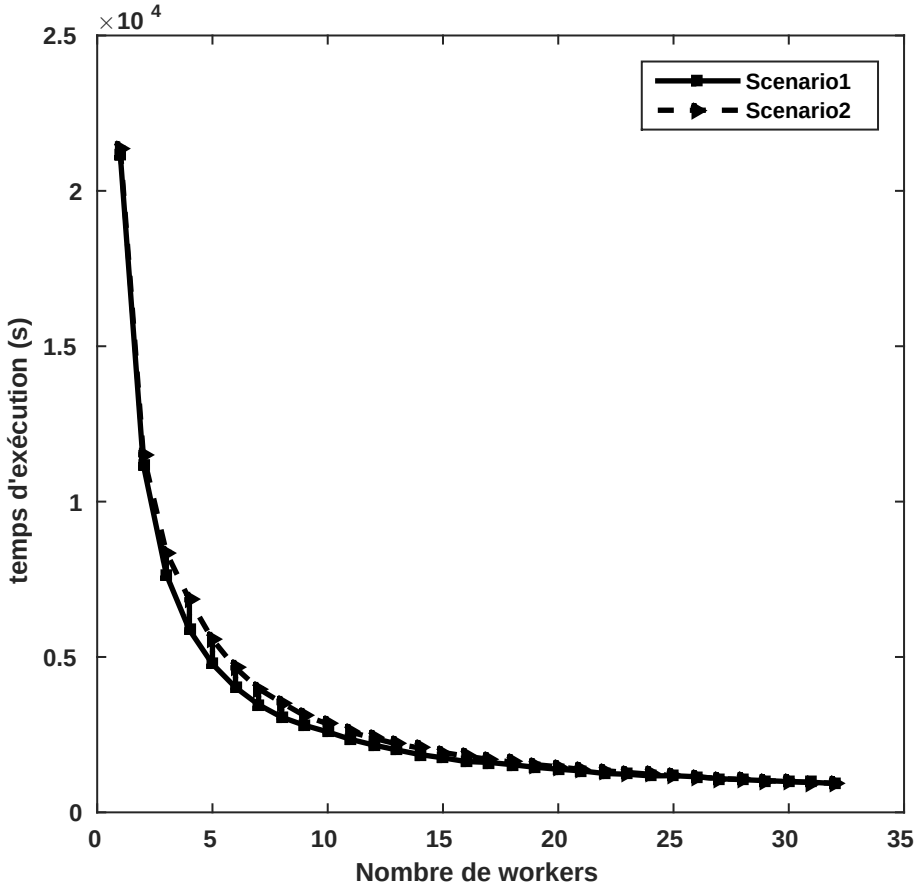


FIGURE 3.8 – Temps d'exécution des deux scénarios en fonction du nombre de workers

cul parallèle pour ce problème. L'utilisation de MATLAB sur une architecture de grappe d'ordinateurs ou cluster s'est avérée une solution avantageuse qui permet de traiter des problèmes de plus en plus larges et complexes en bénéficiant d'un environnement de travail (MATLAB) très apprécié par la richesse des ces toolbox. Le gain mesuré en accélération par rapport au calcul séquentiel prouve l'utilité de telle approche alors même que le modèle de programmation est très différent des habitudes de la programmation séquentielle habituelle.

Chapitre 4

Calcul symbolique des fonctions de forme FEM d'ordre supérieur

Sommaire

4.1 Introduction 75
4.2 Fonctions d'interpolation de base et matrices fondamentales 76
4.3 Évaluation des matrices d'éléments en coordonnées locales 79
4.4 Implémentation parallèle 81
4.5 Résultats et discussions 84
4.6 Conclusion 86

4.1 Introduction

La méthode des éléments finis pour la discrétisation des équations aux dérivées partielles en électromagnétisme est une méthodologie numérique puissante et polyvalente pour la simulation de problèmes électromagnétiques compliqués. Dans une solution d'éléments finis, le domaine entier Ω est subdivisé en un certain nombre d'éléments sur lesquels les fonctions de forme sont définies. Il existe deux principales classes d'éléments utilisés à des fins de simulation électromagnétique : les éléments nodaux et les éléments vectoriels.

Les éléments nodaux utilisant des fonctions de base scalaires conviennent pour approximer des champs scalaires ou vectoriels en approximant séparément chaque composante selon des directions orthogonales. Cette approche présente des problèmes numériques qui

peuvent être surmontés d'une manière ou d'une autre, mais au prix d'une augmentation de la complexité du problème global et de la dégénérescence de la précision de la solution.

Les éléments vectoriels, avec leurs degrés de liberté associés aux bords du maillage, utilisent des fonctions de base vectorielles qui sont plus appropriées pour approcher un champ vectoriel et permettent de forcer les conditions aux limites de façon directe. Ils se sont révélées exemptes des défauts cités auparavant.

Pour le domaine 1D, les éléments sont des segments de ligne avec des polynômes de Lagrange comme fonctions d'interpolation. Les éléments finis les plus couramment utilisées sont les triangles pour les problèmes bidimensionnels (2D) et les tétraèdres pour les problèmes tridimensionnels (3D). En plus de leur capacité à modéliser des géométries très complexes, les éléments triangulaires et tétraédriques peuvent produire des solutions plus précises que les éléments quadrilatéraux et hexaédriques, en réduisant l'erreur de dispersion numérique. La précision peut également être améliorée en utilisant des éléments d'ordre supérieur (quadratique, cubique, ...) [11, 153, 154]. Cependant, la procédure de construction de fonctions d'interpolation d'ordre supérieur pour ces types d'éléments est très fastidieuse. Par conséquent, il est nécessaire d'explorer des approches plus simples et systématiques pour construire les fonctions d'interpolation.

Dans cette section, nous développons un ensemble de fonctions MATLAB qui peuvent être utilisées pour générer des fonctions d'interpolation d'ordre linéaire et arbitraire pour des éléments triangulaires et tétraédriques en utilisant la Toolbox « Calcul symbolique ». La fonction d'interpolation, ses dérivées, les matrices jacobiennes et élémentaires (matrices de raideur et de masse) pour chaque nœud sont calculées symboliquement et stockées sous forme analytique. Les programmes séquentiels et parallèles sont implémentés de manière à pouvoir être facilement généralisés à d'autres types d'éléments finis, tels que les arêtes, les éléments mixtes et/ou d'autres formes, à savoir les prismes quadrilatéraux, hexaédriques et triangulaires . Le calcul numérique des fonctions d'interpolation symboliques peut également être facilement effectué en générant des fichiers MATLAB, C ou FORTRAN correspondants.

4.2 Expressions des fonctions d'interpolation de base et matrices fondamentales

Dans cette section, nous donnons les formules mathématiques pour le calcul des fonctions de base nodales pour les éléments triangulaires et tétraédriques. Le segment de ligne étant

un cas trivial et très simple, nous n'allons pas développer ce cas. En outre, les fonctions de base vectorielles pour les cas 2D ou 3D sont facilement obtenues à partir de contreparties nodales, ceux-ci ne seront pas calculées. Nous abordons aussi Le calcul des matrices fondamentales à savoir la matrice de masse et celle de rigidité.

4.2.1 Élément triangulaire pour les fonctions d'interpolation 2D

La région du problème est discrétisée avec des éléments triangulaires comme montré sur la figure 4.1.

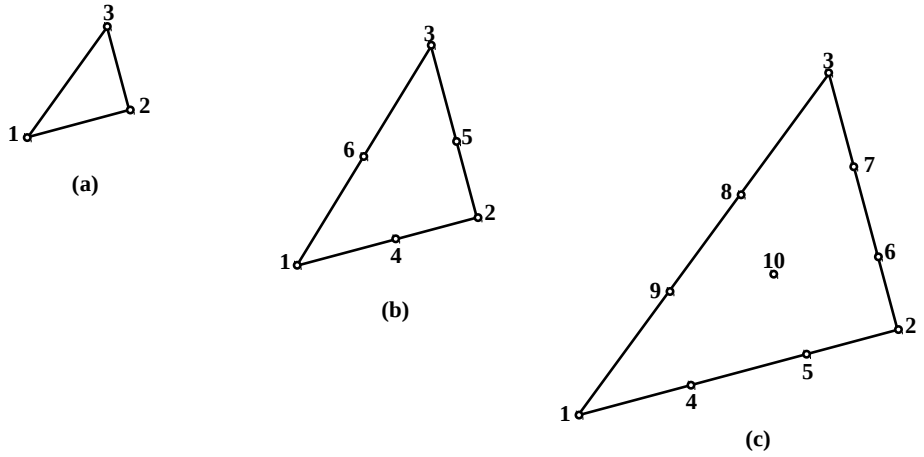


FIGURE 4.1 – Famille des éléments triangulaires : (a) Linéaire, (b) Quadratique , (c) Cubique.

Chaque élément triangulaire a trois fonctions de base basées sur les nœuds. Les fonctions de base nodales, d'ordre arbitraire, pour ce type d'éléments, peuvent être définies par une combinaison de polynôme $P_r^n(\xi)$ comme dans l'équation 4.1.

$$\mathcal{N}_i^e = P_I^n(\xi_1^e)P_J^n(\xi_2^e)P_K^n(\xi_3^e), \qquad I+J+K=n \tag{4.1}$$

avec e représente l'élément, $i = 1, \cdots, m$ l'indice des nœuds de l'élément, $\xi_j (j = 1, \cdots, 3)$ sont les coordonnées locales vérifiant $\sum_{j=1}^{3} \xi_j = 1$. $P_r^n(\xi_j)$ avec $r = I, J, K$ est le polynôme défini par les formules 4.2 et 4.3 suivantes :

$$P_r^n(\xi_j) = \frac{1}{r!}\prod_{p=0}^{r-1}(n\xi - p) \tag{4.2}$$

$$= \begin{cases} \frac{n\xi-r+1}{r} P_{r-1}^n(\xi_j) & \text{if } r > 0\,; \\ 1 & \text{if } r = 0\,; \end{cases} \tag{4.3}$$

$m = \frac{1}{2}(n+1)(n+2)$ est le nombre totale de nœuds.

4.2.2 Élément tétraédrique pour les fonctions d'interpolation 3D

La formulation des éléments finis pour le problème 3D est très similaire au cas 2D. Tout d'abord, la région de solution est divisée en éléments tétraédriques à quatre nœuds comme sur la figure 4.2. Les fonctions de base d'ordre n des éléments tétraédriques peuvent être construites de manière systématique comme suit (13) :

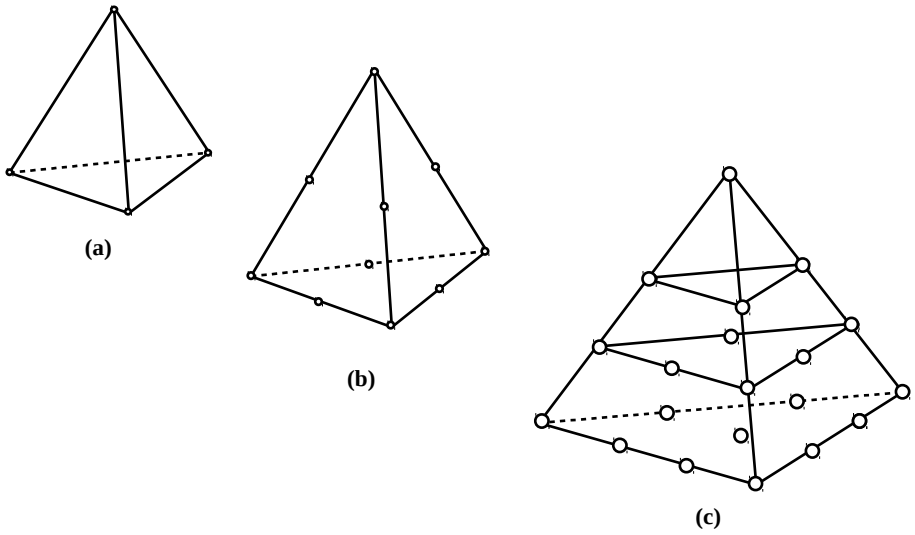


FIGURE 4.2 – Famille des éléments tétraèdres : (a) Linéaire, (b) Quadratique , (c) Cubique.

$$\mathcal{N}_i^e = P_I^n(\xi_1)P_J^n(\xi_2)P_K^n(\xi_3)P_L^n(\xi_4), \qquad I+J+K+L = n \tag{4.4}$$

Le nombre total de nœuds dans le tétraèdre est $m = \frac{1}{6}(n+1)(n+2)(n+3)$.

Une implémentation typique en langage Matlab de la fonction de forme d'ordre supérieur est donnée dans le listing suivant :

```
function [INDICES,N] = shape_function3D(P)
syms tsi1 tsi2 tsi3 tsi4 ;
for n=1:P
   INDICES = [];
   ind= 0;
   for i=0:
      for j=0:n
         ipj = i+j;
```

```
        for k=0:n ipjpk=ipj+k;
           for l=0:n ipjpkpl=ipjpk+l;
              if (ipjpkpl == n)
                 INDICES = [INDICES [i j k l]'];
                 ind = ind+1;
                 expr = prtsi(n,i,tsi1)*prtsi(n,j,tsi2)*...
                 prtsi(n,k,tsi3)*prtsi(n,l,1-(tsi1+tsi2+tsi3));
                 expr1(ind) = factor(expr);
              end
           end
        end
     end
  end
expr2=expr1(ind:-1:1);
N = expr2.';
end
end
```

Où, la fonction *prtsi* représente le polynôme de Silvester. Le listing ci-dessous donne une implémentation récursive possible :

```
function y=prtsi(n,r,xi);
if (r==0)
   y=1 ;
else
   y= (n*xi-r+1)/r*prtsi(n,r-1,xi);
end
```

Pour les fonctions de forme triangulaires 2D, seulement trois appels à la fonction *prtsi* sont effectués dans trois boucles au lieu de quatre.

4.3 Évaluation des matrices d'éléments en coordonnées locales

Dans la formulation par éléments finis des problèmes électromagnétiques, deux matrices fondamentales sont utilisées : masse et rigidité [11]. Les formules pour calculer leurs com-

posantes sont données par 4.5 et 4.6, respectivement.

$$T_{ij}^e = \int_{\Omega_e} d\Omega \mathcal{N}_i^e \mathcal{N}_j^e \tag{4.5}$$

$$S_{ij}^e = \int_{\Omega_e} d\Omega \nabla \mathcal{N}_i^e \cdot \nabla \mathcal{N}_j^e \tag{4.6}$$

Dans (4.5) et (4.6), l'intégration doit être effectuée sur le domaine total des éléments Ω_e.

Pour le cas de l'approximation d'ordre supérieur, la taille des matrices devienne grande, il est alors préférable d'utiliser les coordonnées locales. Une autre motivation pour utiliser les coordonnées locales est la présence d'équations d'intégration et de différenciation simplifiant l'évaluation des matrices fondamentales. La différenciation et l'intégration en coordonnées locales sont réalisées en utilisant la formulation de l'équation 4.7 [11, 153, 154].

$$\begin{bmatrix} \frac{\partial f}{\partial \xi_1} \\ \frac{\partial f}{\partial \xi_2} \\ \frac{\partial f}{\partial \xi_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi_1} & \frac{\partial y}{\partial \xi_1} & \frac{\partial z}{\partial \xi_1} \\ \frac{\partial x}{\partial \xi_2} & \frac{\partial y}{\partial \xi_2} & \frac{\partial z}{\partial \xi_2} \\ \frac{\partial x}{\partial \xi_3} & \frac{\partial y}{\partial \xi_3} & \frac{\partial z}{\partial \xi_3} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \tag{4.7}$$

Pour trouver les dérivées globales, il suffit d'inverser le jacobien **J**. Nous obtenons alors :

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x} & \frac{\partial \xi_2}{\partial x} & \frac{\partial \xi_3}{\partial x} \\ \frac{\partial \xi_1}{\partial y} & \frac{\partial \xi_2}{\partial y} & \frac{\partial \xi_3}{\partial y} \\ \frac{\partial \xi_1}{\partial z} & \frac{\partial \xi_2}{\partial z} & \frac{\partial \xi_3}{\partial z} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial f}{\partial \xi_1} \\ \frac{\partial f}{\partial \xi_2} \\ \frac{\partial f}{\partial \xi_3} \end{bmatrix} = \mathbf{J}^{-1} \cdot \begin{bmatrix} \frac{\partial f}{\partial \xi_1} \\ \frac{\partial f}{\partial \xi_2} \\ \frac{\partial f}{\partial \xi_3} \end{bmatrix} \tag{4.8}$$

Dans l'équation 4.8, l'expression de gauche peut être évaluée car les fonctions f sont spécifiées en coordonnées locales. De plus, comme x, y et z sont explicitement donnés par la relation définissant les coordonnées curvilignes, la matrice jacobienne **J** peut être trouvée explicitement en termes de coordonnées locales.

Pour transformer les variables et le domaine par rapport auquel l'intégration est effectuée, on utilisera un processus standard impliquant le déterminant de **J**. Ainsi, par exemple, un élément de volume devient (équation 4.9) :

$$d\Omega = dxdydz = |\mathbf{J}| \, d\xi_1 d\xi_2 d\xi_3 \tag{4.9}$$

En particulier, le calcul de la matrice $[S^e]$ est plus complexe car il implique l'expression du gradient de la fonction de forme $\mathcal{N}_i^e$ en coordonnées (x, y, z) à celle de la fonction de forme $\mathcal{N}_i^e$ qui est exprimée en termes des coordonnées locales $\xi = (\xi_1, \xi_2, \xi_3)$. Cela peut

être effectué avec la relation (4.10).

$$\nabla_{xyz} f = \mathbf{J}^{-1} \nabla_{\xi} f \tag{4.10}$$

Ainsi, pour les éléments de volume tétraédriques, les matrices scalaires de masse/rigidité des équations (4.5) et (4.6) peuvent être réécrites par les équations 4.11 et 4.12 respectivement.

$$T_{ij}^{e} = 6V \int_0^1 d\xi_3 \int_0^{1-\xi_3} d\xi_2 \int_0^{1-\xi_2-\xi_3} d\xi_1 N_i N_j \tag{4.11}$$

$$S_{ij}^{e} = 6V \int_0^1 d\xi_3 \int_0^{1-\xi_3} d\xi_2 \int_0^{1-\xi_2-\xi_3} d\xi_1 \left(\nabla_{\xi} N_i\right)^T \left[\left(\mathbf{J}^{-1}\right)^T \mathbf{J}^{-1}\right] \nabla_{\xi} N_j \tag{4.12}$$

L'intégrale de l'équation (4.12) implique le calcul de $\mathbf{J}^{-1}$ qui existe analytiquement pour les éléments triangulaires et tétraédriques. Pour l'élément triangulaire, nous obtenons l'équation (4.13) :

$$\left(\mathbf{J}^{-1}\right)^T \mathbf{J}^{-1} = \frac{1}{(2A)^2} \begin{bmatrix} a_1^2 + b_1^2 & a_1 a_2 + b_1 b_2 \\ a_1 a_2 + b_1 b_2 & a_2^2 + b_2^2 \end{bmatrix} \tag{4.13}$$

Et pour l'élément tétraédrique, on obtient (4.14) :

$$\left(\mathbf{J}^{-1}\right)^T \mathbf{J}^{-1} = \frac{1}{(6V)^2} \begin{bmatrix} a_1^2 + b_1^2 + c_1^2 & a_1 a_2 + b_1 b_2 & a_1 c_1 + b_1 c_1 \\ a_1 a_2 + b_1 b_2 & a_2^2 + b_2^2 & a_1 c_1 + b_1 c_1 \\ a_1^2 + b_1^2 + c_1^2 & a_1 a_2 + b_1 b_2 & a_1 c_1 + b_1 c_1 \end{bmatrix} \tag{4.14}$$

Pour les problèmes bidimensionnels, nous ignorons tous les termes contenant z et/ou ξ_3 dans les équations (4.7) à (4.12) ; aussi le terme $6V$ est remplacé par $2A$.

