Modélisation électromagnétique 3D d'inducteurs multibrins – Développement d'une méthode intégrale parallélisée

Raphaël Scapolan

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mécanique des fluides, procédés, énergétique**

Arrêté ministériel : 7 août 2006

Présentée par

Raphaël SCAPOLAN

Thèse dirigée par **Annie GAGNOUD**
et codirigée par **Yves DU TERRAIL**

préparée au sein **Laboratoire de Science et Ingénierie des Matériaux et Procédés (SIMaP)**
et de **l'École Doctorale Ingénierie - Matériaux Mécanique Energétique Environnement Procédés Production (I-MEP2)**

Modélisation électromagnétique 3D d'inducteurs multibrins – Développement d'une méthode intégrale parallélisée

Thèse soutenue publiquement le **13 novembre 2014**,
devant le jury composé de :

Monsieur, Mouloud FELIACHI
Professeur, Institut Universitaire de Technologie de Saint-Nazaire, Rapporteur
Monsieur, André NICOLET
Professeur, Université Aix-Marseille, Rapporteur
Monsieur, François BAY
Professeur, Mines Paris Tech, Examinateur
Monsieur, Bernard PAYA
Ingénieur chercheur, EDF, Examinateur
Madame, Annie GAGNOUD
Directeur de recherche, laboratoire SIMaP, Directeur de thèse
Monsieur, Yves DU TERRAIL COUVAT
Ingénieur de recherche, Grenoble INP, Co-Directeur de thèse

Université Joseph Fourier / Université Pierre Mendès France /
Université Stendhal / Université de Savoie / Grenoble INP

Remerciements

Je souhaite commencer par remercier mes deux encadrants, Annie Gagnoud et Yves Du Terrail, pour m'avoir donné l'occasion de travailler sur une thématique originale et pour leur excellente collaboration tout au long de ces années de thèse. Leur expérience et leurs conseils ont largement contribué à faire de cette entreprise une réussite, malgré certains revers et contre-temps qui sont le propre, et le charme, de la recherche.

Je remercie également Messieurs François Bay, Mouloud Feliachi, André Nicolet et Bernard Paya pour avoir accepté d'examiner le produit de ce travail.

Je tiens à saluer l'ensemble des membres de l'équipe du laboratoire EPM pour leur bon accueil et la bonne ambiance générale qu'ils savent conserver.

J'adresse mes remerciements et mes encouragements à tous les modélisateurs, occasionnels ou réguliers, avec qui j'ai eu le plaisir de partager mon deuxième bureau,... pardon, la salle informatique : Sébastien, Marcio, Alimata, Ahmed, Christian, et milles excuses à ceux que j'oublie. Merci pour les bons échanges que nous avons eu ainsi que pour avoir pu disposer ensemble du réseau de stations de travail dans la joie et la bonne humeur ! Bon courage à vous tous pour vos thèses, post-docs, et autres aventures.

Je tiens à dédier ce travail à mes chers parents pour leur fidèle soutien tout au long de ces années. Vos conseils, votre expérience et votre affection m'ont été et me seront toujours des plus précieux.

Enfin, je souhaite également témoigner de mon estime la plus sincère à tous mes amis grenoblois cosmopolites : la tribut des 5 K. corso-auvergnat-malgaches, les dynasties italiennes P. et *tutti quanti*, les franco-égyptiens H., et tant d'autres... sans compter quelques savoyards exilés. Je tiens à vous remercier pour la profondeur et la chaleur de l'amitié que vous m'avez donnée au cours de ces quelques années passées en votre compagnie, amitié qui fait de vous ma famille étendue. Pour tout ce que vous m'avez apporté et appris et en raison de la force de ce qui nous lie, je ne saurai vous écarter de la dédicace de cette thèse.

A mes parents,
A mes amis les plus chers.

« A mesure que s'étend l'île de la connaissance,
grandissent aussi les rivages de notre ignorance. »

John A. Wheeler, 1911-2008, physicien.

Résumé

Afin d'optimiser la conception des installations industrielles de chauffage par induction et de les rendre plus performantes et économes en énergie, l'emploi d'inducteurs multibrins est envisagé. Ces structures, aussi appelées fil de Litz, sont des câbles constitués de plusieurs centaines, voire plusieurs milliers de brins. Ces câbles ont pour but de supprimer l'effet de peau occasionné par l'emploi de hautes fréquences dans les procédés de chauffage inductif et observé habituellement dans des inducteurs pleins. Cependant, dans ce genre de structure, les courants circulant dans les brins s'influencent mutuellement générant ainsi des effets de proximité. Ceux-ci occasionnent des pertes qui peuvent être importantes et fortement dépendantes de la géométrie interne complexe de ces câbles.