Dans le calcul des intégrales des équations (4.11) et (4.12), il est utile d'utiliser la propriété suivante :

$$\int_0^1 \int_0^{1-\xi_3} \int_0^{1-\xi_2-\xi_3} \xi_1^i \xi_2^j \xi_3^k \xi_4^l d\xi_1 d\xi_2 d\xi_3 = \frac{i!\,j!\,k!\,l!}{(i+j+k+l+3)!} 6V \tag{4.15}$$

4.4 Implémentation parallèle

Dans cette section, nous allons décrire le processus de parallélisation du code Matlab permettant de calculer symboliquement les éléments des matrices de masse et de raideur pour les fonctions de forme d'ordre supérieur. La plupart des logiciels et programmes utilisant la méthode des éléments finis, peu importe qu'ils soient commerciaux ou libres, ne four-

nissent pas une forme explicite des fonctions de base d'ordre supérieur supérieurs à trois pour les tétraèdres. Par exemple, HFSS-15.0 utilise des fonctions de base d'ordres allant jusqu'à deux [155]. Dans la référence [11], l'auteur donne des expressions explicites pour les fonctions de forme 3D d'ordre supérieur allant jusqu'à une valeur de trois. Il est donc nécessaire de développer une méthode systématique pour générer des expressions symboliques de telles fonctions de base pour des ordres supérieurs à trois. Pour ce faire, nous avons développé un ensemble de fonctions en utilisant le toolbox « MatLab Symbolic Toolbox ». Cependant, le temps de calcul peut devenir excessif. Pour remédier à cet inconvénient, la parallélisation est proposée comme une solution efficace.

Considérons le problème de calcul des matrices fondamentales $[T]^e$ et $[S]^e$ lorsque des fonctions de forme d'ordre supérieur sont utilisées. Dans le cas où $n = 10$ nous avons des matrices d'ordre $m = \frac{1}{2}(n+1)(n+2) = 66$. Puisque les matrices sont symétriques, nous ne devons calculer que la moitié des éléments de la matrice. c'est-à-dire $M = 66 * (66+1)/2 = 2211$ éléments pour chaque matrice.

4.4.1 Méthode 1 pour le calcul de la matrice

La première solution consiste à distribuer la matrice suivant la première dimension non singulière selon le schéma de distribution par colonne ou par ligne de Matlab. La figure 4.3 schématise la distribution des éléments suivant les colonnes. Le nombre d'éléments par

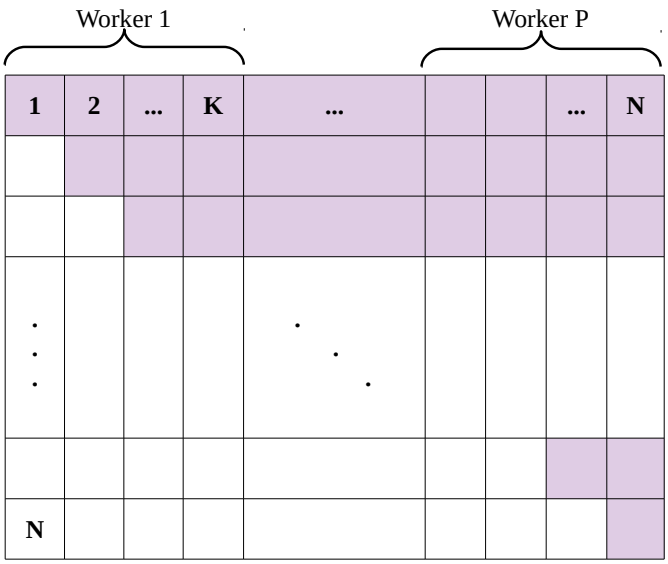


FIGURE 4.3 – Distribution de la matrice suivant les colonnes

worker est donné par la formule 4.16 :

$$NbrElement(j) = \sum_{(j-1)K+1}^{jK} i \qquad \text{where} \qquad 1 \leq j \leq P \tag{4.16}$$

Avec P le nombre de workers disponible, $K = \lceil m/P \rceil$ est le nombre de colonnes attribué à chaque worker et m est l'ordre de la matrice et j est l'indice du worker. Basé sur la figure 4.3, le premier worker calcule $\sum_1^K i = K(K+1)/2$ éléments. C'est la même valeur si l'on prend $j = 1$ dans la formule générale (4.16). De la même manière, le worker P va calculer $somme(m, m-1, ..., m-k+1)$ ou $\sum_{m-K+1}^{m} i$, ce qui correspond à la même valeur calculée par la formule générale (4.16).

Par exemple, pour $n = 10$ on a une matrice de rang $m = 66$, et si nous avons 6 workers $P = 6$, alors $K = \lceil m/P \rceil = \lceil 66/6 \rceil = 11$. Le worker 1 lui sera attribué $11 * (11+1)/2 = 66$ éléments, alors que le processeur 6 (le dernier) aura à calculer $(66*67/2 - (55*56/2) = 671$ éléments.

On remarque que cette solution est inefficace du fait que les workers n'exécutent pas la même charge de travail et que le temps d'exécution global dépendra du worker le plus chargé. Il est clair que l'équilibrage de charge est mauvais.

De plus, si le nombre de workers est important, certains d'entre eux seront inutilisés. en effet, la distribution est faite selon la dimension (par lignes ou par colonnes) et par conséquent, si le nombre de workers est supérieur au nombre de lignes, certains processeurs seront inactifs.

4.4.2 Méthode 2 pour le calcul de la matrice

La deuxième méthode consiste à faire la distribution selon les éléments et non pas suivant la dimension de la matrice. En effet, le problème est transformé en un vecteur (1D) et la distribution est représentée dans la figure 4.4. Si on prend le même exemple précédent, nous aurons $m = 2211$ éléments de matrice à distribuer aux $P = 6$ workers disponible. Le nombre d'éléments par worker est alors $\lceil 2211/6 \rceil = 308$. Cependant, trois des six workers vont devoir calculer un élément en plus parce que $308 * 6 = 2208$ alors que nous avons 2211 éléments. Le fait que la distribution soit faite par élément permet une discrétisation plus fine du problème. En effet, Le nombre d'éléments est grand et par conséquent, la distribution est meilleure. L'équilibrage de charge est nettement meilleur que le premier cas de façon à assurer un temps d'exécution approximativement égal pour chaque processeur.

Lignes	1	2	3	4	5	6	...	42	43					66
worker	Worker1 = 308 66+65+64+63+50						Worker2 = 308, Worker3= 308 Worker4 = 309, Worker5= 309					Worker6= 309 1+2+...+24+9		

FIGURE 4.4 – Distribution des éléments de la matrice en mode 1D

4.5 Résultats et discussions

Dans le tableau 4.1, nous donnons un exemple des dix fonctions de forme pour un élément 2D triangulaire pour un ordre $n = 3$. Le nombre de nœuds est égale à $\frac{1}{2}(n+1)(n+2) = 10$.

TABLE 4.1 – Les fonctions de formes sous format symbolique pour P=3,m=10

Index	Expression
$N_1^e =$	$\frac{9\,\xi_1{}^3}{2} - \frac{9\,\xi_1{}^2}{2} + \xi_1$
$N_2^e =$	$\frac{(27\,\xi_2)}{2}\,\xi_1{}^2 + -\frac{9\,\xi_2}{2}\,\xi_1$
$N_3^e =$	$-\frac{27\,\xi_1{}^3}{2} + \left(18 - \frac{27\,\xi_2}{2}\right)\,\xi_1{}^2 + \left(\frac{9\,\xi_2}{2} - \frac{9}{2}\right)\,\xi_1$
$N_4^e =$	$\left(9\,\xi_2\,\left(\frac{3\,\xi_2}{2} - \frac{1}{2}\right)\right)\,\xi_1$
$N_5^e =$	$(-27\,\xi_2)\,\xi_1{}^2 + (-9\,\xi_2\,(3\,\xi_2 - 3))\,\xi_1$
$N_6^e =$	$\frac{27\,\xi_1{}^3}{2} + \left(27\,\xi_2 - \frac{45}{2}\right)\,\xi_1{}^2 + \left(3\,(3\,\xi_2 - 3)\,\left(\frac{3\,\xi_2}{2} - 1\right)\right)\,\xi_1$
$N_7^e =$	$\xi_2 - \frac{9\,\xi_2{}^2}{2} + \frac{9\,\xi_2{}^3}{2}$
$N_8^e =$	$\left(-9\,\mathrm{i}_2\,\left(\frac{3\,\xi_2}{2} - \frac{1}{2}\right)\right)\,\xi_1 - 3\,\xi_2\,(3\,\xi_2 - 3)\,\left(\frac{3\,\xi_2}{2} - \frac{1}{2}\right)$
$N_9^e =$	$\frac{(27\,\xi_2)}{2}\,\xi_1{}^2 + \left(\frac{9\,\xi_2\,(3\,\xi_2 - 3)}{2} + 9\,\xi_2\,\left(\frac{3\,\xi_2}{2} - 1\right)\right)\,\xi_1 + 3\,\xi_2\,(3\,\xi_2 - 3)\,\left(\frac{3\,\xi_2}{2} - 1\right)$
$N_{10}^e =$	$-\frac{9\,\xi_1{}^3}{2} + \left(9 - \frac{27\,\xi_2}{2}\right)\,\xi_1{}^2 + \left(-\left(9\,\xi_2 - \frac{15}{2}\right)\,\left(\xi_2 - \frac{1}{3}\right) - (3\,\xi_2 - 3)\,\left(\frac{3\,\xi_2}{2} - 1\right)\right)$ $\xi_1 - (3\,\xi_2 - 3)\,\left(\frac{3\,\xi_2}{2} - 1\right)\,\left(\xi_2 - \frac{1}{3}\right)$

Le tableau 4.2 présente quelques fonctions de formes d'ordre supérieur avec $n = 3$ pour un élément 3D tétraédrique. Le nombre de nœuds est égale à $\frac{1}{6}(n+1)(n+2)(n+3) = 20$.

TABLE 4.2 – Les fonctions de formes sous format symbolique pour P=3,m=10

Index	Expression
$N_1^e =$	$\frac{9\,\xi_1^{\,3}}{2} - \frac{9\,\xi_1^{\,2}}{2} + \xi_1$
$N_2^e =$	$\frac{(27\,\xi_2)}{2}\,\xi_1^{\,2} + -\frac{9\,\xi_2}{2}\,\xi_1$
$N_3^e =$	$\frac{(27\,\xi_3)}{2}\,\xi_1^{\,2} + -\frac{9\,\xi_3}{2}\,\xi_1$
$N_4^e =$	$-\frac{27\,\xi_1^{\,3}}{2} + \left(18 - \frac{27\,\xi_3}{2} - \frac{27\,\xi_2}{2}\right)\,\xi_1^{\,2} + \left(\frac{9\,\xi_2}{2} + \frac{9\,\xi_3}{2} - \frac{9}{2}\right)\,\xi_1$
$N_{10}^e =$	$\frac{27\,\xi_1^{\,3}}{2} + \left(27\,\xi_2 + 27\,\xi_3 - \frac{45}{2}\right)\,\xi_1^{\,2} + \left(3\,(3\,\xi_2 + 3\,\xi_3 - 3)\,\left(\frac{3\,\xi_2}{2} + \frac{3\,\xi_3}{2} - 1\right)\right)\,\xi_1$
$N_{19}^e =$	$\frac{(27\,\xi_3)}{2}\,\xi_1^{\,2} + \left(\frac{9\,\xi_3\,(3\,\xi_2+3\,\xi_3-3)}{2} + 9\,\xi_3\,\left(\frac{3\,\xi_2}{2} + \frac{3\,\xi_3}{2} - 1\right)\right)\,\xi_1$ $+3\,\xi_3\,(3\,\xi_2 + 3\,\xi_3 - 3)\,\left(\frac{3\,\xi_2}{2} + \frac{3\,\xi_3}{2} - 1\right)$
$N_{20}^e =$	$-\frac{9\,\xi_1^{\,3}}{2} + \left(9 - \frac{27\,\xi_3}{2} - \frac{27\,\xi_2}{2}\right)\,\xi_1^{\,2} - (3\,\xi_2 + 3\,\xi_3 - 3)\,\left(\frac{3\,\xi_2}{2} + \frac{3\,\xi_3}{2} - 1\right)\,\left(\xi_2 + \xi_3 - \frac{1}{3}\right)$ $\left(-\left(\frac{3\,\xi_2}{2} + \frac{3\,\xi_3}{2} - 1\right)\,(6\,\xi_2 + 6\,\xi_3 - 4) - \frac{3\,(3\,\xi_2+3\,\xi_3-3)\,\left(\xi_2+\xi_3-\frac{1}{3}\right)}{2}\right)\,\xi_1$

La figure 4.5 montre le temps d'exécution en fonction du nombre de workers. La courbe de

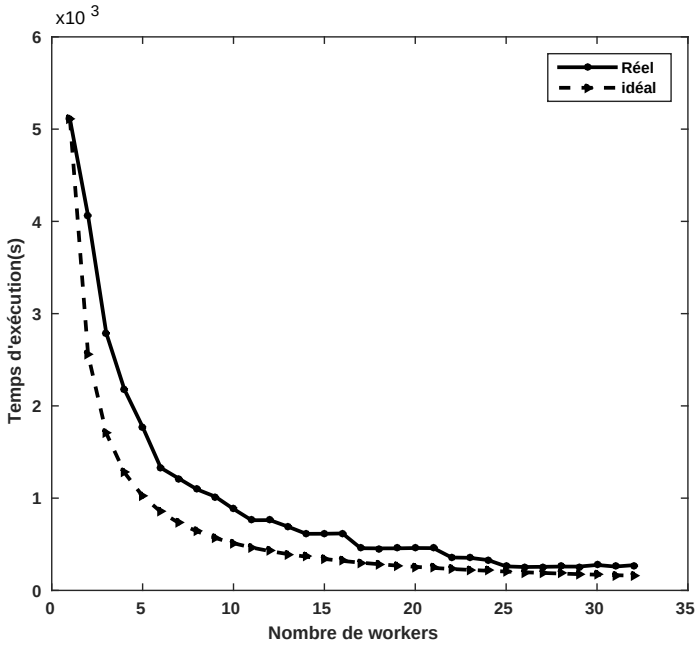


FIGURE 4.5 – Temps d'exécution en fonction du nombre de workers

diminution du temps d'exécution montre un comportement très proche de la courbe idéale en $1/x$. En effet, en adoptant un schéma de distribution de charge optimal, il est toujours possible de gagner en accélération.

La figure 4.6 montre l'accélération du code parallèle développé en fonction du nombre de workers. La figure 4.7 montre l'efficacité du code parallèle développé en fonction du nombre

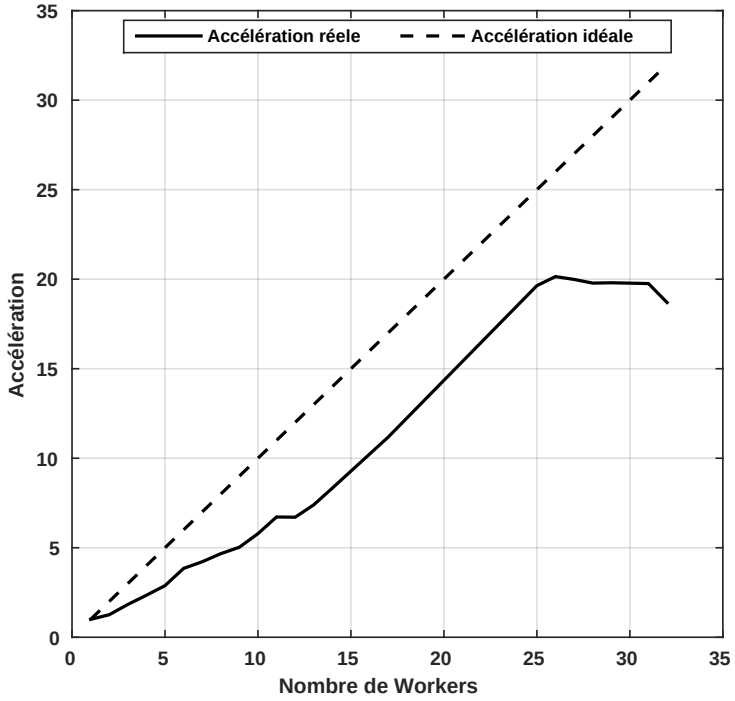


FIGURE 4.6 – Accélération en fonction du nombre de workers

de workers.

4.6 Conclusion

Dans ce chapitre, nous avons implémenter un algorithme parallèle pour le calcul des fonctions de forme d'ordre supérieur de la FEM sous format symbolique sous MATLAB. Les fonctions de formes d'ordre allant jusqu'à dix peuvent être calculées et stockés sous format symboliques pour être utilisées ultérieurement. Ce calcul nécessite une grande puissance surtout pour les ordres supérieurs à trois et l'utilisation du calcul parallèle s'avère être efficace pour réduire le temps de calcul. Nous avons ainsi atteint un gain en accélération d'une valeur proche de 21 en utilisant un schémas de distribution de charge adapté.

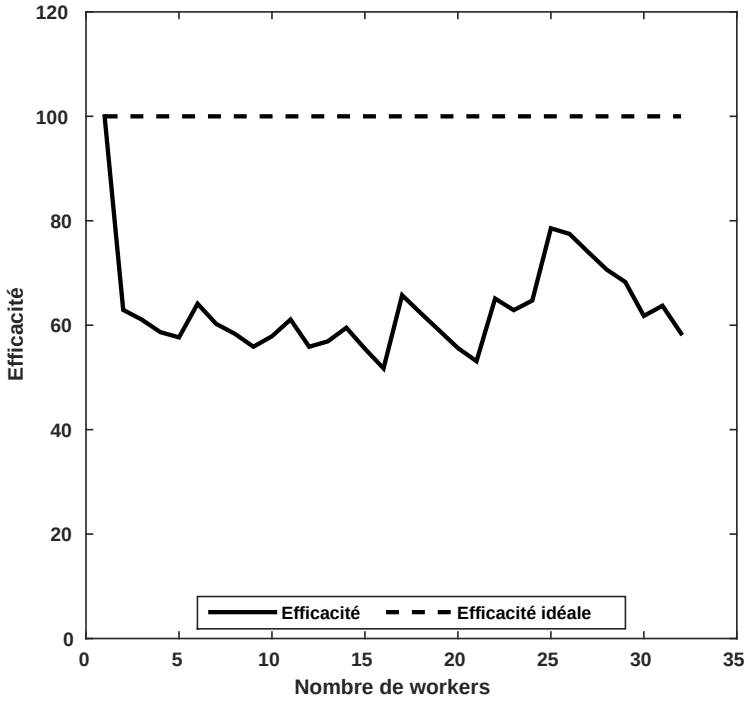


FIGURE 4.7 – Efficacité en fonction du nombre de workers

Chapitre 5

Analyse d'un réseaux d'antennes de type bowtie

Sommaire

5.1 Introduction 88
5.2 Structure de l'antenne 89
5.3 Méthode d'analyse et Matrice d'impédance 89
5.4 Processus de parallélisation 90
5.5 Étude de la scalabilty (évolutivité) 99
5.6 Conclusion 101

5.1 Introduction

Dans de nombreuses applications, une antenne doit fonctionner sur une large gamme de fréquences. Une telle antenne est appelée antenne à large bande. Les paramètres les plus importants de toute antenne sont ses caractéristiques dans le domaine fréquentiel. L'impédance d'entrée, la puissance rayonnée, les diagrammes de directivité et le gain doivent être calculés dans une bande passante donnée [156]. En générale, le diagramme de rayonnement d'un seul élément est relativement large et chaque élément fournit de faibles valeurs de directivité (gain). Dans de nombreuses applications, il est nécessaire de concevoir des antennes avec des caractéristiques très directives (gains très élevés) pour répondre aux exigences de la communication à longue distance. Ceci peut seulement être accompli en augmentant la taille électrique de l'antenne.

L'augmentation des dimensions des éléments individuels conduit souvent à des caractéristiques plus directives. Une autre manière d'agrandir les dimensions de l'antenne, sans nécessairement augmenter la taille des éléments individuels, consiste à former un ensemble d'éléments rayonnants dans une configuration électrique et géométrique. Cette nouvelle antenne, formée de multi-éléments, est appelée réseau d'antennes. Dans la plupart des cas, les éléments du réseau sont identiques [46].

Nous allons, dans ce chapitre discuter de l'implémentation parallèle des programmes d'analyses d'un réseau d'antenne de type bowtie étudié par Makarov [156]. Nous démontrerons que l'utilisation d'un clusters de calcul est nécessaire si nous voulons étudier des structures aussi complexes. Il est alors possible d'augmenter la taille de la structure en augmentant le nombre d'éléments du réseau et/ou augmenter la précision de la discrétisation des équations dans la solution.