Pour concevoir des inducteurs multibrins à faibles pertes, il est primordial de comprendre leur comportement électromagnétique et de savoir les modéliser. La majorité des modèles existants ont été développés pour la modélisation d'inducteurs de plaques de cuisson domestiques ou de bobinages de transformateurs. Qu'ils soient analytiques ou numériques, ces modèles exploitent les symétries de ces géométries afin de simplifier les calculs. Ces simplifications ne permettent pas de tenir compte de la géométrie interne complexe et tridimensionnelle (3D) des inducteurs. De plus, beaucoup de ces modèles utilisent une hypothèse particulière sur la distribution du courant entre les brins qui n'est pas adaptée à la modélisation d'inducteurs industriels.

En raison de la complexité de la géométrie et des importants rapports de tailles entre le diamètre des brins (dizaines de microns) et les dimensions de l'inducteur (dizaines de centimètres ou mètre), les méthodes numériques classiques telles que la Méthode des Éléments Finis ne sont pas adaptées. En effet, les tailles du maillage et du système linéaire peuvent rapidement être trop importantes pour les capacités en mémoire des calculateurs. De surcroît, le contrôle de la qualité du maillage entre les brins n'est pas assuré.

Dans cette thèse, nous présentons le développement d'un logiciel parallèle de calcul électromagnétique dévolu à la modélisation 3D d'inducteurs multibrins. Nous avons conçu un modèle numérique innovant adapté aux systèmes inductifs sur une large bande de fréquences. Nous décrivons une méthode innovante de construction de la géométrie de ces câbles. Ce logiciel est basé sur une méthode numérique de type intégrale ayant l'avantage de ne pas nécessiter le maillage des espaces entre les brins. Cette méthode s'appuie sur les lois d'Ohm et de Biot et Savart. Deux modèles sont implémentés : un modèle volumique, permettant la modélisation vo-

lumique locale de câbles de quelques dizaines de brins et un modèle filiforme, qui permet d'effectuer des calculs globaux sur des géométries de câbles plus proches des configurations industrielles (plusieurs milliers de brins). Les études réalisées montrent l'impact de la géométrie sur le comportement électromagnétique de ce type d'inducteur.

L'emploi du calcul parallèle est une des grandes forces de ce logiciel. Les calculs sont fortement accélérés grâce à la parallélisation de la construction du système linéaire et de sa résolution. Les performances de cet outil ont été éprouvées avec succès sur divers systèmes de calcul intensif.

Ce travail de recherche a été réalisé dans le cadre du projet Innovative Solutions for Induction Systems (ISIS), financé par l'Agence Nationale de la Recherche dans le but de répondre à la problématique du programme Efficacité Énergétique et réduction des émissions de CO2 dans les Systèmes Industriels (EESI). Ce projet a impliqué : EDF R&D, Fives-Celes, Atys Consulting Group, ARMINES-CEP, CNRS-CRISMAT, CNRS-SIMAP et INPT-LAPLACE.

Ces travaux ont bénéficié d'un accès aux moyens de calcul de l'IDRIS au travers de l'allocation de ressources 2014-i2014057126 attribuée par GENCI.

Mots-clé : modélisation numérique, fils de Litz, méthode intégrale, chauffage par induction, calcul parallèle, efficacité énergétique

Abstract

In order to optimize the design of industrial installations of induction heating, to make it more efficient and to reduce their energy consumption, the use of multi-wires inductors is considered. These structures, also called Litz wires, are cable constituted by hundreds, even thousands of wires. The aim of these cables is to remove the skin effect due to the use of high frequencies into inductive processes and usually observed into full inductors. However, in that kind of structure, the currents circulating into the wires affect each other which generate proximity effects. These effects cause losses which can be important and strongly dependent on the complex internal geometry of that cables.

To design lossless multi-wires inductors, it is primordial to be able to model it and to understand their electromagnetic behavior. Most of the existing models had been developed for the modeling of domestic induction heating appliances and transformers windings. Being analytical or numerical, these models exploit the symmetries of these geometries to simplify the calculations. These simplifications cannot take into account the internal geometry of the inductors which is complex and three-dimensional (3D). Moreover, most of these models use a particular hypothesis on the distribution of the current into the wires which is not appropriated for the modeling of industrial inductors.

Considering the complexity of the geometry and the important size ratio between the diameter of the wires (tens of microns) and the dimensions of the inductor (tens of centimeters or meter), the classical methods as the Finite Elements Method are not adapted. Indeed, sizes of the mesh and of the linear system can become too much important for the memory capacities of computers. Furthermore, the control of the mesh quality between the wires is not assured.