5.2 Structure de l'antenne

La figure 5.1 montre la structure de l'antenne étudiée. L'antenne est composée d'une matrice

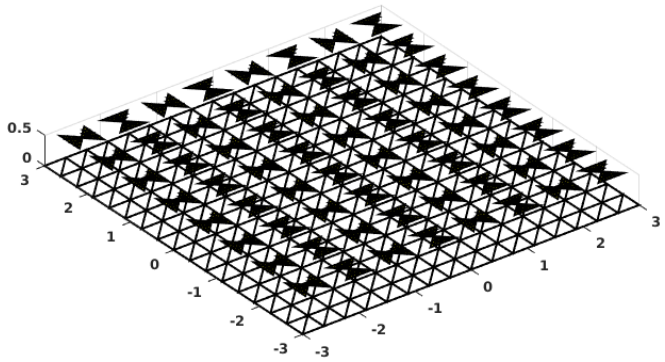


FIGURE 5.1 – Structure du réseau d'antennes étudié par Makarov [156]

8 × 8 éléments de petit antennes bowtie (Figure 5.2) déposé sur un plan de masse. chaque élément mesure $0.5 \times 0.45m$ avec un angle d'ouverture de 45. Le plan de masse mesure $6 \times 6m$ et la distance entre le plan de masse et le réseau est $0.5m$. Cette conception est basée sur un des prototypes développés par « Seavey Engineering Associates, Inc., Massachusetts » [156].

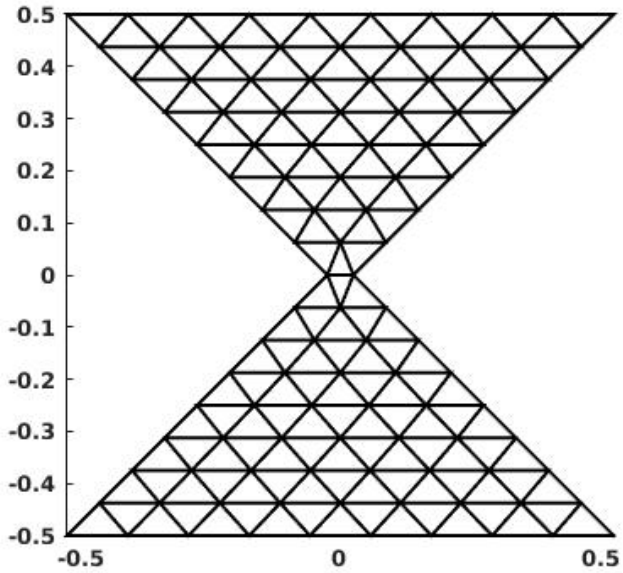


FIGURE 5.2 – Antenne bowtie

5.3 Méthode d'analyse et Matrice d'impédance

Nous n'allons pas donner tous les détails de la méthode utilisée pour l'analyse de l'antenne. Pour plus de détails théoriques et d'implémentation, nous y invitons le lecteur à consulter la référence [156].

La méthode des moments utilisée est basée sur les éléments vectorielles RWG [87]. Premièrement, la surface conductrice de l'antenne est divisée en triangles (Figure 5.3). Chaque paire de triangles, ayant une arête commune constitue un élément vectoriel RWG (Figure). Un des deux triangles est noté avec un signe (+) et l'autre avec un signe (-). Une fonction de base donnée par l'équation () est assigné à l'élément RWG.

$$f(\mathbf{r}) = \begin{cases} (l/2A^+)\rho^+(\mathbf{r}), & \mathbf{r} \in T^+ \\ (l/2A^+)\rho^-(\mathbf{r}), & \mathbf{r} \in T^- \\ 0, & \textit{Otherwise} \end{cases} \tag{5.1}$$

Avec l est la longueur de l'arête et $A^{\pm}$ est la surface du triangle $T^{\pm}$ respectivement. Le vecteur ρ^+ relie le sommet ou vertex libre du triangle T^+ au point d'observation $\mathbf{r}$. Le vecteur ρ^- relie le point d'observation au sommet ou vertex libre du triangle T^-.

La fonction de base (5.1) correspond approximativement à un petit dipôle électrique de longueur $d = |\mathbf{r}^{c-} - \mathbf{r}^{c+}|$ (Figure 5.3). L'indice c indique le centre du triangle $T^{\pm}$. La taille de

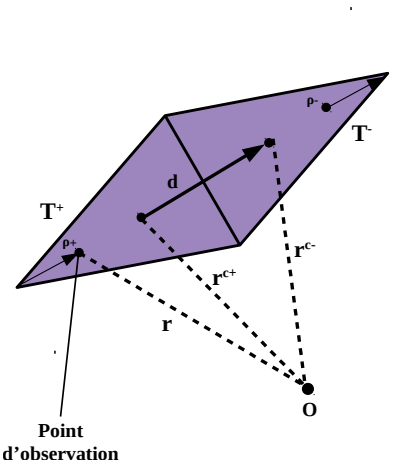


FIGURE 5.3 – Schéma d'un élément arête RWG et l'interprétation en dipôle

la matrice impédance est égale au nombre des arêtes.

5.4 Processus de parallélisation

Dans la référence [156], l'auteur a utilisé les paramètres suivants : chaque bowtie est divisé en 30 triangles ; la plaque de masse en 512 triangles. Le nombre total d'arêtes RWG de la structure est 2976 (le nombre de triangles est 2432). Chaque bowtie est une antenne alimentée au centre (type dipôle), avec l'arête d'alimentation situé exactement dans la jonction centrale (Figure 5.2).

La figure (5.4) présente l'algorithme de calcul des caractéristiques de l'antenne avec les noms de scripts originales de la référence [156].

Le mappage des éléments du réseau correspond à celui de la figure (5.5).

Les calculs font reparaître le problème de la mémoire RAM nécessaire pour résoudre le problème dans la cas ou on augmente le nombre des triangles. En effet, pour des configurations plus grandes ou/et une discrétisation plus fine (nombre de triangles plus grand), la mémoire de travail peut rapidement atteindre des dizaines de Gigaoctets et le temps de calcul croit exponentiellement. Nous proposons de paralléliser l'application et de l'exécuter sur notre cluster. Ainsi, nous pouvons calculer des structures beaucoup plus grandes avec une discrétisation plus fine.

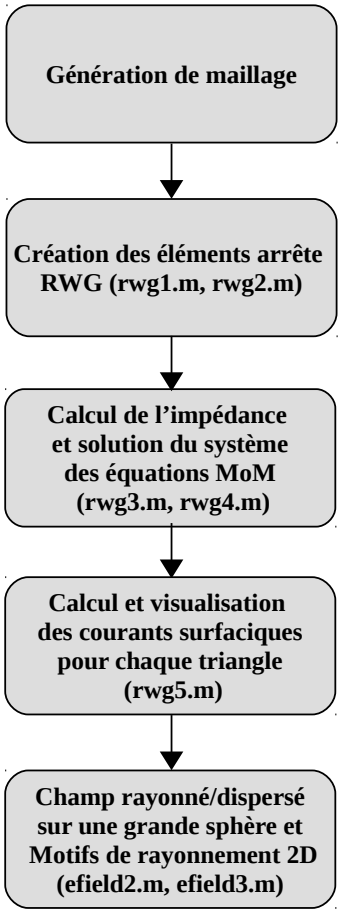


FIGURE 5.4 – Organigramme du calcul des caractéristiques de l'antenne

5.4.1 Partie#1 : calcul des caractéristiques des arêtes

Dans cette partie, les différents paramètres du maillage sont calculés. Cette partie utilise le maillage préalablement calculé et sauvegardé dans un fichier. Pour chaque paire de triangles, on définit les paramètres : arête commune, longueur de l'arête, surface du triangle, les sous-triangles et leurs barycentres, ainsi que les différents vecteurs schématisés dans la figure 5.3.

L'algorithme consiste à parcourir tous la triangles et pour chaque triangle, il faut trouver les triangles adjacents pour former une paire et trouver l'arête commune. Il est alors primordiale de trouver la partition optimale qui permet une distribution des triangles sur les workers et ainsi assurer une distribution de charge optimale. Nous avons développer une fonction qui permet de trouver cette partition en fonction du nombre de workers disponibles. Basé sur le modèle SPMD, la parallélisation du code a nécessité une réécriture du code pour minimiser les communications inter-processus. L'occupation mémoire de cette

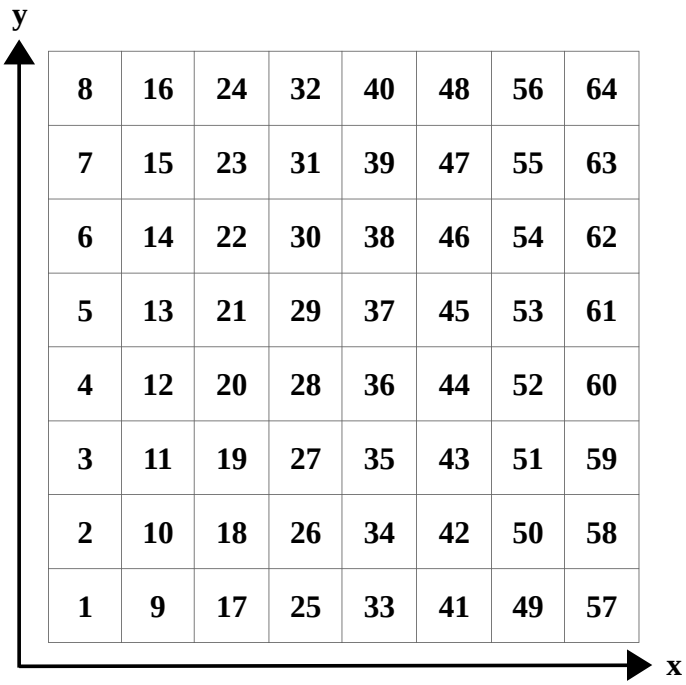


FIGURE 5.5 – Schémas de la numérotation des éléments du réseau d'antennes

partie est petite et il est possible de calculer des structures très complexes. Cependant, il serait impossible de résoudre le système linéaire de la méthode de moment de la partie deux pour un système dépassant quelques centaines d'antennes bowtie.

La figure (5.6) représente le temps d'exécution de la partie #1 en fonction du nombre de workers.

La figure (5.7) représente l'accélération de la partie #1 en fonction du nombre de workers.

Les figures montrent une accélération s'approchant du cas idéale. Il faut noter par contre que le temps de chargement des données de maillage à partir de fichier n'est pas compté. Cependant, les données utilisées occupent quelques mégaoctets et le temps de distribution au différents workers est négligeable. L'accélération maximale atteinte est proche de 28, ce qui représente une valeur très proche du cas idéale qui correspond à une accélération maximale de 32.

5.4.2 Partie#2 : Calcul de la matrice impédance

La matrice d'impédance est une matrice carrée qui détermine l'interaction électromagnétique entre les différents arêtes. Si les arêtes m et n sont considérées comme des dipôles électriques petits et finis, l'élément Z_{mn} décrit la contribution du dipôle n (à travers le champ

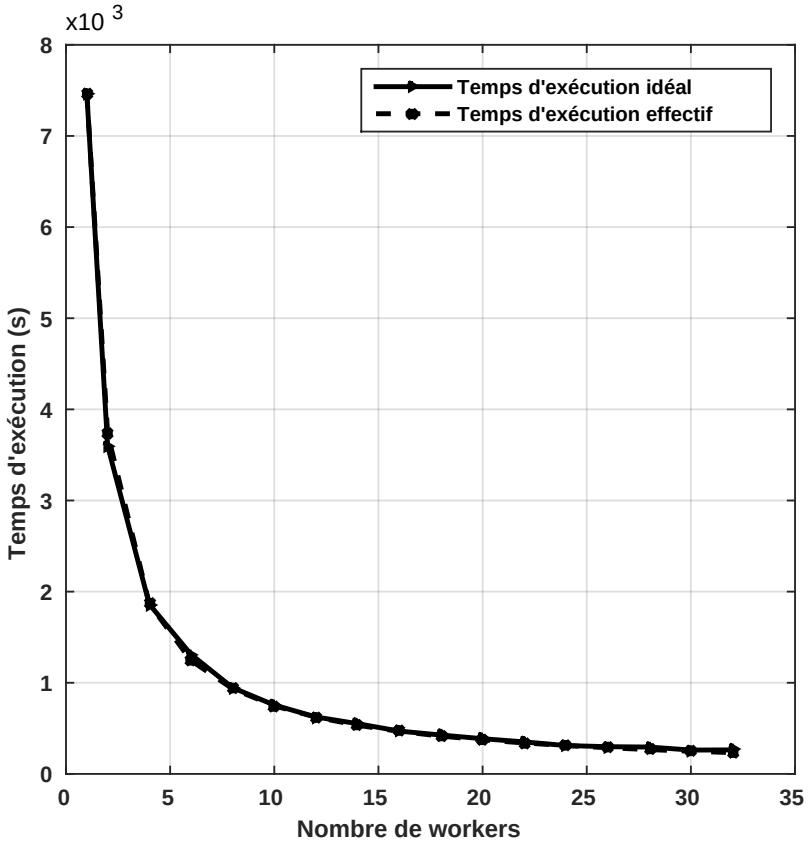


FIGURE 5.6 – Temps d'exécution de la partie 1 en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes

rayonné) au courant électrique du dipôle m et vice versa. Ainsi, la taille de la matrice d'impédance est égale au nombre d'arêtes [156].

Le calcul de la matrice d'impédance d'une antenne constitue le plus grand pourcentage dans le processus d'analyse d'une antenne. En même temps, le calcul de la matrice d'impédance est très compliqué et est considéré comme la première source des erreurs de programmation. La matrice d'impédance ne dépend pas de l'énoncé du problème (rayonnement ou diffusion), mais elle dépend de la fréquence, de la permittivité électrique (constante diélectrique) de l'espace libre ε et de la perméabilité magnétique μ. Le script originale a été modifié de manière à optimiser les calculs, minimiser les communications inter-processus en choisissant le meilleur schéma possible pour la distribution des éléments de la matrice en prenant en considération les étapes suivantes (Solution du système linéaire de la méthode des moments). En effet, il est pratiquement impossible de choisir deux schémas différents pour les deux étapes du fait que la matrice Z peut occuper plus de 50 Gigaoctets impos-

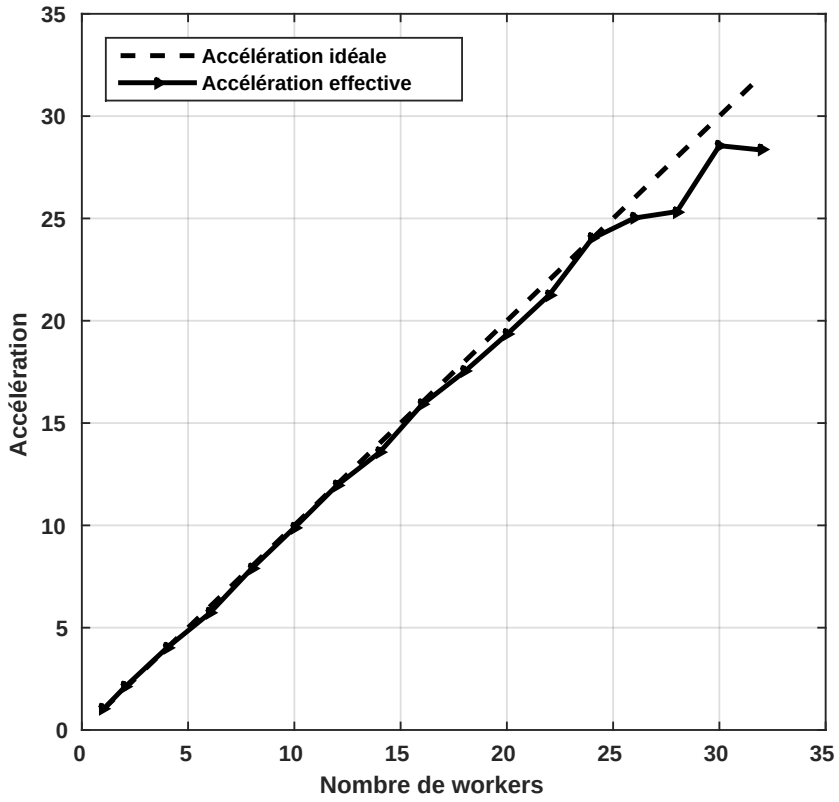


FIGURE 5.7 – Accélération de la partie 1 en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes

sible à redistribuer par manque d'espace et parce que le temps nécessaire pour redistribuer la matrice suivant un autre schéma est très grand selon les tests réalisés.

Dans cette partie, il est impossible d'utiliser moins de huit workers distribués sur les 8 nœuds parce qu'il faut utiliser toute la mémoire disponible sur les tous nœuds étant donnée la taille de la matrice impédance. De cd fait, les courbes de mesure des performances parallèles en fonction du nombre de workers sont données à partir d'un nombre de workers égale à huit.

La table 5.1 donne les valeurs de l'impédance terminale $Z_{in}(\Omega)$ d'un échantillon d'éléments du réseau à la fréquence $160MHz$. Les indices sont choisis au hasard et les données rapportées représentent les éléments dont les lignes et les colonnes appartiennent à l'ensemble $1, 2, 5, 6, 9, 10, 14$.

A cette fréquence, les éléments d'antenne présentent une réflexion minimale correspon-

TABLE 5.1 – Impédance terminal ×100 à la fréquence $160 MHz\Omega$

0.56-0.20j	0.30-0.14j	0.41-0.31j	0.21-0.10j	0.40-0.30j	0.19-0.10j	0.26-0.09j
0.18-0.36j	0.35+0.01j	0.13-0.50j	0.23-0.02j	0.17-0.51j	0.22-0.04j	0.27+0.003j
0.36-0.36j	0.38-0.01j	0.36-0.50j	0.24-0.03j	0.33-0.47j	0.22-0.05j	0.29-0.001j
0.29-0.42j	0.39-0.05j	0.25-0.53j	0.24-0.05j	0.28-0.51j	0.22-0.06j	0.30-0.03j
0.38-0.45j	0.39-0.05j	0.28-0.52j	0.24-0.04j	0.28-0.49j	0.22-0.05j	0.29-0.02j
0.32-0.37j	0.38-0.02j	0.30-0.51j	0.23-0.03j	0.32-0.48j	0.20-0.05j	0.28-0.01j
0.40-0.32j	0.41-0.06j	0.40-0.50j	0.25-0.06j	0.41-0.44j	0.23-0.08j	0.30-0.03j

dant à la résonance. Le rapport maximum-minimum des grandeurs de la matrice d'impédance d'entrée est $frac\ max\ left|Z_{ij}^{in}(f)\ right|\ min\ left|Z_{ij}^{in}(f)\ right|$ = 3.029, pour i, j = 1, $ldots, N$ et f = 160 MHz. Ceci indique qu'il existe des courants électriques non-évanouissants sur toutes les antennes Bowtie.

La figure 5.8 présente le temps d'exécution de la partie calcul de la matrice impédance en fonction du nombre de workers. Il est clair que la courbe idéale ne peut être calculée parce que cette partie ne peut être exécuté séquentiellement (sur un seul worker) comme cité précédemment.

L'accélération ne peut être calculée parce qu'elle dépend du temps d'exécution séquentiel. D'après la figure 5.8, nous pouvons remarquer que le gain est substantiel malgré le volume des données échangées lors des communications inter-processus. Ceci est du essentiellement au choix du schéma de distribution de la matrice impédance. Le gain est approximativement égale à 6 lorsqu'on passe de 8 workers à 32.

5.4.3 Partie#3 : Détermination des courants et solution de l'équation MoM

Dans cette partie, nous utilisons principalement une implémentation basée sur le modèle SPMD du script original pour trouver les courants des arêtes. En identifiant l'arête de l'alimentation, on fixe le voltage en ajustant une onde d'amplitude fixe de $1V$. Le système linéaire est ensuite résolu en utilisant une version surchargée de la division matricielle de MATLAB basée sur la méthode par élimination de Gauss. La solution est un vecteur dont les composantes sont le courant sur chaque arête obtenue en rassemblant toutes les parties calculées sur les différents workers. Après avoir obtenu la distribution sur ces éléments de l'antenne, tous les paramètres d'antenne peuvent être estimés. Pour cette application, les coefficients de réflexion et le diagramme de rayonnement du réseau 2D sont rapportés.

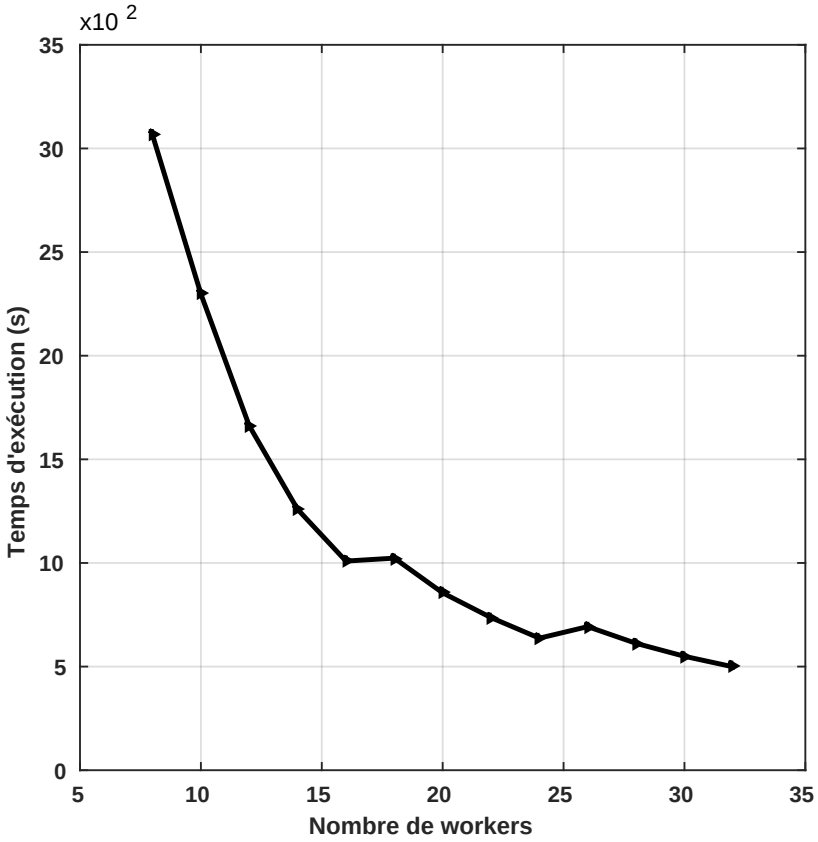


FIGURE 5.8 – Temps d'exécution de la partie calcul de la matrice d'impédance en fonction du nombre de workers pour un réseau 16 × 16 et 38816 arêtes.

La figure 5.9 présente les coefficients de réflexion en fonction de la fréquence pour quatre éléments pris arbitrairement parmi les éléments du réseau. Nous remarquons que presque tous les éléments du réseau présentent un niveau de paramètre S bas (inférieur à $-10dB$) indiquant une résonance découplage à la fréquence $160MHz$.

Dans la figure 5.10, nous donnons le diagramme de rayonnement 2D du réseau d'antennes. L'augmentation du nombre d'éléments a permis d'augmenter le gain de l'antenne qui est passé de $20.2dB$ pour l'antenne 8×8 à $40dB$ pour l'antenne 16×16. Nous remarquons aussi une amélioration de largeur à mi-puissance (Half Power Beam Width, HPBW) qui passe de 20 deg approximativement à 9 deg pour les mêmes configurations.

Les résultats obtenus montrent que les performances recherchées de gain élevé et de forte directivité lors de l'augmentation du nombre d'éléments dans un réseau d'antennes sont satisfaites.

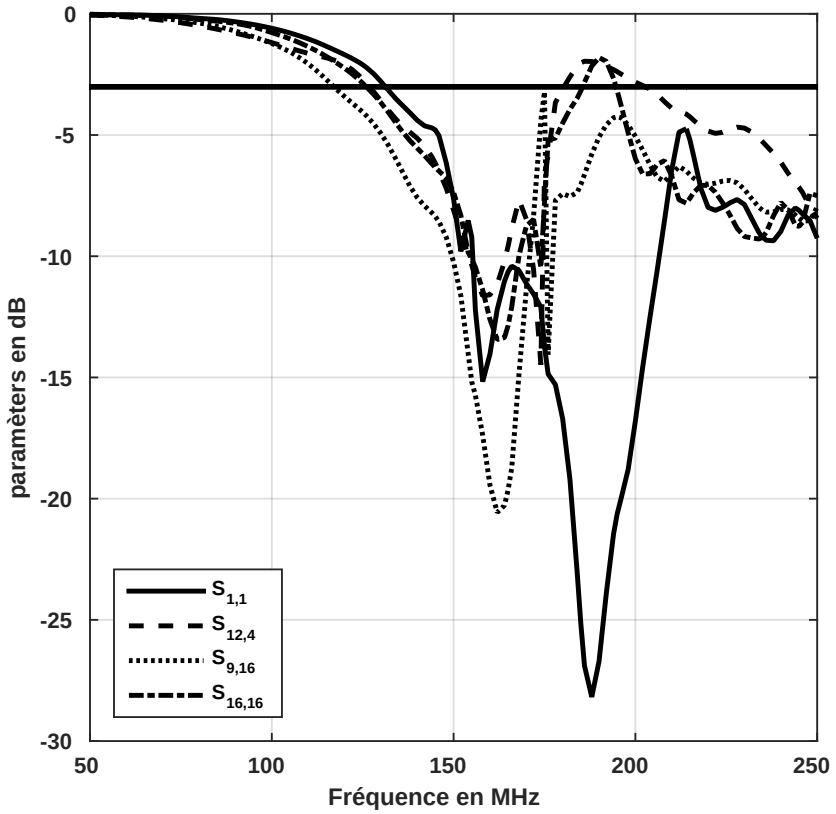


FIGURE 5.9 – Exemple de paramètres S pour quatre éléments en fonction de la fréquence d'un réseau Bowtie de 16 × 16 éléments

5.5 Étude de la scalabilty (évolutivité)

L'évolutivité d'un code parallèle est un critère très important qui garantie des résultats pour des configurations large impossible à simuler sur une seule machine, donc invérifiable par les méthodes classiques. Pour s'assurer que notre code MATLAB est bien évolutif, nous avons réaliser une expérience pour mesurer le comportement du code développé lorsque deux problèmes de taille différentes sont exécutés sur le cluster. Le premier problème n'est limité ni par la mémoire utilisé ni par le nombre de processeurs utilisés et de ce fait, nous pouvons mesurer les performances en utilisant un nombre de workers allant de 1 à 32 qui est le nombre maximum de workers disponibles. Le deuxième problème est par contre limité par la mémoire utilisé (memory-bounded), c'est à dire qu'il ne peut pas être exécuté sur un seul nœud.

Dans la première expérience, nous avons utilisé une configuration ayant 15041 arêtes dé-

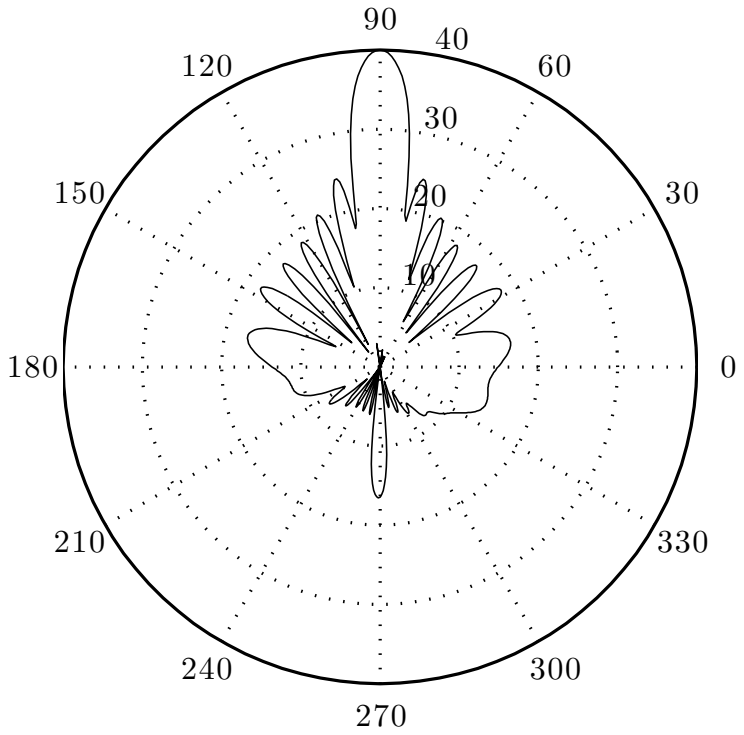


FIGURE 5.10 – Diagramme de rayonnement de l'antenne de 16 × 16 éléments à la fréquence 160Mhz

nommée (case1). Dans le deuxième cas (case2), nous avons utilisé une configuration avec 43421 arêtes correspondant au rang de la matrice impédance et dont le système linéaire correspondant ne peut être résolu que dans la cas où toute la mémoire disponible de tous les nœuds est utilisée.

Les figures 5.11, 5.12 et 5.13 représentent les temps d'exécutions exprimés en pourcentage du temps global des trois parties cité auparavant dans les deux cas : case1 et case2.

D'après ces figures, nous pouvons remarquer que les courbes des temps d'exécutions gardent la même allure dans les deux cas. La différence dans les pourcentage est petite et elle est due principalement à l'augmentation des communications inter-processus dans l'étape de la résolution du système linéaire. Nous pouvons conclure que la stratégie de parallélisation est évolutive et la scalabilité est bonne.

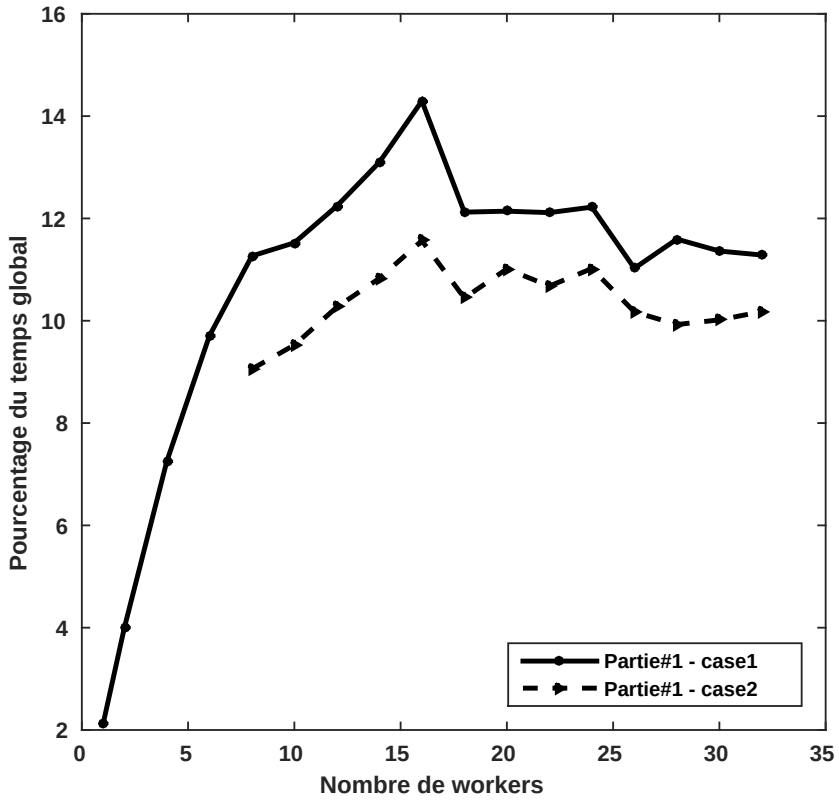


FIGURE 5.11 – Temps d'exécution de la partie#1 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers

5.6 Conclusion

Dans ce chapitre, nous avons proposé l'implémentation parallèle d'un code MATLAB pour l'analyse d'un réseau d'antennes bowtie. Malgré quelques difficultés rencontrées sur la manière d'optimiser les schémas de distribution des données, nous avons réussi à créer un code évolutif est relativement efficace qui a permis d'augmenter considérablement les possibilités de simulation des structures plus complexes. Nous avons réalisé plusieurs expériences sur notre cluster de calcul et ses performances ont été mesurées et analysées. Il serait intéressant de réaliser ces expériences sur des machines plus grande avec des réseaux hautes vitesses (InfiniBand ou MyriNet) pour voir le comportement du code ainsi développé.

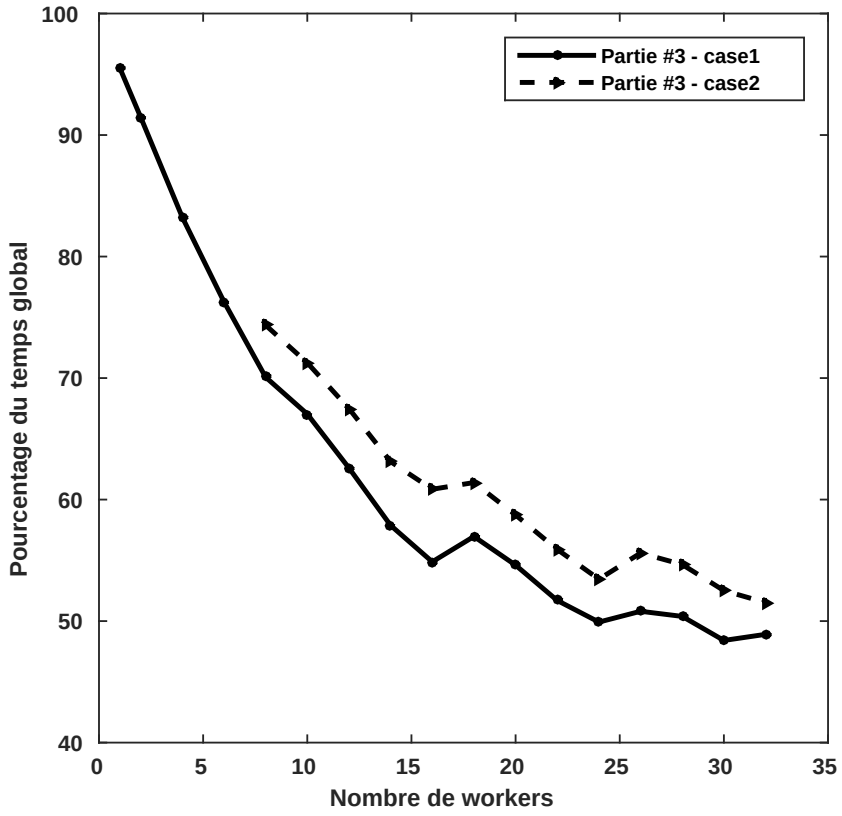


FIGURE 5.12 – Temps d'exécution de la partie#2 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers

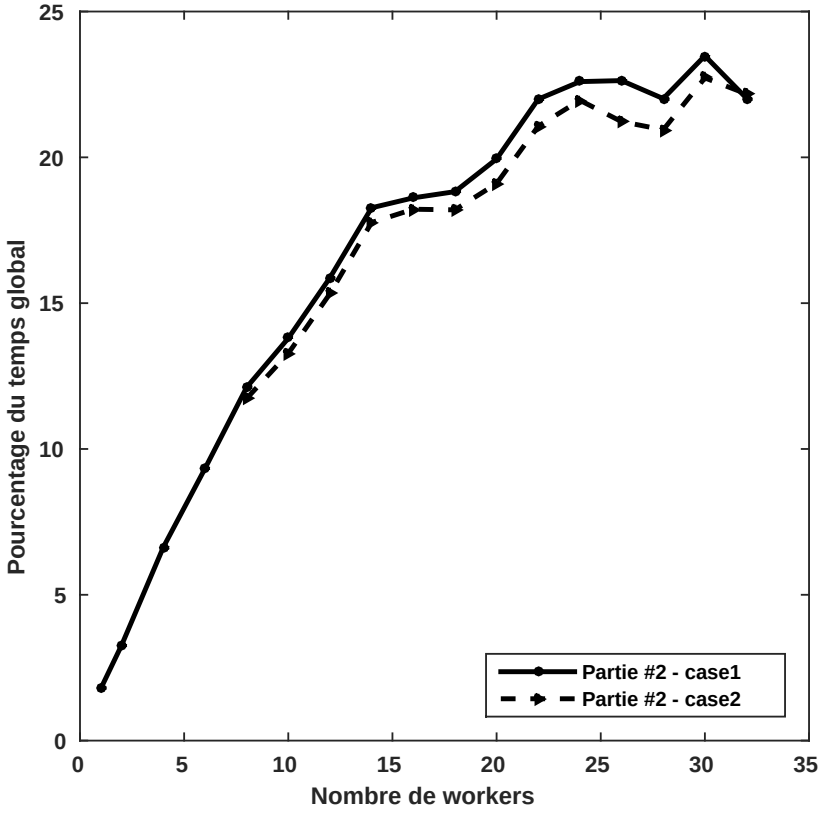


FIGURE 5.13 – Temps d'exécution de la partie#3 des case1 et case2 en pourcentage du temps global en fonction du nombre de workers

Conclusion Générale

Dans cette thèse, nous nous sommes intéressés à l'application des paradigmes de la programmation parallèle aux calculs des problèmes électromagnétiques en utilisant un cluster Beowulf. Plusieurs approches ont été utilisées, suivant les problèmes et les performances recherchées.

Dans le chapitre 2, nous avons discuter de la procédure de conception d'un cluster de calcul Beowulf utilisant un système open source de type Linux nommé ROCKS. Cette solution simplifie grandement la tâche de maintenance, de mise à jour et de déploiement d'un cluster. L'ajout et la suppression des nœuds de calcul est transparente et avec une configuration adéquate, il est possible de rendre les programmes utilisateurs moins dépendant de l'architecture du cluster.

Dans le chapitre 3, nous avons effectuer le développement d'une application de caractérisation d'une antenne microstrip circulaire multicouches. En premier lieu, le programme séquentiel a été développé et toutes les techniques d'optimisation dont l'utilisation des opérateurs vectoriels ont été appliquées pour assurer une comparaison correcte avec le code parallèle. Nous avons ensuite procéder à la parallélisation du code séquentiel en utilisant le modèle SPMD où nous avons pu observer une légère augmentation de la complexité du code mais qui reste marginale par rapport au gain constaté.

Le chapitre 4 est consacré au développement d'une méthode systématique pour générer et calculer les fonctions de formes d'ordre supérieur de la méthode FEM sous format symbolique. Nous nous sommes servi du moteur de calcul symbolique de MATLAB pour effectuer les calculs. Le code parallèle développé présente des bonnes caractéristiques d'extensibilité et un gain substantiel en terme de temps d'exécution.

Dans le chapitre 5, nous avons paralléliser un code de calcul des caractéristiques d'un réseau d'antenne bowtie, principalement l'impédance d'entrée, le gain et le diagramme de

rayonnement en utilisant la méthode des moments basé les éléments RWG. Il a été montré que cette solution permet d'augmenter le nombre d'élément du réseau par un facteur de 4 et le pas de discrétisation par un facteur de 6. Ainsi, il est possible de simuler des structures complexes nécessitant une grande quantité de mémoire RAM pour stocker et résoudre le système linéaire ainsi obtenu.

Parmi les perspectives, nous pouvons citer le travail sur des architectures à base d'unité de calcul graphique ou en anglais graphical processing unit GPU. Cependant, il faut tenir compte des limitations qui caractérise les GPU en terme de performances et les communications entre GPU et le CPU ce qui rend ce travail complexe étant donné les calculs nécessaires pour les méthodes numériques en électromagnétisme.

Nous pouvons aussi citer le développement dans un langage non propriétaire (C/C++, Octave, Scilab, Python, etc), parce que chaque noeud de calcul nécessite une licence à part de MATLAB, ce qui dépasse de loin les capacité d'un chercheur ou d'une unité de recherche.

Annexe A

L'interface de passage de message MPI

1.1 Introduction

Dans cet annexe, nous allons discuter d'un certain nombre de fonctions et de techniques de programmation très importantes. Une application MPI peut être vue comme une collection de tâches concurrentes en communication. Un programme MPI comprend du code écrit par le programmeur d'application qui sera lié à une bibliothèque de fonctions fournie par l'implémentation MPI. Chaque tâche se voit attribuer un rang unique dans un certain contexte : un nombre entier compris entre 0 et $n - 1$ pour une application MPI composée de n tâches. Ces rangs sont utilisés par les tâches MPI pour s'identifier mutuellement dans l'envoi et la réception de messages, pour exécuter des opérations collectives et pour coopérer en général. Les tâches MPI peuvent s'exécuter simultanément sur le même processeur ou sur des processeurs différents.

1.2 Les communicateurs

Une exigence importante dans tous les systèmes de transmission de messages est de garantir un espace de communication sûr dans lequel les messages non liés sont séparés les uns des autres. Par exemple, les messages de bibliothèque peuvent être envoyés et reçus sans interférence d'autres messages générés dans le système.