In this thesis, we present the development of a parallel software of electromagnetic computation intended to the 3D modeling of multi-wires inductors. We design an innovative numerical method adapted to inductive processes in a large range of frequencies. We describe an original method for building the geometry of these inductors. This software uses an integral method in which the meshing of spaces between the wires is not necessary. This numerical method is based on the Ohm's and Biot and Savart's laws. Two models are implemented : a volume model, enabling the local and volume modeling of cables with some tens of wires, and a filiform model, which permits to perform global computations on geometries close to industrial configurations (many thousands of wires). The studies realized show the impact of the geometry on the electromagnetic behavior of that type of inductor.

Abstract

The utilization of parallel computing is one of the great forces of this software. Computations are accelerated thanks to the parallelization of the building of the linear system and its solving. Performances of this tool had been successfully tested on various systems of high performance computing.

This work has been realized as a part of the project Innovative Solutions for Induction System (ISIS), funded by the French National Research Agency to bring solutions to the program Energy Efficiency and reduction of CO2 Emissions for Industrial systems (EESI). This project involved : EDF R&D, Fives-Celes, Atys Consulting Group, ARMINES-CEP, CNRS-CRISMAT, CNRS-SIMAP and INPT-LAPLACE.

This work was granted access to the HPC resources of IDRIS under the allocation 2014-i2014057126 made by GENCI.

Keywords : numeric modeling, Litz wires, integral method, heating induction, parallel computing, energy efficiency

Sommaire

Introduction

De nos jours, la réponse aux besoins énergétiques de l'humanité constitue un véritable défi. En effet, face à la hausse continuelle de la demande en énergie, la production se trouve de plus en plus contrainte.

Tout d'abord, la majorité des ressources énergétiques actuelles (environ 80 % [1]) est constituée de ressources fossiles (pétrole, gaz, charbon et hydrocarbures diverses) dont la quantité est limitée et dont l'utilisation a un impact écologique important. De plus, ces matières premières sont l'objet d'enjeux géopolitiques importants, ce qui renforce l'instabilité de leur prix et tend à l'accroître. Quant aux centrales nucléaires, elles ont un impact écologique bien moindre, en fonctionnement normal, mais les risques en cas de défaillance et la question du retraitement et du stockage des déchets à longue durée de vie constituent les principaux inconvénients de l'énergie nucléaire. Ceci est vrai d'un point de vue technique. C'est également le cas d'un point de vue politique car, de manière générale, l'acceptabilité de cette source d'énergie dans l'opinion publique est en baisse [2]. Enfin, à l'instar des ressources fossiles, les ressources en combustible sont limitées et leur caractère local peut être l'objet de tensions entre pays. Ajoutons que, d'après le rapport [2], les précédentes sources d'énergies sont exploitées avec un rendement faible, de l'ordre de 30 %. Enfin, les sources d'énergies renouvelables sont encore très peu exploitées et, en l'état actuel des techniques, ne permettent pas de combler totalement les besoins énergétiques mondiaux.

Si des efforts sont entrepris dans le but d'accroître la production d'énergie, de nombreuses recherches sont également menées afin d'en réduire la consommation dans divers domaines, notamment dans les secteurs de l'industrie et des transports qui absorbent chacun environ un tiers de l'offre énergétique mondiale [1]. En France, en 2009, les secteurs liés à la sidérurgie, la fonderie et la métallurgie ont constitué environ 23 % de l'énergie totale consommée par l'industrie [3].

Dans ce contexte, l'induction électromagnétique présente bien des avantages notamment en ce qui concerne le chauffage industriel [4]. Parmi les points forts que possède la technique du chauffage par induction, citons :

— une installation compacte et plus simple que les installations utilisant d'autres sources (gaz, vapeur, flamme) ;
— une exploitation facilitée par une faible latence au démarrage et à l'arrêt du procédé ;

— un contrôle précis de la qualité du chauffage en termes de temps et de localisation de la zone à traiter ;
— un chauffage sans contact et sans combustion, ce qui permet d'éviter la contamination ou la dégradation des objets chauffés.

Ajoutons que, suivant les installations, le rendement peut atteindre 90 %. Ces nombreux atouts font de l'induction électromagnétique un procédé de choix en matière de chauffage industriel. Dans l'article [4], les auteurs présentent une revue détaillée des domaines d'application du chauffage par induction.

Nous avons pris soin de préciser que l'excellent rendement de l'induction électromagnétique est dépendant de l'installation dans laquelle elle est utilisée. En effet, le rendement dépend principalement de la nature du matériau de l'objet à chauffer (couramment appelée *charge*), de sa géométrie et de la géométrie de l'installation. Certaines configurations sont favorables à un bon rendement. Dans d'autres cas, l'essentiel de la puissance consommée est perdue dans l'installation, notamment dans l'inducteur.

Ces pertes peuvent être causées par *l'effet de peau* électromagnétique. L'effet de peau apparaît dans un conducteur soumis à un champ électromagnétique alternatif. En fonction de la fréquence f du champ, la densité de courant électrique circulant dans le conducteur tend à se répartir majoritairement dans une couche en surface du conducteur. Cette couche a une épaisseur δ exprimée selon la relation 1.

$$\delta = \sqrt{\frac{2}{\sigma\mu\omega}} \tag{1}$$

avec :
— σ, la conductivité électrique du matériau ;
— μ, sa perméabilité électrique ;
— $\omega = 2\pi f$, la pulsation associée à la fréquence f.

L'épaisseur de peau diminue donc si la fréquence augmente. Il en résulte que la section de passage du courant est d'autant réduite. Par conséquent, la résistance du conducteur et les pertes Joule en son sein augmentent. Les pertes varient suivant la forme de la section du conducteur [5]. C'est pourquoi, l'emploi d'inducteurs pleins pour des applications à hautes fréquences présente deux inconvénients. D'une part, la proportion d'énergie transmise à la charge peut être faible ($\approx$ 60 %). D'autre part, il faut évacuer l'énergie perdue dans les inducteurs pour ne pas risquer leur dégradation. Ceux-ci sont généralement refroidis par une circulation d'eau interne.

Face à ces difficultés, une idée originale consiste à employer des inducteurs divisés, formés de brins cylindriques individuels et recouverts d'isolant. Plusieurs brins sont torsadés et forment un paquet. Plusieurs paquets sont torsadés ensemble, formant un paquet plus gros. Le processus est répété jusqu'à obtenir un câble multibrins du diamètre désiré. Un tel agencement porte le nom de *fil de Litz*. L'idée de base de cette conception consiste à emmêler des brins de section inférieure à l'épaisseur de peau dans le but de contrer l'effet de peau en forçant le courant à se répartir uniformément dans la section du câble.

Malheureusement, un nouveau phénomène apparaît alors : *l'effet de proximité*. Celui-ci se manifeste lorsqu'un minimum de deux fils voisins sont parcourus chacun par un courant alternatif. S'ils circulent dans le même sens, alors les deux courants ont tendance à se repousser mutuellement. Ainsi, la densité de courant n'est pas uniforme dans la section des brins : elle est faible dans la zone où les brins sont voisins et maximale à l'opposé. L'inverse se produit si les courants circulent en sens opposés : les courant s'attirent mutuellement [5]. De nouvelles pertes apparaissent donc au sein des inducteurs multibrins, les pertes par effet de proximité.

La question de l'influence de la géométrie interne d'un inducteur multibrins est donc essentielle. L'objectif du présent travail de thèse consiste à fournir des outils permettant de modéliser des inducteurs soumis à une tension sinusoïdale et d'explorer l'impact de certains paramètres géométriques sur leur comportement électromagnétique. A cette fin, le développement d'un logiciel parallèle a été entrepris.

Dans le premier chapitre, nous ferons un état des lieux sur les techniques de modélisation des inducteurs multibrins. Nous présenterons également les principes essentiels du calcul parallèle. Dans le deuxième chapitre, nous décrirons le logiciel de calcul que nous avons implémenté. Cette description comprendra une présentation du modèle de construction de la géométrie des inducteurs, une présentation de la méthode numérique employée pour la modélisation électromagnétique ainsi que son implémentation dans le code parallèle et une validation du logiciel. Dans le chapitre suivant, nous exposerons les résultats de simulations menées dans le but d'étudier l'influence de la géométrie de l'inducteur sur ses pertes. Nous mettrons également en évidence certaines limites du logiciel. C'est pourquoi, dans le dernier chapitre, nous présenterons une deuxième méthode, également implémentée dans le logiciel et permettant de simuler des inducteurs proches des installations industrielles.

Chapitre 1

Etat de l'art

Ce chapitre présente l'état de l'art dans deux domaines. Tout d'abord, nous présenterons une revue bibliographique des modèles utilisés pour la modélisation de câbles en fil de Litz. Enfin, nous décrirons de manière succincte l'évolution des capacités de calcul des ordinateurs ayant permis le développement du calcul parallèle.