Le concept de communicateur est introduit dans MPI pour atteindre cette exigence de communication sécurisée. Un communicateur peut être considéré comme une liaison d'un contexte de communication à un groupe de tâches. Un communicateur est un objet ac-

cessible via un descripteur de type `MPI_COMM`. Les communicateurs peuvent être classés en intra-communicateurs pour des opérations au sein d'un même groupe de tâches, et en inter-communicateurs pour des opérations entre différents groupes de tâches. Lorsqu'une application MPI est lancée, toutes les tâches sont associées à un communicateur « `WORLD` ». Lorsqu'un nouveau contexte est nécessaire, Un programme effectue un appel de synchronisation pour dériver le nouveau contexte du contexte existant.

A tout instant, on peut connaître

— Le nombre de processus gérés par un communicateur donné

— Le rang du processus courant dans un communicateur donné

1.2.1 Groupes de tâches

Les tâches dans MPI peuvent appartenir à des groupes nommés. Un groupe est un objet accessible via un descripteur du type prédéfini `MPI_Group`. Les groupes de tâches fournissent des contextes dans lesquels les opérations MPI peuvent être limitées aux seuls membres d'un groupe particulier. Les membres d'un groupe se voient attribuer des identifiants uniques au sein du groupe appelé `ranks`. Un groupe est un ensemble ordonné de rangs, contigus et démarrant à partir de zéro. Le tableau (A.1) liste les fonctions relatives à la gestion des groupes dans un programme MPI.

1.2.2 Communicateur par défaut

MPI fournit le communicateur prédéfini `MPI_COMM_WORLD` en tant que communicateur par défaut. Une fois `MPI_Init()` appelé, ce communicateur définit un contexte unique incluant l'ensemble des tâches MPI disponibles pour le calcul. Le communicateur `MPI_COMM_WORLD` a la même valeur dans tous les processus et ne peut pas être modifié pendant la durée de vie d'une tâche. MPI fournit également le communicateur prédéfini `MPI_COMM_SELF`, qui inclut uniquement le processus appelant lui-même.

1.2.3 Rang des tâches

Les tâches associées à un communicateur se voient attribuer des identifiants entiers consécutifs commençant par zéro. Ces identifiants, appelés rangs (rank en anglais), sont utilisés pour distinguer les différentes tâches au sein du même groupe. Par exemple, des tâches de

TABLE A.1 – Liste des fonctions de gestion des groupes MPI.

Fonction	Description
`MPI_Group_compare`	Compare deux groupes
`MPI_Group_difference`	Crée un groupe à partir de la différence de deux groupes
`MPI_Group_excl`	Produit un groupe en réorganisant un groupe existant et en ne prenant que les membres non listés
`MPI_Group_free`	Libère un groupe
`MPI_Group_incl`	Produit un groupe en réorganisant un groupe existant et en ne prenant que les membres listés
`MPI_Group_intersection`	Produit un groupe comme l'intersection de deux groupes existants
`MPI_Group_range_excl`	Produit un groupe en excluant des plages de processus d'un groupe existant
`MPI_Group_range_incl`	Produit un groupe à partir de plages de processus d'un groupe existant
`MPI_Group_rank`	Renvoie le rang d'un processus dans le groupe donné
`MPI_Group_size`	Renvoie la taille d'un groupe
`MPI_Group_translate_ranks`	translater les rangs des processus d'un groupe à ceux d'un autre groupe
`MPI_Group_union`	Produit un groupe en combinant deux groupes

rangs différents peuvent être attribuées des travaux différents. Une tâche peut connaître son rang dans un communicateur en appelant la fonction `MPI_Comm_rank()` comme suit :

```
MPI_Comm communicator;//descripteur d'un communicateur
int my_rank;// variable qui doit contenir le rang de la tache
MPI_Comm_rank(communicator, &my_rank);
```

1.2.4 Groupe de communicateurs

Le groupe associé à un communicateur peut être récupéré en utilisant la fonction `MPI_Comm_group()`. Cette fonction prend un communicateur existant et renvoie son groupe de tâches correspondant. Puisque MPI ne fournit pas de fonctions pour créer des groupes à partir de zéro, `MPI_Comm_group()` est important pour créer un groupe de base à partir de laquelle d'autres groupes peuvent être formés. Le format de cette fonction est donné comme suit :

```
MPI_Comm communicator; // descripteur de communicateur
MPI_Group corresponding_group;// descripteur de groupe
MPI_Comm_group(communicator, &corresponding_group)
```

La taille du groupe associé à un communicateur peut être déterminée en appelant la fonction `MPI_Comm_size()`. Cette fonction prend un communicateur existant et renvoie la taille de son groupe correspondant comme suit :

```
MPI_Comm communicator;// descripteur de communicateur
int number_of_tasks;
MPI_Comm_size(communicator, &number_of_tasks)
```

1.3 Topologie virtuelle

Les schémas de communication logiques entre les tâches peuvent ne pas être reflétés adéquatement en utilisant la numérotation linéaire des tâches dans un groupe. Parfois, il serait logique d'arranger les tâches dans le groupe associé à un communicateur sous différentes formes telles que des grilles bidimensionnelles ou tridimensionnelles, par exemple.

En plus de lier un contexte à un groupe de tâches, un communicateur peut être associé à une topologie. La topologie peut être utilisée pour associer certains schémas d'adressage aux tâches au sein d'un groupe. Il existe deux types de topologies virtuelles dans MPI : la topologie cartésienne et la topologie en graphe.

1.3.1 La topologie cartésienne

La fonction collective `MPI_Cart_create()` montrée ci-dessous peut être utilisée pour créer des structures cartésiennes de dimension arbitraire. elle reçoit en entrée les paramètres suivants et renvoie un descripteur de communicateur avec la nouvelle structure cartésienne :

- Un communicateur existant, qui définit l'ensemble des tâches sur lesquelles la topologie doit être mappée.
- Le nombre de dimensions dans la structure cartésienne.
- La taille de chaque dimension
- Si la structure est périodique ou non en chaque dimension.
- Si le système est autorisé ou non à optimiser le mappage de la topologie virtuelle sur les processeurs physiques sous-jacents. Cela peut entraîner une modification de la numérotation des tâches d'origine dans le groupe associé au communicateur existant :

Par exemple, supposons qu'une application MPI a été lancée avec six tâches : $\{T0, T1, T2, T3, T4, T5\}$ ayant des rangs $\{0, 1, 2, 3, 4, 5\}$, respectivement. Au début, les six tâches sont référencés par le communicateur `MPI_COMM_WORLD`. Maintenant, supposons que nous voulons associer une structure en grille 2×3 avec les tâches dans `MPI_COMM_WORLD`. Il suffit de créer un nouveau communicateur `gridcomm` comme indiqué ci-dessous. Pour empêcher le système à ne pas modifier la numérotation des des tâches, nous devons mettre la variable `mapping` à `false`.

```
MPI_Comm gridcomm;// nouveau communicateur
int sizeofdims[2];
int periods[2];
int mapping = 0;
sizeofdims[0] = 2;
sizeofdims[1] = 3;
periods[0] = periods[1] = 0;
MPI_Cart_create(MPI_WORLD_COMM, 2, sizeofdims, periods, mapping, gridcomm);
```

MPI fournit des fonctions permettant à une tâche de connaître son rang et ses coordonnées dans la structure cartésienne.

1.3.2 Topologie en graphe

La fonction `MPI_Graph_create()` montrée ci-dessous peut être utilisée pour créer un nouveau communicateur auquel une information de topologie en graphe est attachée. Il prend en entrée les paramètres suivants et retourne un descripteur d'un communicateur avec la nouvelle topologie en graphe.

— Un communicateur existant, qui définit l'ensemble des tâches sur lesquelles la topologie doit être mappée.

— Le nombre de nœuds dans le graphe.

— Informations sur le degré (nombre de voisins) de chaque nœud.

— Les arêtes dans le graphe.

— Si le système est autorisé ou non à optimiser le mappage de la topologie virtuelle sur les processeurs physiques sous-jacents. Cela peut entraîner une modification de la numérotation des tâches d'origine dans le groupe associé au communicateur existant.

Par exemple, supposons qu'une application MPI a été lancée avec six tâches : $\{T0, T1, T2, T3, T4, T5\}$ ayant des rangs $\{0, 1, 2, 3, 4, 5\}$, respectivement. Au début, les six tâches sont référencés par

le communicateur `MPI_COMM_WORLD`. Maintenant, supposons que nous voulons associer une structure en graphe représentée dans la figure (A.1).

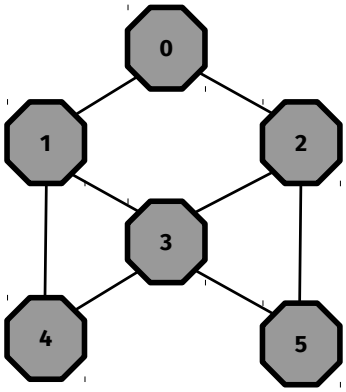


FIGURE A.1 – Graphe de l'exemple étudié.

Le segment de code suivant montre comment associer le graphe de la figure (A.1) au nouveau communicateur `graphcomm` en utilisant la fonction `MPI_Graph_create()` :

```
MPI_Comm graphcomm;// nouveau communicateur
int nnodes = 6;
int index[6] = {2,5,8,12,14,16};
int edges[16] = {1,2,0,3,4,0,3,5,1,2,4,5,1,3,2,3};
int mapping = 0;
MPI_Graph_create(MPI_WORLD_COMM, nnodes, index, edges,mapping, graphcomm)
```

1.4 Les communications MPI

La communication entre les tâches MPI est basée sur le paradigme de transmission de message. MPI utilise un riche ensemble de fonctions pour envoyer et recevoir des messages. La communication entre deux tâches implique les composants suivants :

1. Émetteur, généralement identifié par son rang.
2. Récepteur, généralement identifié par son rang.
3. Données de message
4. Étiquette de message, qui permet de gérer plusieurs messages entre deux tâches dans l'ordre.
5. Communicateur, qui fournit un contexte pour la communication.

1.4.1 Communications point à point

La communication point à point est une communication entre deux processus. L'un d'eux envoie un message (c'est l'émetteur), l'autre le reçoit (c'est le récepteur). Ce message doit contenir un certain nombre d'informations pour assurer une bonne réception et interprétation par le récepteur, à savoir :

- le rang du processus émetteur (`rang_proc_source`),
- le rang du processus récepteur (`rang_proc_dest`),
- l'étiquette du message (`tag_emis` pour message émis, `tag_recu` pour message reçu),
- le nom du communicateur (`comm`),
- le type des données échangées (`type_emis` pour message émis, `type_recu` pour message reçu),
- le nom données échangées (`val_emis` pour message émis, `val_recu` pour message reçu).
- la taille des données échangées (`taille_emis` pour message émis, `taille_recu` pour message reçu) (scalaire, vecteur, matrice, ...).

Plusieurs modes de transfert sont possibles pour échanger des messages. Les fonctions de base pour envoyer et recevoir des messages dans MPI sont l'envoi et la réception en mode bloquant. Dans ce mode, l'exécution est suspendue jusqu'à ce que le message soit reçu convenablement.

Envoi standard La fonction `MPI_Send` est la fonction d'envoi standard MPI.

```
MPI_Send(const void *buf, int count, MPI_Datatype datatype, int dest, int
tag, MPI_Comm comm)
```

Réception standard La fonction `MPI_Recv` est la fonction de réception en mode bloquant standard.

```
 MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int
tag, MPI_Comm comm, MPI_Status *status)
```

Envoi bufferisé (B) Dans l'envoi standard décrit ci-dessus, un message sortant peut être mis dans un tampon ou non avant l'émission en fonction de la décision du système MPI. En utilisant le mode bufferisé, la mise en mémoire tampon des messages est garantie.

```
MPI_Bsend(const void *buf, int count, MPI_Datatype datatype, int dest,
int tag, MPI_Comm comm)
```

Envoi synchrone (S) La communication synchrone peut être accomplie si l'émetteur et le récepteur sont en mode bloquant. Mais, puisque la réception standard est par défaut bloquante, il suffit alors d'un envoi bloquant pour pouvoir accomplir une communication synchrone. MPI fournit la fonction `MPI_Ssend` pour cet objectif.

```
MPI_Ssend(const void *buf, int count, MPI_Datatype datatype, int dest,
int tag, MPI_Comm comm)
```

1.4.2 Les communications en mode non bloquant

MPI prend en charge la communication non bloquante dans laquelle une tâche peut commencer une opération d'envoi ou de réception, effectuer un autre travail, puis revenir pour vérifier l'état de fin de l'opération du transfert. Ces modes d'envoi/réception non bloquante peuvent être utilisées de manière interchangeable avec les modes normales d'émission/réception et il n'est pas obligatoire d'associer les opérations de même type.

L'utilisation des fonctions d'envoi (réception) non bloquantes peut être accomplie en trois étapes :

1. Lancez un envoi (réception) en utilisant `MPI_Isend()` (`MPI_Irecv()`).
2. Effectuer des calculs pendant le temps de communication.
3. Terminez la communication en utilisant `MPI_Wait()` et `MPI_Test()`.

L'initialisation d'une opération d'envoi en mode standard peut être effectuée en utilisant la fonction :

```
MPI_ISend(buf, count, datatype, to_whom, tag, comm, &request)
```

De même, l'initialisation d'une opération de réception non bloquante peut être effectuée à l'aide de la fonction suivante :

```
MPI_Irecv(buf, count, data_type, from_whom, tag, comm, &request)
```

MPI fournit des fonctions permettant de tester la fin des opérations de communications non bloquantes qui ont été initiées. L'achèvement d'une opération de réception non bloquante indique que les données ont déjà été placées dans le tampon de réception et que le récepteur est prêt à y accéder.

La fonction suivante est utilisée pour vérifier l'achèvement d'une opération de communication :

```
MPI_Test(request, &flag, &status)
```

Le paramètre `flag` sera positionné à `true` si l'opération est terminée et `false`. Le paramètre `status` renverra des informations supplémentaires sur l'état du transfert.

La fonction suivante attendra la fin d'une opération de communication :

```
MPI_Wait(request, &status)
```

1.4.3 Synchronisation

Les constructions de synchronisation sont utilisées pour forcer un certain ordre d'exécution parmi les activités des tâches parallèles. Dans certains cas, des tâches parallèles doivent être synchronisées entre elles à un moment donné au cours de l'exécution. Les membres d'un groupe peuvent avoir besoin d'attendre à un point de synchronisation jusqu'à ce que toutes les tâches atteignent le même point. En plus des opérations bloquante qui représente une synchronisation par passage de message, on peut réaliser une synchronisation par l'utilisation de barrière.

Les tâches d'un groupe peuvent se synchroniser à un point de synchronisation à l'aide d'une barrière. L'exécution est suspendue au niveau de la barrière jusqu'à ce que toutes les tâches atteignent la barrière (Figure A.2). Le groupe peut inclure toutes les tâches ou seulement un sous-ensemble des tâches en fonction du communicateur. La construction `MPI_Barrier()` prend un communicateur en entrée comme suit :

```
MPI_Barrier (communicateur)
```

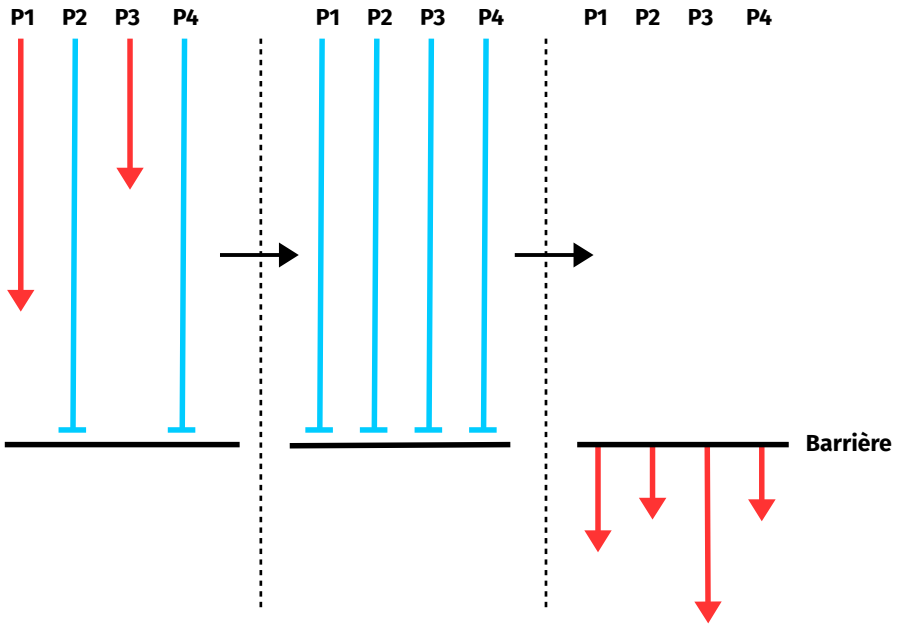


FIGURE A.2 – Synchronisation globale : `MPI_BARRIER()`.

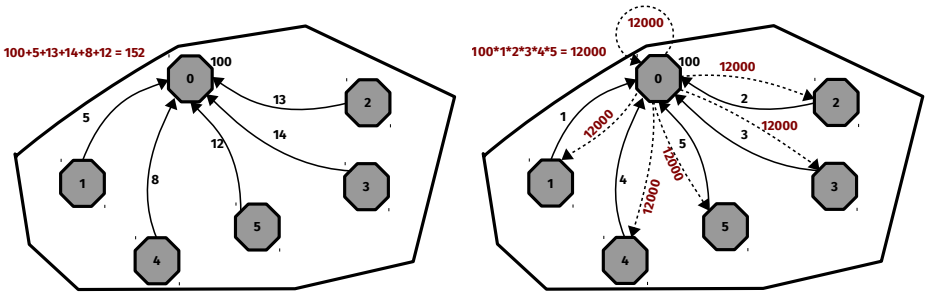
1.4.4 Les communications collectives

Les opérations collectives dans MPI sont les opérations qui sont appliquées à tous les membres du groupe d'un communicateur. Une opération collective est généralement définie en termes de groupe de tâches. L'opération est exécutée lorsque toutes les tâches du groupe appellent une fonction collective avec les paramètres correspondants. Il existe trois types d'opérations collectives : le contrôle des tâches, le calcul global et le transfert de données. La fonction de barrière `MPI_Barrier()` qui a été discutée dans la section précédente peut être classée comme une opération collective de contrôle de tâches.

Réduction répartie

Une réduction est une opération appliquée à un ensemble d'éléments pour en obtenir une seule valeur. Des exemples typiques sont la somme des éléments d'un vecteur $SUM(A(:))$ ou la recherche de l'élément de valeur maximum dans un vecteur $MAX(V(:))$.

MPI propose des fonctions pour opérer des réductions sur des données réparties sur un ensemble de processus. Le résultat est obtenu sur un seul processus (`MPI_REDUCE()`) (Figure A.3) ou bien sur tous les processus (`MPI_ALLREDUCE()` (Figure A.3), qui est en fait équivalent à un `MPI_REDUCE()` suivi d'un `MPI_BCAST()`).



(a) MPI_Reduce() (b) MPI_AllReduce()

FIGURE A.3 – Réduction répartie `MPI_Reduce()` avec l'opérateur somme et `MPI_AllReduce()` avec l'opérateur produit

Les opérations de transfert de données

MPI prend en charge une grande variété de fonctions collectives de transfert de données. Les opérations de base supportées sont la diffusion globale (*broadcast*), la diffusion sélective (*scatter*) et la collecte de données (*gather*). Dans une diffusion globale, un processus

envoie le même message à tous les membres du groupe. Une opération de diffusion sélective permet à un processus d'envoyer un message différent à chaque membre. Dans l'opération de collecte des données (gather), qui est l'opération duelle de la diffusion, un processus reçoit un message de chaque membre du groupe. Ces opérations de base peuvent être combinées pour former des opérations plus complexes.

Diffusion Globale

MPI fournit la fonction `MPI_BCAST()` pour diffuser un message du processus racine à tout les processus du groupe du communicateur (Figure A.4) :

```
MPI_BCAST (buf,count, data_type, root, comm, code)
```

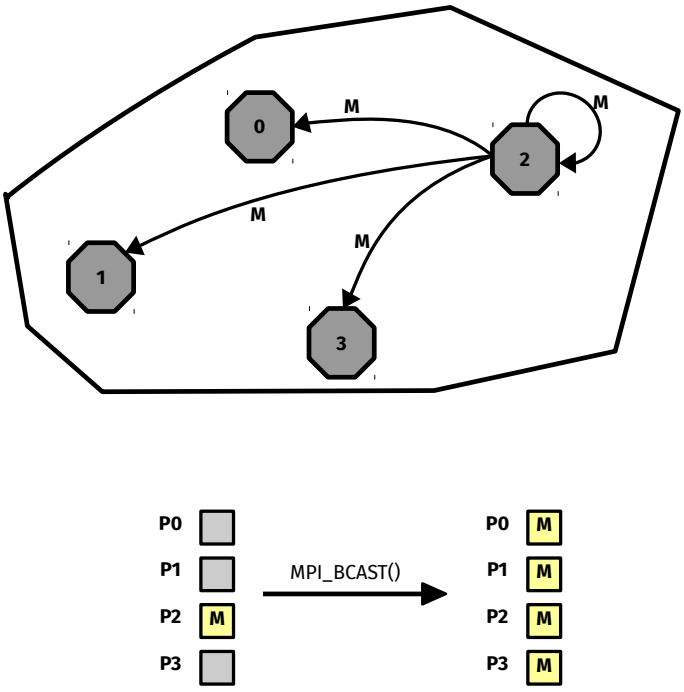


Figure A.4 – Diffusion générale : `MPI_BCAST()`.

Diffusion sélective

MPI fournit la fonction `MPI_SCATER()` pour diffuser un message divisé en tranche du processus racine à tout les processus du groupe du communicateur (Figure A.5) :

```
MPI_Scatter(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
    root, comm)
```

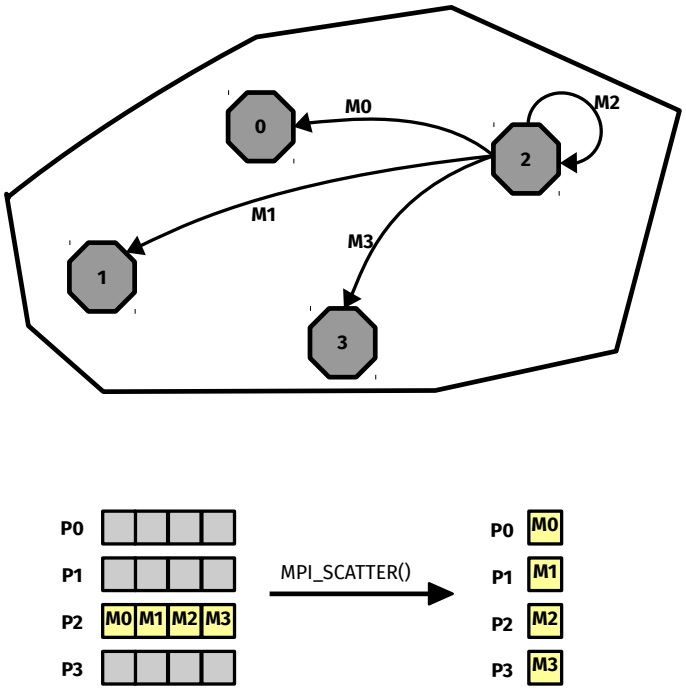


FIGURE A.5 – Diffusion sélective : `MPI_SCATTER()`.

Collecte

MPI fournit la fonction `MPI_GATHER()` pour collecter des messages à partir de tous les processus d'un groupe de communicateur dans un processus racine (Figure A.6) :

```
MPI_Gather(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
    root, comm)
```

En plus de `MPI_GATHER()`, MPI offre une fonction de collecte générale `MPI_ALLGATHER()` qui correspond à un `MPI_GATHER()` suivi d'un `MPI_BCAST()`.

```
MPI_AllGATHER(sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype,
    comm)
```

Pour la collecte des messages de tailles différentes à partir des processus d'un groupe décris par un communicateur, MPI offre une fonction de collecte générale `MPI_GATHERV()`.

```
MPI_GATHEREV(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
    recvtype, root, comm)
```

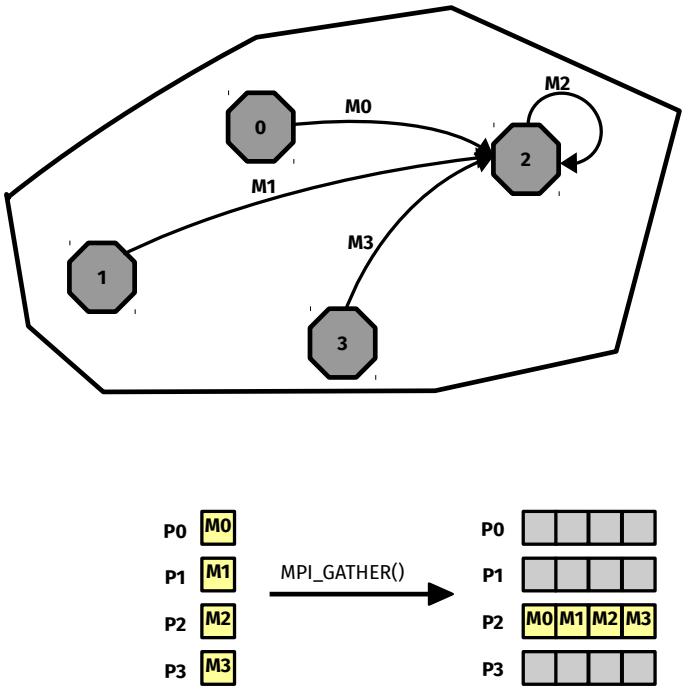


FIGURE A.6 – Collecte : `MPI_GATHER()`.

Annexe B

Rocks, une boite à outils open-source pour cluster réel et virtuel

2.1 Réalisation d'un cluster Beowulf

Un cluster Beowulf se compose principalement de deux composants matériels et deux composants logiciels. La configuration initiale d'un cluster n'est pas une tache facile, mais elle n'est pas non plus intenable. Un cluster Beowulf standard comprend au moins deux types de nœuds : nœud de connexion et nœud de calcul, donc l'homogénéité en fonction de la tache est déjà divisée [157].

2.1.1 Les parties Matérielles

La partie hardware comprend les nœuds de calcul et le réseau qui les interconnectent pour former un système unique. La figure B.1 montre une architecture de cluster Beowulf composée d'un nœud maître (Master node ou frontend) et un ensemble de nœuds de calcul. Le nœud maître sert de passerelle entre les nœuds restants et les utilisateurs. Les nœuds de calcul possèdent des systèmes d'exploitation minimaux et sont exclusivement dédiés au cluster.

2.1.2 Les parties logicielles

Les deux composants logiciels sont la collection d'outils utilisés pour développer des programmes d'application parallèles utilisateur et l'environnement logiciel pour la gestion des

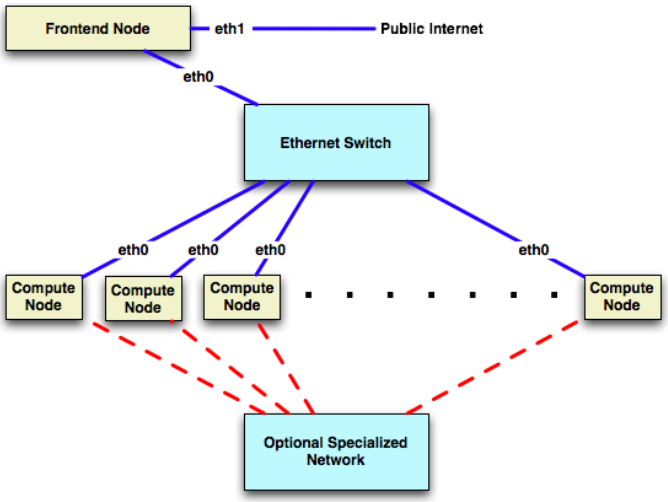


FIGURE B.1 – Diagramme de connexion des éléments d'un cluster -Rocks-

ressources du cluster.

2.2 La solution open source Rocks cluster

Rocks Cluster (initialement appelé NPACI Rocks) est une distribution Linux destinée aux grappes de calcul haute performance. Il a été lancé par National Partnership for Advanced Computational Infrastructure (NPACI) et le San Diego Supercomputer Center (SDSC) en 2000 et a été initialement financé en partie par une subvention de la National Science Foundation (NSF). Rocks était initialement basé sur la distribution RedHat Linux, mais les versions récentes de Rocks sont basées sur Community enterprise Operating System (CentOS), avec un installateur spécialement modifié qui simplifie l'installation de masse sur de nombreux ordinateurs. Rocks comprend de nombreux outils (tels que MPI) qui ne font pas partie de CentOS, mais sont des composants intégrés qui transforment un groupe d'ordinateurs en cluster de calcul haute performance. En plus, Rocks offre un service qui permet à un utilisateur de surveiller à distance le statut d'une installation en utilisant telnet [14].

2.2.1 Noeud principal

Le nœud principal est installé avec un pack logiciel standard largement utilisé pour le développement d'applications sur cluster et l'exécution des applications parallèles. Le HPC

Roll (un roll est un ensemble de bibliothèques, fichiers de configuration et de programmes qui forment un paquet logiciel) installé par défaut fournit des outils logiciels configurés qui peuvent être utilisés pour exécuter des applications parallèles sur le cluster (MPI, frameworks C/C++ et Fortran). On trouve aussi les bibliothèques, les compilateurs et les outils de benchmarking et de maintenance. Il est possible d'installer plusieurs logiciels de calcul scientifiques packagés spécialement pour Rocks. Cependant, il est possible d'installer n'importe quel logiciel avec la méthode classique avec un effort sur la configuration pour un environnement parallèle. A cet effet, nous avons installé le logiciel MATLAB et nous l'avons configuré pour être utilisé dans tous nos calculs parallèles.

2.2.2 Noeud de calcul

Les nœuds de calcul exécutent tous les travaux de calculs et doivent par conséquence avoir un système minimaliste sans toutefois nécessiter un gros travail pour leur installation et mise à jour. En plus, il est nécessaire que le déploiement d'un logiciel quelconque soit le plus simple indépendamment du nombre de nœuds.

Rocks utilise un outil nommé RedHat Kickstart qui permet de rendre l'installation des nœuds automatique à travers des fichiers de configurations utilisateur.

2.3 Réalisation du cluster Rocks-UBBA

2.3.1 Partie hardware

Nous avons construit un cluster composé d'un nœud principal (frontend) et 8 nœuds de calcul, chacun avec un processeur Intel Core i5 quad-core 3.1 GHz et 8 Go de RAM DDR3, totalisant 32 cœurs et 793.6 GFlops en puissance de calcul. Un commutateur Gigabit est utilisé pour interconnecter les nœuds du cluster (Figure B.2).

2.3.2 Installation du noeud principal

Le nœud principal ou frontend est un serveur avec deux interfaces réseau. L'interface publique est connectée à Internet et l'interface privée est connectée au réseau interne du cluster. Rocks considère la première interface (exemple :eth0) comme l'interface privée et la seconde (par exemple, eth1) comme l'interface publique.

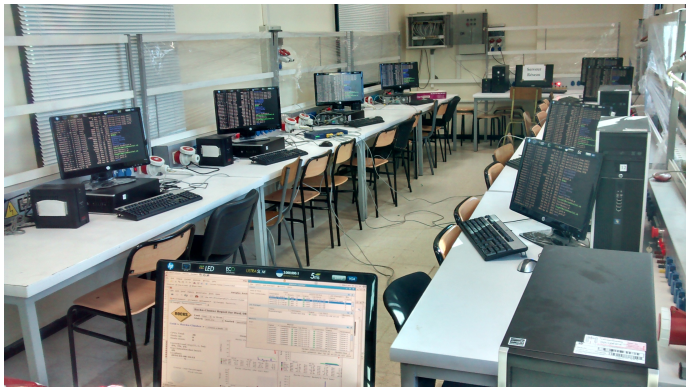


FIGURE B.2 – Photographie du cluster Rocks-BBA

La première étape de la procédure d'installation est l'installation du frontend. Ce dernier est installé avec un système basé sur une distribution open source Centos [1] et un ensemble d'outils pré-configurés pour la gestion et le déploiement d'un cluster HPC. Le système ainsi installé comprend un ensemble de bibliothèques et de compilateurs communément utilisés pour le développement et l'exécution des applications parallèles [14].

La procédure d'installation nécessite un peux plus d'étapes qu'une installation standard d'un poste de travail (Figures B.3).

Une fois l'installation terminée, Rocks crée une image système qui sera installé sur les nœuds de calcul.

2.3.3 Installation des nœuds de calcul

L'installation d'un nœud de calcul est réalisée en utilisant l'image crée sur le nœud principal. Un noeud de calcul doit émettre une requête DHCP qui sera capturé par le frontend exécutant l'utilitaire `insert-ether` (Figure B.4). Le frontend mettra à jour la base de donnée des nœuds de calcul (MAC, IP, HOSTNAME,...) et il va lancer l'installation du noeud en utilisant le protocole HTTPS et l'utilitaire `kicksatart`.

2.3.4 Ganglia un outil de surveillance

Rocks inclus Ganglia, un outil de surveillance distribué et évolutif pour les systèmes informatiques, les clusters et les réseaux haute performance. Le logiciel est utilisé pour visualiser les statistiques en direct ou enregistrées.

1. https://www.centos.org/

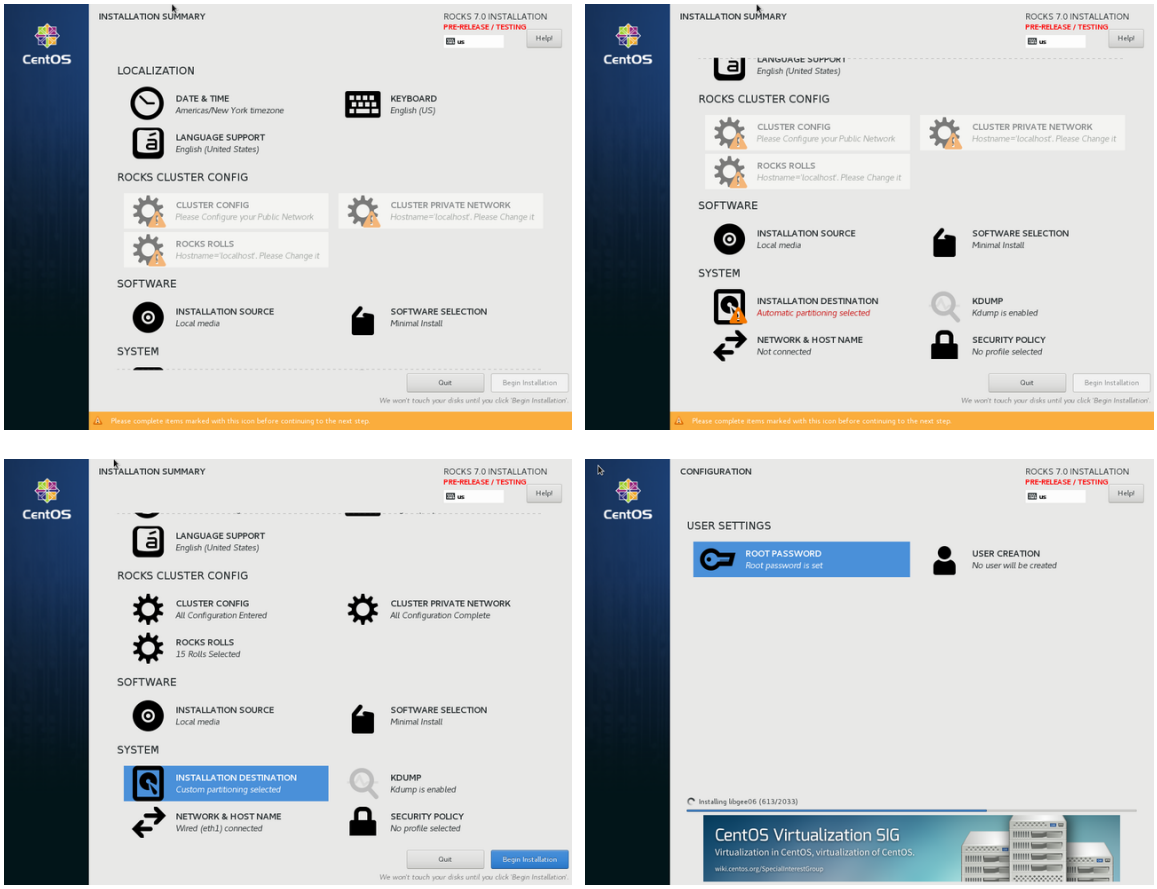
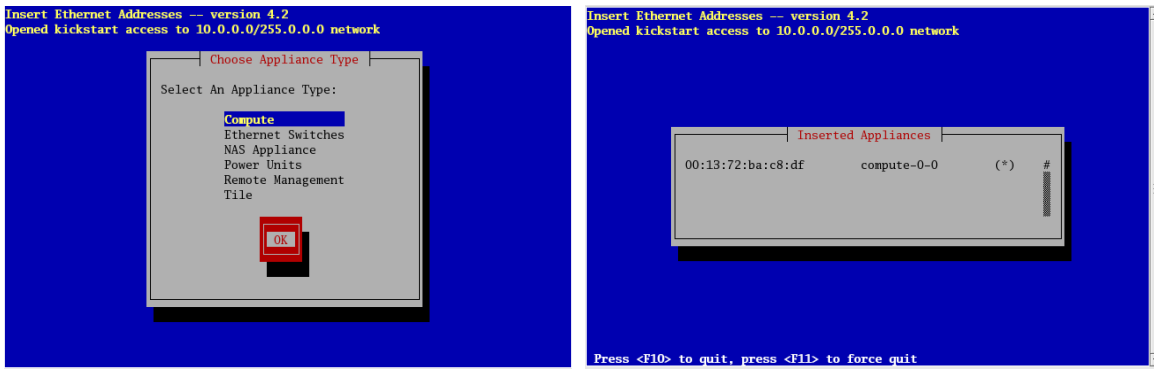


FIGURE B.3 – Capture d'écrans de la procédure d'installation du noeud principal sous Rocks



(a) Frontend en attente (b) Nœud de calcul inséré

FIGURE B.4 – Procédure de capture de la demande d'installation du nœud de calcul

L'interface web fournie une interface graphique aux informations de cluster en direct fournies par les moniteurs Ganglia s'exécutant sur chaque nœud de cluster. Les moniteurs collectent des valeurs pour diverses mesures telles que la charge du processeur, la mémoire

libre, l'utilisation du disque, les E/S réseau, la version du système d'exploitation, etc. Ces métriques sont envoyées via le réseau privé du cluster.

La figure (B.5) présente des graphes obtenus à partir de l'interface web de ganglia.

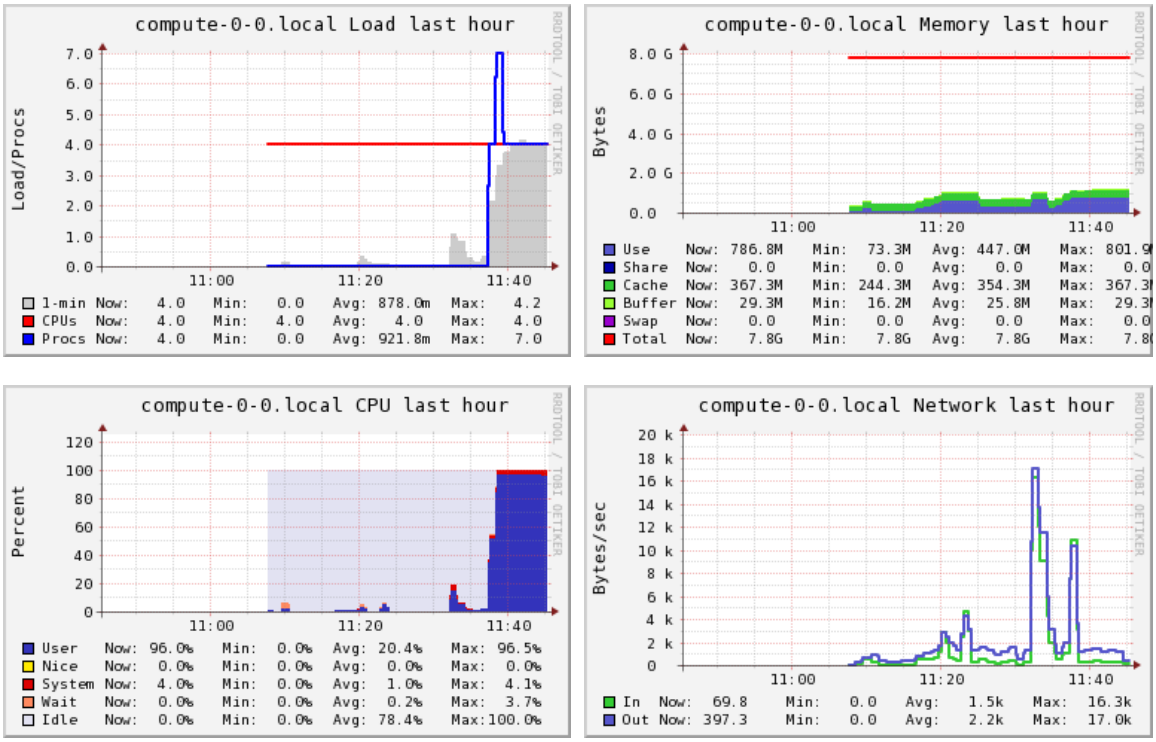


FIGURE B.5 – Exemples de graphes du rapport de surveillance d'un noeud de calcul tirés de Ganglia

Il est possible d'avoir des mesures en temps réel ou ceux enregistrés pour des périodes allant de une heure à plusieurs jours. Dans la figure (B.6), on donne un exemple des mesures de l'activité réseaux (messages MPI et autres).

2.4 Commandes utiles - administration cluster

La ligne de commande est l'interface administrative principale d'un Cluster Rocks. Rocks fournit un ensemble de commandes très riche qui peut être utilisé pour l'administration du système, la configuration et les rapports.

La ligne de commande suit une syntaxe simple pour accéder à l'information sur le cluster à partir de la base de données et l'utiliser pour définir les propriétés du cluster, définir l'état des nœuds, reporter l'état de l'installation du cluster et générer des fichiers de configuration

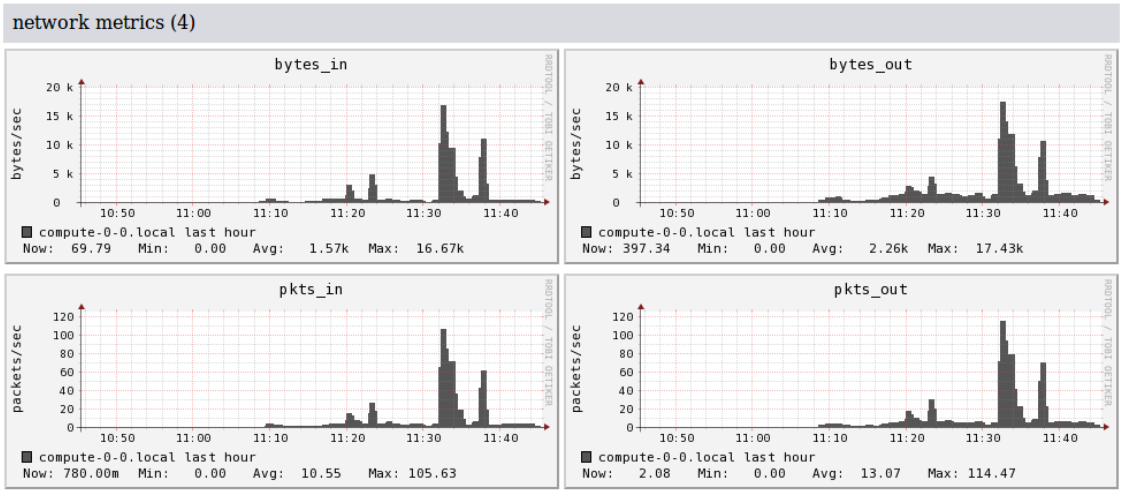


FIGURE B.6 – Exemple de mesures de l'activité réseaux d'un noeud de calcul

et système.

Le tableau (B.1) présente quelques commandes en ligne de commande pour l'administration d'un cluster Rocks.

TABLE B.1 – Quelques commandes en ligne de commande pour l'administration d'un cluster Rocks.

Commande	Description
`rocks list host`	Lister tous le nœuds dans la cluster
`rocks list host interface compute`	Lister les interfaces réseau des nœuds de calcul
`rocks list host attr compute-0-0`	Lister les attributs d'un noeud particulier (`compute-0-0`)
`rocks sync users`	Distribuer des informations des comptes utilisateurs à l'ensemble du cluster
`rocks run host compute "commande"`	Exécuter « commande » sur tous les nœuds de calcul
`rocks list host boot`	Définir un hôte (`compute-0-0`) pour une réinstallation au redémarrage
`rocks set host boot compute-0-0 action=install`	

Bibliographie

[1] T. Rylander, P. Ingelström et A. Bondeson : *Computational Electromagnetics*. Texts in Applied Mathematics. Springer New York, 2 édition, 2013. ISBN 978-1-4614-5350-5.

[2] A. Bondeson, T. Rylander et P. Ingelström : *Computational Electromagnetics*. Texts in Applied Mathematics. Springer New York, 2005. ISBN 9780387261584.

[3] Y. Zhang et T.K. Sarkar : *Parallel Solution of Integral Equation-Based EM Problems in the Frequency Domain*. Wiley Series in Microwave and Optical Engineering. Wiley, 2009. ISBN 9780470495087.

[4] P. Sumithra et D. Thiripurasundari : Review on computational electromagnetics. *Advanced Electromagnetics*, 6(1):42–55, march, 10th 2017.

[5] E.K. Miller : A selective survey of computational electromagnetics. *IEEE Transactions on Antennas and Propagation*, 36(9):1281–1305, Sep 1988. ISSN 0018-926X.

[6] R.F. Harrington : *Field Computation by Moment Methods*. Wiley-IEEE Press, 1993. ISBN 0780310144.

[7] K.S. Yee : Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14 (3):302–307, May 1966. ISSN 0018-926X.

[8] R. Courant : Variational methods for the solution of problems of equilibrium and vibrations. *BULL. AMER. MATH. SOC*, 49:1–23, 1943.

[9] W.C. Chew, E. Michielssen, J.M. Song et J.M. Jin, éditeurs. *Fast and Efficient Algorithms in Computational Electromagnetics*. Artech House, Inc., Norwood, MA, USA, 2001. ISBN 1580531520.

[10] D.B. Davidson : *Computational electromagnetics for RF and microwave engineering*. Cambridge University Press,, 2nd édition, 2011. ISBN 978-0-521-51891-8.

[11] M.N.O. Sadiku : *Numerical Techniques in Electromagnetics with MATLAB*. CRC Press, Inc., Boca Raton, FL, USA, third édition, 2009. ISBN 142006309X, 9781420063097.

[12] J.M. Jin et D.J. Riley : *Finite Element Analysis of Antennas and Arrays*. John Wiley and Sons, Inc., 2009. ISBN 9780470401286.

[13] W.C. Gibson : *The Method of Moments in Electromagnetics*. Chapman and Hall/CRC, 2008. ISBN 978-1-4200-6145-1.

[14] P.M. Papadopoulos, M.J. Katz et G. Bruno : NPACI rocks : tools and techniques for easily deploying manageable Linux clusters. *In Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 258–267, Oct 2001.

[15] T.E. Anderson, D.E. Culler et D.A. Patterson : A case for NOW (networks of workstations). *IEEE Micro*, 15(1):54–64, février 1995.

[16] T. Sterling, D.J. Becker, D. Savarese, J.E. Dorband, U.A. Ranawake et C.V. Packer : Beowulf : A parallel workstation for scientific computation. *In In Proceedings of the 24th International Conference on Parallel Processing*, pages 11–14. CRC Press, 1995.

[17] B. Jang, S. Do, H. Pien et D. Kaeli : Architecture-aware optimization targeting multithreaded stream computing. *In Proceedings of 2Nd Workshop on General Purpose Processing on Graphics Processing Units*, GPGPU-2, pages 62–70, New York, NY, USA, 2009. ACM.

[18] S. Ryoo, C.I. Rodrigues, S.S. Baghsorkhi, S.S. Stone, D.B. Kirk et W. Hwu : Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. *In Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '08, pages 73–82, New York, NY, USA, 2008. ACM.

[19] D.E. Culler, A. Gupta et J.P. Singh : *Parallel Computer Architecture : A Hardware/Software Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st édition, 1997. ISBN 1558603433.

[20] M.J. Flynn : Very high-speed computing systems. *Proceedings of the IEEE*, 54(12): 1901–1909, Dec 1966. ISSN 0018-9219.

[21] M.J. Flynn : Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, 21(9):948–960, septembre 1972.

[22] T. Rauber et G. Rünger : *Parallel programming : for multicore and cluster systems*. Springer-Verlag, Berlin, 2010.

[23] J.E. Thornton : The CDC 6600 project. *IEEE Annals of the History of Computing*, 2(4):338–348, 1980.

[24] T. Rauber et G. Rünger : *Parallel Programming For Multicore and Cluster Systems*. Springer, 2010. ISBN 978-3-642-04817-3.

[25] D.R. Butenhof : *Programming with POSIX thread*. Addison-Wesley, 1997. ISBN 0-201-63392-2.

[26] L. Dagum et R. Menon : OpenMP : an industry standard API for shared-memory programming. *IEEE Computational Science and Engineering*, 5(1):46–55, Jan 1998. ISSN 1070-9924.

[27] C. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald et R. Menon : *Parallel Programming in OpenMP*. Academic Press, 2001. ISBN 1-55860-671-8.

[28] J. Reinders : *Intel Threading Building Blocks : Outfitting C++ for Multi-Core Processor Parallelism*. O'Reilly, 2007. ISBN 978-0-596-51480-8.

[29] R.D. Blumofe, C.F. Joerg, B.C. Kuszmaul, C.E. Leiserson, K.H. Randall et Y. Zhou : Cilk : An efficient multithreaded runtime system. *Journal of Parallel and Distributed Computing*, 37(1):55–69, 1996. ISSN 0743-7315.

[30] M. Frigo, C.E. Leiserson et K.H. Randall : The implementation of the Cilk-5 multithreaded language. *SIGPLAN Not.*, 33(5):212–223, may 1998. ISSN 0362-1340.

[31] V.S. Sunderam : PVM : A framework for parallel distributed computing. *Concurrency : Practice and Experience*, 2(4):315–339, 1990. ISSN 1096-9128.

[32] V.S Sunderam, G.A Geist, J. Dongarra et R. Manchek : The PVM concurrent computing system : Evolution, experiences, and trends. *Parallel Computing*, 20(4):531 – 545, 1994. ISSN 0167-8191.

[33] Message P Forum : MPI : A message-passing interface standard. Rapport technique, University of Tennessee, Knoxville, TN, USA, 1994.

[34] M. Martinasso : *Analyse et modélisation des communications concurentes dans les réseaux haute performance*. Thèse de doctorat, Université Joseph-Fourier, Grenoble I, France, 2007.

[35] W. Gropp, E. Lusk, N. Doss et A. Skjellum : A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22 (6):789–828, 1996. ISSN 0167-8191.

[36] W. Gropp : MPICH2 : A new start for MPI implementations. *In Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 7–, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44296-0.

[37] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham et T.S. Woodall : Open MPI : Goals, Concept, and Design of a Next Generation MPI Implementation. *In* D. Kranzlmüller, P. Kacsuk et J. Dongarra, éditeurs : *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 97–104, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30218-6.

[38] H. El-Rewini et M. Abd-El-Barr : *Advanced Computer Architecture and Parallel Processing*. John Wiley and Sons, Inc., 2005. ISBN 9780471218760.

[39] J.L. Baer : *Microprocessor Architecture : From Simple Pipelines to Chip Multiprocessors*. Cambridge University Press, New York, NY, USA, 1st édition, 2009. ISBN 0521769922,9780521769921.

[40] J.L. Hennessy et D.A. Patterson : *Computer Architecture : A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th édition, 2011. ISBN 012383872X, 9780123838728.

[41] G.M. Amdahl : Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the AFIPS conference proceedings, vol. 30 (atlantic city, n.j., apr. 18-20), afips press, reston, va., 1967, pp. 483-485, when dr. amdahl was at international business machines corporation, sunnyvale, california. *IEEE Solid-State Circuits Society Newsletter*, 12(3):19–20, Summer 2007. ISSN 1098-4232.

[42] A. Krishnamurthy, J. Nehrbass, J.C. Chaves et S. Samsi : Survey of parallel matlab techniques and applications to signal and image processing. *In Acoustics, Speech and Signal Processing, 2007. ICASSP 2007.*, volume 4, pages IV–1181–IV–1184, 2007.

[43] M. Goryawala, M.R. Guillen, R. Bhatt, A. Mcgoron et M. Adjouadi : A comparative study on the performance of the parallel and distributing computing operation in matlab. *In Advanced Information Networking and Applications (AINA)*, pages 150–157, April 2010.

[44] S. Leman : *Contribution à la Résolution de Problèmes de Compatibilité Électromagnétique par le Formalisme des Circuits Électriques de KRON*. Thèse de doctorat, Université des Sciences et Technologie de Lille, France, 2009.

[45] K. Yee : Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14 (3):302–307, May 1966. ISSN 0018-926X.

[46] C.A. Balanis : *Antenna Theory, Analysis and Design*. Wiley, fourth édition, 2016. ISBN 978-1-118-642060-1.

[47] George B. Arfken, Hans J. Weber et Frank E. Harris : *Mathematical Methods for Physicists*. Academic Press, sixth édition, juillet 2005. ISBN 0120598760.

[48] M.U. Tyn et L. Debnath : *Linear partial differential equations for scientists and engineers*. Birkhäuser, 4th édition, 2007.

[49] Jianming Jin : *The Finite Element Method in Electromagnetics*. Wiley-IEEE Press, 3rd édition, 2014. ISBN 1118571363, 9781118571361.

[50] A. Ishimaru : *Electromagnetic Wave Propagation, Radiation, and Scattering*. Prentice Hall, 1991. ISBN 9780132738712.

[51] D. Jones : A critique of the variational method in scattering problems. *IRE Transactions on Antennas and Propagation*, 4(3):297–301, July 1956. ISSN 0096-1973.

[52] A. Taflove et M.E. Brodwin : Numerical solution of steady-state electromagnetic scattering problems using the time-dependent maxwell's equations. *IEEE Transactions on Microwave Theory and Techniques*, 23(8):623–630, Aug 1975. ISSN 0018-9480.

[53] G. Mur : Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations. *IEEE Transactions on Electromagnetic Compatibility*, EMC-23(4):377–382, Nov 1981. ISSN 0018-9375.

[54] J.P. Berenger : A perfect matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114:185–200, 10 1994.

[55] J.B. Schneider : Understanding the finite-difference time-domain method, April 2017. URL www.eecs.wsu.edu/~schneidj/ufdtd.

[56] A. Taflove et S.C. Hagness : *Computational Electrodynamics : The Finite-difference Time-domain Method*. Numéro vol. 1 *in* Antennas and Propagation Library. Artech House, 2000. ISBN 9781580530767.

[57] M.N.O. Sadiku, S.O. Agbo et V. Bemmel : Stability criterion for finite-difference time-domain algorithm [EM theory]. *In IEEE Proceedings on Southeastcon*, volume 1, pages 48–50, Apr 1990.

[58] D.S. Katz, E.T. Thiele et A. Taflove : Validation and extension to three dimensions of the berenger PML absorbing boundary condition for FD-TD meshes. *IEEE Microwave and Guided Wave Letters*, 4(8):268–270, Aug 1994. ISSN 1051-8207.

[59] D.T. Prescott : Reflection analysis of FDTD boundary conditions. i. time-space absorbing boundaries. *IEEE Transactions on Microwave Theory and Techniques*, 45 (8):1162–1170, Aug 1997. ISSN 0018-9480.

[60] B.M. Irons : A frontal solution program for finite element analysis. *International Journal for Numerical Methods in Engineering*, 2(1):5–32, 1970. ISSN 1097-0207.

[61] J.W.H. Liu : The multifrontal method for sparse matrix solution : Theory and practice. *SIAM Review*, 34(1):82–109, 1992.

[62] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donuto, J. Dongurvu, V. Eijkhout, R. Pozo, C. Romine et H. Van der Vorst : *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994.

[63] T.A. Davis : *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006. ISBN 0898716136.

[64] T.A. Davis, S. Rajamanickam et W.M. Sid-Lakhdar : A survey of direct methods for sparse linear systems. *Acta Numerica*, 25:383–566, 2016.

[65] Intel math kernel library,developer reference, revision : 015, 2017. URL https://software.intel.com/en-us/mkl.

[66] A. Gupta : WSMP : Watson sparse matrix package, part i - direct solution of symmetric sparse systems. Rapport technique 21886 (98462), IBM T.J. Watson Research Center, Nov. 2000.

[67] A. Gupta, M. Joshi et V. Kumar : WSMP : A high-performance shared- and distributed-memory parallel sparse linear equation solver. Rapport technique 22038 (98932), IBM T.J. Watson Research Center, Apr. 2001.

[68] T.A. Davis : Algorithm 832 : UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, juin 2004. ISSN 0098-3500.

[69] SuiteSparse : a suite of sparse matrix software, 2017. URL http://faculty.cse.tamu.edu/davis/suitesparse.html.

[70] T.A. Davis : Algorithm 915, SuiteSparseQR : Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Trans. Math. Softw.*, 38(1):8 :22, décembre 2011. ISSN 0098-3500.

[71] MUMPS : a parallel sparse direct solver, 2017. URL http://mumps.enseeiht.fr/.

[72] X.S. Li, J.W. Demmel, J.R. Gilbert, iL. Grigori, M. Shao et I. Yamazaki : SuperLU Users' Guide. Rapport technique LBNL-44289, Lawrence Berkeley National Laboratory, September 1999.

[73] X.S. Li : An overview of SuperLU : Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software*, 31(3):302–325, September 2005.

[74] PaStiX : Parallel sparse matrix package, 2017. URL https://gitlab.inria.fr/solverstack/pastix.

[75] Pascal Hénon, Pierre Ramet et Jean Roman : PaStiX : A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing*, 28(2):301–321, 2002.

[76] Pierre Ramet : *Heterogeneous architectures, Hybrid methods, Hierarchical matrices for Sparse Linear Solvers*. Habilitation à diriger des recherches, Université de Bordeaux, Nov 2017.

[77] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, H. Zhang et H. Zhang : PETSc Web page. http://www.mcs.anl.gov/petsc, 2017. URL http://www.mcs.anl.gov/petsc.

[78] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang et H. Zhang : PETSc users manual. Rapport technique ANL-95/11 - Revision 3.8, Argonne National Laboratory, 2017. URL http://www.mcs.anl.gov/petsc.

[79] S. Balay, W.D. Gropp, L.C. McInnes et B.F. Smith : Efficient management of parallelism in object oriented numerical software libraries. *In* E. Arge, A. M. Bruaset et H. P. Langtangen, éditeurs : *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[80] Y. Saad : SPARSKIT : A Basic Tool-Kit for Sparse Matrix Computations, Version 2. Rapport technique LBNL-44289, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, June 1994. http://www-users.cs.umn.edu/~saad/software/SPARSKIT/.

[81] Joseph S. Shang : Computational electromagnetics. *ACM Comput. Surv.*, 28(1):97–99, mar 1996. ISSN 0360-0300.

[82] J.E. Patterson, T. Cwik, R.D. Ferraro, N. Jacobi, P.C. Liewer, T.G. Lockhart, G.A. Lyzenga, J.W. Parker et D.A. Simoni : Parallel computation applied to electromagnetic scattering and radiation analysis. *Electromagnetics*, 10(1-2):21–39, 1990.

[83] T. Cwik, J. Partee et J. Patterson : Method of moment solutions to scattering problems in a parallel processing environment. *IEEE Transactions on Magnetics*, 27 (5):3837–3840, Sept 1991. ISSN 0018-9464.

[84] D.B. Davidson : Large parallel processing revisited : a second tutorial. *IEEE Antennas and Propagation Magazine*, 34(5):9–21, Oct 1992. ISSN 1045-9243.

[85] T. Cwik, R. van de Geijn et J. Patterson : Application of massively parallel computation to integral equation models of electromagnetic scattering. *J. Opt. Soc. Am. A*, 11(4):1538–1545, Apr 1994.

[86] R.A. van de Geijn : *Using PLAPACK : Parallel Linear Algebra Package*. MIT Press, Cambridge, MA, USA, 1997. ISBN 0-262-72026-4.

[87] S.M Rao, D.R Wilton et D.R Glisson : Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Trans. Antennas and Propagation.*, 30(3):409–418, 1982.

[88] L. S. Blackford, J. Choi, A. Cleary, E. D'Azeuedo, J. Demmel, I. Dhillon, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker et R. C. Whaley : *ScaLAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. ISBN 0-89871-397-8.

[89] U. Jakobus : *Parallel Computation of Electromagnetic Fields Based on Integral Equations*, pages 377–386. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-642-58600-2.

[90] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek et V. Sunderam : *PVM : Parallel Virtual Machine : A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, MA, USA, 1994. ISBN 0-262-57108-0.

[91] E.F. Knott et T.B.A. Senior : Comparison of three high-frequency diffraction techniques. *Proceedings of the IEEE*, 62(11):1468–1474, Nov 1974. ISSN 0018-9219.

[92] J.S. Asvestas : The physical optics method in electromagnetic scattering. *Journal of Mathematical Physics*, 21(2):290–299, 1980. URL https://doi.org/10.1063/1.524413.

[93] T. Sarkar, E. Arvas et S. Rao : Application of FFT and the conjugate gradient method for the solution of electromagnetic radiation from electrically large and small conducting bodies. *IEEE Transactions on Antennas and Propagation*, 34(5):635–640, May 1986. ISSN 0018-926X.

[94] E. Bleszynski, M. Bleszynski et T. Jaroszewicz : AIM : Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems. *Radio Science*, 31(5):1225–1251, Sept 1996. ISSN 1944-799X.

[95] V. Rokhlin : Rapid solution of integral equations of scattering theory in two dimensions. *Journal of Computational Physics*, 86(2):414 – 439, 1990. ISSN 0021-9991.

[96] X.-C. Nie, L.-W. Li et N. Yuan : Precorrected-FFT algorithm for solving combined field integral equations in electromagnetic scattering. *Journal of Electromagnetic Waves and Applications*, 16(8):1171–1187, 2002.

[97] J. Song, C. Lu et W.C. Chew : Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Transactions on Antennas and Propagation*, 45(10):1488–1493, Oct 1997. ISSN 0018-926X.

[98] J.M. Song, C.C. Lu, W.C. Chew et S.W. Lee : Fast illinois solver code (FISC). *IEEE Antennas and Propagation Magazine*, 40(3):27–34, Jun 1998. ISSN 1045-9243.

[99] M.A. Stalzer : A parallel fast multipole method for the helmholtz equation. *Parallel Processing Letters*, 05(02):263–274, 1995.

[100] E.J.L. Lu et D.I. Okunbor : A massively parallel fast multipole algorithm in three dimensions. *In Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*, pages 40–48, Aug 1996.

[101] E.J.L. Lu et D.I. Okunbor : Parallel implementation of 3D FMA using MPI. *In MPI Developer's Conference, 1996. Proceedings., Second*, pages 119–124, Jul 1996.

[102] S.V. Velamparambil, J.M. Song, W.C. Chew et K. Gallivan : ScaleME : a portable scaleable multipole engine for electromagnetic and acoustic integral equation solvers. *In IEEE Antennas and Propagation Society International Symposium. 1998 Digest*, volume 3, pages 1774–1777 vol.3, June 1998.

[103] S.V. Velamparambil, J.E. Schutt-Aine, J.G. Nickel, Song J.M. et W.C. Chew : Solving large scale electromagnetic problems using a Linux cluster and parallel MLFMA. *In IEEE Antennas and Propagation Society International Symposium. 1999 Digest*, volume 1, pages 636–639 vol.1, July 1999.

[104] S. Velamparambil, Weng Cho Chew et M. L. Hastriter : Scalable electromagnetic scattering computations. *In IEEE Antennas and Propagation Society International Symposium*, volume 3, pages 176–, June 2002.

[105] S. Velamparambil, W.C. Chew et J. Song : 10 million unknowns : is it that big ? [computational electromagnetics]. *IEEE Antennas and Propagation Magazine*, 45 (2):43–58, April 2003. ISSN 1045-9243.

[106] S. Velamparambil et W.C. Chew : Analysis and performance of a distributed memory multilevel fast multipole algorithm. *IEEE Transactions on Antennas and Propagation*, 53(8):2719–2727, Aug 2005. ISSN 0018-926X.

[107] Martin Nilsson : A parallel shared memory implementation of the fast multipole method for electromagnetics. Rapport technique 2003-049, Department of Information Technology, Uppsala University, octobre 2003.

[108] C. Waltz, K. Sertel, M.A. Carr, B.C. Usner et J.L. Volakis : Massively parallel fast multipole method solutions of large electromagnetic scattering problems. *IEEE Transactions on Antennas and Propagation*, 55(6):1810–1816, June 2007. ISSN 0018-926X.

[109] E. D'Azevedo et J. Dongarra : The design and implementation of the parallel out-of-core ScaLAPACK LU, QR, and Cholesky factorization routines. *Concurrency : Practice and Experience*, 12(15):1481–1493, 2000. ISSN 1096-9128.

[110] E. Lezar et D.B. Davidson : GPU-accelerated method of moments by example : Monostatic scattering. *IEEE Antennas and Propagation Magazine*, 52(6):120–135, Dec 2010. ISSN 1045-9243.

[111] Y. Chen, Y. Zhang, Z. Lin, X. Zhao et S. Jiang : GPU accelerated parallel MoM for simulating microstrip antenna array. *In Proceedings of 2014 3rd Asia-Pacific Conference on Antennas and Propagation*, pages 1027–1029, July 2014.

[112] A. Schröder, H.D. Brüns et C. Schuster : Fast evaluation of electromagnetic fields using a parallelized adaptive cross approximation. *IEEE Transactions on Antennas and Propagation*, 62(5):2818–2822, May 2014. ISSN 0018-926X.

[113] Y. Zhang, Z. Lin, X. Zhao et T.K. Sarkar : Performance of a massively parallel higher-order method of moments code using thousands of CPUs and its applications. *IEEE Transactions on Antennas and Propagation*, 62(12):6317–6324, Dec 2014. ISSN 0018-926X.

[114] Y. Chen, S. Zuo, Y. Zhang, X. Zhao et H. Zhang : Large-scale parallel method of moments on CPU/MIC heterogeneous clusters. *IEEE Transactions on Antennas and Propagation*, 65(7):3782–3787, July 2017. ISSN 0018-926X.

[115] A.B. Manić, A.P. Smull, F.H. Rouet, X.S. Li et B.M. Notaroš : Efficient scalable parallel higher order direct MoM-SIE method with hierarchically semisepa-

rable structures for 3-D scattering. *IEEE Transactions on Antennas and Propagation*, 65(5):2467–2478, May 2017. ISSN 0018-926X.

[116] Y. Chen, Z. Lin, D. Garcia-Donoro, X. Zhao et Y. Zhang : Performance of a massively parallel method of moment solver and its application. *Applied Computational Electromagnetics Society Journal*, 32(10):872–881, Oct 2017. ISSN 1054-4887.

[117] J. Mackerle : Implementing finite element methods on supercomputers, workstations and pcs : a bibliography (1985-1995). *Engineering Computations*, 13(1):33–85, 1996.

[118] C.T. Wolfe, U. Navsariwala et S.D. Gedney : A parallel finite-element tearing and interconnecting algorithm for solution of the vector wave equation with PML absorbing medium. *IEEE Transactions on Antennas and Propagation*, 48(2):278–284, Feb 2000. ISSN 0018-926X.

[119] S. McFee, Q. Wu, M. Dorica et D. Giannacopoulos : Parallel and distributed processing for h-p adaptive finite-element analysis : a comparison of simulated and empirical studies. *IEEE Transactions on Magnetics*, 40(2):928–933, March 2004. ISSN 0018-9464.

[120] M. L. Yang, H. W. Gao et X. Q. Sheng : Parallel domain-decomposition-based algorithm of hybrid FE-BI-MLFMA method for 3-D scattering by large inhomogeneous objects. *IEEE Transactions on Antennas and Propagation*, 61(9):4675–4684, Sept 2013. ISSN 0018-926X.

[121] H.T. Meng et J.M. Jin : Acceleration of the dual-field domain decomposition algorithm using MPI-CUDA on large-scale computing systems. *IEEE Transactions on Antennas and Propagation*, 62(9):4706–4715, Sept 2014. ISSN 0018-926X.

[122] H.T. Meng, B.L. Nie, S. Wong, C. Macon et J.M. Jin : GPU accelerated finite-element computation for electromagnetic analysis. *IEEE Antennas and Propagation Magazine*, 56(2):39–62, April 2014. ISSN 1045-9243.

[123] H.T. Meng, B.L. Nie, S. Wong, C. Macon et J.M. Jin : GPU accelerated finite-element computation for electromagnetic analysis. *IEEE Antennas and Propagation Magazine*, 56(2):39–62, April 2014. ISSN 1045-9243.

[124] J. Guan, S. Yan et J.M. Jin : An accurate and efficient finite element-boundary integral method with GPU acceleration for 3-D electromagnetic analysis. *IEEE Transactions on Antennas and Propagation*, 62(12):6325–6336, Dec 2014. ISSN 0018-926X.

[125] V. Dolean, M.J. Gander et L. Gerardo-Giorda : Optimized schwarz methods for maxwell's equations. *SIAM Journal on Scientific Computing*, 31(3):2193–2213, 2009.

[126] B. Stupfel et M. Mognot : A domain decomposition method for the vector wave equation. *IEEE Transactions on Antennas and Propagation*, 48(5):653–660, May 2000. ISSN 0018-926X.

[127] Seung-Cheol Lee, Marinos N. Vouvakis et Jin-Fa Lee : A non-overlapping domain decomposition method with non-matching grids for modeling large finite antenna arrays. *Journal of Computational Physics*, 203(1):1–21, 2005. ISSN 0021-9991.

[128] Z. Peng et J.F. Lee : Non-conformal domain decomposition method with second-order transmission conditions for time-harmonic electromagnetics. *Journal of Computational Physics*, 229(16):5615 – 5629, 2010. ISSN 0021-9991.

[129] V. Dolean, M.J. Gander, S. Lanteri, J.F. Lee et Z. Peng : Effective transmission conditions for domain decomposition methods applied to the time-harmonic curl-curl maxwell's equations. *Journal of Computational Physics*, 280:232–247, 2015. ISSN 0021-9991.

[130] P. Aristidou, D. Fabozzi et T. Van Cutsem : Dynamic simulation of large-scale power systems using a parallel schur-complement-based decomposition method. *IEEE Transactions on Parallel and Distributed Systems*, 25(10):2561–2570, Oct 2014. ISSN 1045-9219.

[131] Y.J. Li et J.M. Jin : Parallel implementation of the FETI-DPEM algorithm for general 3D EM simulations. *Journal of Computational Physics*, 228(9):3255–3267, 2009. ISSN 0021-9991.

[132] Stylianos Dosopoulos, Bo Zhao et Jin-Fa Lee : Non-conformal and parallel discontinuous galerkin time domain method for maxwell's equations : EM analysis of ic packages. *Journal of Computational Physics*, 238:48–70, 2013. ISSN 0021-9991.

[133] W.J. Wang, R. Xu, H.Y. Li, Y. Liu, X.Y. Guo, Y. Xu, H.L. Li, H.J. Zhou et W.Y. Yin : Massively parallel simulation of large-scale electromagnetic problems using one high-performance computing scheme and domain decomposition method. *IEEE*

Transactions on Electromagnetic Compatibility, 59(5):1523–1531, Oct 2017. ISSN 0018-9375.

[134] Q. Liu, W. Zhao, J. Cheng, Z. Mo, A. Zhang et J. Liu : A programming framework for large scale numerical simulations on unstructured mesh. *In IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 310–315, April 2016.

[135] W.P. Pala, A. Taflove, M.J. Piket et R.M. Joseph : Parallel finite difference-time domain calculations. *In International Conference on Computation in Electromagnetics*, pages 83–85, Nov 1991.

[136] V. Varadarajan et R. Mittra : Finite-difference time-domain (FDTD) analysis using distributed computing. *IEEE Microwave and Guided Wave Letters*, 4(5):144–145, May 1994. ISSN 1051-8207.

[137] Z.M. Liu, A.S. Mohan, T.A. Aubrey et W.R. Belcher : Techniques for implementation of the FDTD method on a CM-5 parallel computer. *IEEE Antennas and Propagation Magazine*, 37(5):64–71, Oct 1995. ISSN 1045-9243.

[138] A.D. Tinniswood, P.S. Excell, M. Hargreaves, S. Whittle et D. Spicer : Parallel computation of large-scale FDTD problems. *In Third International Conference on Computation in Electromagnetics (Conf. Publ. No. 420)*, pages 7–12, April 1996.

[139] C. Guiffaut et K. Mahdjoubi : A parallel FDTD algorithm using the MPI library. *IEEE Antennas and Propagation Magazine*, 43(2):94–103, Apr 2001. ISSN 1045-9243.

[140] C. Guiffaut et K. Mahdjoubi : Generalized unsplit E-H PML with JEC technique for dispersive media, Jan 2000.

[141] U. Andersson : *Time-Domain Methods for the Maxwell Equations*. Doctoral dissertation, Royal Institute of Technology, Stockholm, Suède, February, 2001.

[142] Y. Zhang, J. Song et C.H. Liang : MPI-based parallelized locally conformal FDTD for modeling slot antennas and new periodic structures in microstrip. *Journal of Electromagnetic Waves and Applications*, 18(10):1321–1335, 2004.

[143] Y. Zhang, W. Ding et C.H. Liang : Study on the optimum virtual topology for MPI based parallel conformal FDTD algorithm on pc clusters. *Journal of Electromagnetic Waves and Applications*, 19(13):1817–1831, 2005.

[144] T.P. Stefanski : Fast implementation of FDTD-compatible green's function on multicore processor. *IEEE Antennas and Wireless Propagation Letters*, 11:81–84, 2012. ISSN 1536-1225.

[145] S. Jiang, Y. Zhang, Z. Lin et X. Zhao : An optimized parallel FDTD topology for challenging electromagnetic simulations on supercomputers. *International Journal of Antennas and Propagation*, 2015:1–10, Dec 2015.

[146] K. Zheng, H. Luo, Z. Mu, J. Li et G. Wei : Parallel TSS-FDTD method for analyzing underwater low-frequency electromagnetic propagation. *IEEE Antennas and Wireless Propagation Letters*, 15:1217–1220, 2016. ISSN 1536-1225.

[147] M Zhang, C. Liao, X. Xiong, Z. Ye et Y. Li : Solution and design technique for beam waveguide antenna system by using a parallel hybrid-dimensional FDTD method. *IEEE Antennas and Wireless Propagation Letters*, 16:364–368, June 2017. ISSN 1536-1225.

[148] F. Bouttout : *Analyse rigoureuse de l'antenne microbande circulaire multicouche. Application a la structure annulaire*. Thèse de doctorat, Université Ferhat ABBAS, Sétif, Oct. 2001.

[149] U.K. Revankar et A. Kumar : Experimental investigation of three-layer electromagnetically coupled circular microstrip antennas. *Electronics Letters*, 27(13):1187–1189, June 1991. ISSN 0013-5194.

[150] W. Chew et T.M. Habashy : The use of vector transforms in solving some electromagnetic scattering problems. *Antennas and Propagation, IEEE Transactions on*, 34 (7):871–879, 1986. ISSN 0018-926X.

[151] S.V. Adve et K. Gharachorloo : Shared memory consistency models : A tutorial. *IEEE Computer*, 29:66–76, 1995.

[152] F. Yu, S.C. Yang, F. Wang, G.C. Chen et C.C. Chan : Symbolic consistency checking of OpenMP parallel programs. *SIGPLAN Not.*, 47(5):139–148, juin 2012. ISSN 0362-1340.

[153] G. Pelosi, S. Selleri et R. Coccioli : *Quick Finite Element Methods for Electromagnetic Waves with Cdrom*. Artech House, Inc., Norwood, MA, USA, 1998. ISBN 0890068488.

[154] O.C. Zienkiewicz, L.R. Taylor et J.Z. Zhu : *The Finite Element Method : Its Basis and Fundamentals*, volume I. Butterworth-Heinemann, 01 2005.

[155] ANSYS® academic research, release 15.0, 2015. URL http://www.ansys.com.

[156] S. Makarov : *Antenna and EM Modeling with Matlab*. Wiley, 2002. ISBN 0-471-46740-5.

[157] W. Gropp, E. Lusk et T.L. Sterling : *Beowulf Cluster Computing with Linux*. Scientific and engineering computation. MIT Press, 2003. ISBN 9780262692922.

ملخص

في هذا العمل ، نقدم تقنيات لحسابات عالية الأداء لمشاكل في ميدان الحقل الكهرومغناطيسي باستخدام مجموعة عنقودية Beowulf و تطبيق MATLAB. أولاً قمنا بإنشاء مجموعة تحتوي على عقدة رئيسية واحدة ،وثمانية عقد حوسبة متماثلة ، ومحول جيجابت إيثرنت (GbE) وتم إستخدام Rocks وهي توزيعة لينكس (Distribution) مفتوحة المصدر لتبسيط استخدام المجموعة وعمليات الصيانة. ثانيا، قمنا بتطوير وتنفيذ ثلاثة تطبيقات تخص حسابات كهرومغنطيسية على المجموعة باستخدام تقنيات البرمجة المتوازية في تطبيق MATLAB. في التطبيق الأول ، يتم حساب مقاومة الدخل لهوائي شريطي دائري متعدد الطبقات ذو مدخل نوع مجس محوري ويعمل في نطاق تردد واسع حيث يتم الوصول الى تسريع قيمته 23. في التطبيق الثاني، يتم تقديم طريقة منهجية لتوليد وحساب تعبيرات رمزية للدوال الأساسية للرتب العليا 2D/3D لطريقة FEM حتى القيمة 10. أخيرا، قمنا بإنجاز تطبيق لحساب معاملات لمجموعة 2D من هوائيات نوع بووتي(Bowtie). أظهرت نتائج المحاكاة أن حجم المصفوفة يمكن زيادته بشكل كبير مما يؤدي إلى نظام خطي بحجم أكبر 100 مرة من النظام قيد الدراسة.

كلمات مفتاح: حسابات كهرومغنطيسية، حسابات عالية الأداء، هوائي شريطي، مجموعة هوائيات، الحساب المتوازي.

Résumé

Dans cette thèse, nous nous sommes intéressés à l'application des techniques de calcul parallèles aux problèmes de calculs en électromagnétiques. Dans une première étape, nous avons construit un cluster de calcul Beowulf composé d'un nœud principal et de huit nœuds de calcul en utilisant des ordinateurs standards interconnectés par un réseau Gigabit. Pour l'exploitation du cluster, nous avons utilisé la solution open source ROCKS basée sur un système de type Linux.
Ensuite, Trois applications de calcul électromagnétiques ont été développées et implémentées sur le cluster en utilisant parallèle MATLAB. Dans la première application, les caractéristiques d'une antenne micro-bande circulaire multicouches opérant en large bande sont calculées. La parallélisation est effectuées en utilisant le modèle de programmation SPMD. Nous avons ensuite développé et implémenter une méthode systématique pour générer et calculer des expressions sous format symbolique des fonctions de base d'ordre supérieur de la FEM. La dernière application concerne la parallélisation de la méthode des moments spatiale pour le calcul d'un réseau d'antennes 2D de type bowtie.

Mots clés : Calcul électromagnétique, calcul parallèle, calcul haute performance, calcul symbolique, SPMD, FEM ordre supérieur, réseaux antennes, antenne microstrip.

Abstract

In this work, efficient high performance computation of electromagnetic field problems using a Beowulf cluster and MATLAB® is presented. A cluster with one master node, eight identical computing nodes, and a Gigabit Ethernet (GbE) switch is firstly built. An open source Rocks toolkit solution is used to simplify the process of deploying high-performance parallel computing clusters. Then, three computational electromagnetic applications are developed and implemented on the cluster using parallel MatLab. In the first application, is calculated the input impedance of a multilayer multiconductor circular microstrip antenna that is excited with coaxial probe and operating in a wide frequency band. Parallelization is then performed over the frequency vector permitting a speedup ratio of 23. In the second application, is presented a systematic method for generating and computing symbolic expressions of 2D/3D FEM basis functions of higher orders up to 10. The final application is dedicated to the parallelization of the spatial moment method code for 2D array of bowtie antenna. The simulation results have shown that the array size can be increased considerably leading to a linear system with a size up to 100 times greater than the one under study.

Keywords : Computational electromagnetics, parallel computation, high performance computing, symbolique compuation, SPMD, High order FEM, array antenna, microstrip antenna