1.1 Exposé de la problématique

L'utilisation du fil de Litz semble constituer une solution prometteuse pour l'élaboration d'inducteurs plus économes en énergie. Afin d'optimiser leur conception, il est nécessaire de disposer de modèles permettant de calculer le comportement électromagnétique d'inducteurs multibrins. Ces modèles doivent rendre compte de l'influence de la géométrie interne sur les pertes par effet de proximité entre les brins. Divers paramètres entrent en jeu parmi lesquels les pas et les sens de torsadage des brins et des paquets de brins ainsi que l'agencement de ces derniers dans l'inducteur.

La modélisation de ce type d'inducteur se heurte à deux problèmes. Le premier réside dans la difficulté à représenter géométriquement les formes tridimensionnelles complexes que prennent les brins dans ces structures. Ces formes ne sont pas descriptibles à partir de formes géométriques simples ni grâce à des lois analytiques. Associé à cela, le grand nombre de brins formant les inducteurs, de quelques dizaines à plusieurs milliers, constitue un second problème. Tous les brins s'influençant mutuellement, le calcul du comportement global de l'inducteur n'est pas aisé.

Nous présentons ci-après les principaux modèles existants pour la modélisation d'inducteurs en fil de Litz. Nous évoquerons également quelques publications remarquables parues très récemment.

1.2 Etat de la modélisation électromagnétique du fil de Litz

La grande majorité de la littérature traitant de la modélisation du fil de Litz dans le domaine de l'induction électromagnétique est répartie entre deux domaines d'applications : les transformateurs et, plus récemment, les plaques de cuisson domestiques. Deux types de modélisation apparaissent : l'approche analytique et l'approche numérique.

1.2.1 Modèles analytiques

Afin de contourner les difficultés évoquées en partie 1.1, des modèles analytiques ont été développés. Ceux-ci sont d'abord apparus dans le contexte de la modélisation des transformateurs.

Modélisation du bobinage des transformateurs

Dans les grandes lignes, les transformateurs sont constitués de plusieurs bobinages entourant un noyau magnétique. Ces bobinages peuvent être de diverses formes et sont souvent constitués à partir d'un fil unique ou d'un fil de Litz. Compte tenu de leur géométrie, des simplifications peuvent être faites et, la plupart du temps, la symétrie cylindrique est utilisée. C'est sur cette dernière que s'appuient les modèles analytiques employés pour modéliser le bobinage des transformateurs.

Dans l'article [6], les auteurs font une revue des méthodes utilisées pour le calcul de la résistance en régime alternatif sinusoïdal des bobinages de transformateurs. Toutes s'inspirent de la méthode des plaques de Dowell [7] qui, par approximations successives, remplace un bobinage formé d'un fil unique par un feuilletage – ensemble de plaques – ayant une résistance équivalente en régime continu. L'espace isolant entre les éléments du bobinage réel est pris en compte *via* un facteur de porosité. La résistance en régime alternatif est calculée à partir de la résistance en régime continu multipliée par un facteur tenant compte de la symétrie cylindrique de la bobine, du facteur de porosité et de l'épaisseur de peau correspondante à la fréquence de travail. En l'état, cette méthode ainsi que les autres méthodes présentées dans l'article [6] ne sont pas adaptées au calcul de la résistance en régime alternatif d'une bobine de transformateur fabriquée avec du fil de Litz.

Dans l'article [8], Schutz utilise un modèle en circuit électrique équivalent basé sur la méthode de Dowell pour réaliser le calcul de la résistance d'une bobine de transformateur en fil de Litz. Une des hypothèses de ce modèle est que le fil de Litz permet une répartition homogène du courant dans le câble. Dans l'ensemble, les résultats donnés par le modèle sont cohérents avec les mesures effectuées pour des fréquences inférieures à 3 MHz.

Dans les publications [9] et [10], Ferreira présente une adaptation de la méthode de Dowell pour le calcul de la résistance en régime alternatif de bobinages en fil de Litz. Pour ce faire, l'auteur suppose que le courant traversant la bobine est réparti équitablement entre tous les brins. Le calcul de la résistance s'appuie sur la solution analytique des courants créés dans un conducteur rond plein soumis à un courant alternatif et exprimé grâce aux fonctions de Bessel. La comparaison des résultats prédis avec des mesures confirme la validité de la méthode. Une approche similaire est présentée dans l'article [11].

Dans l'article [12], Sullivan propose une méthode de calcul du nombre optimal de brins à choisir pour la conception d'un bobinage en fil de Litz. Comme pour la méthode Ferreira, une des hypothèses est que le courant est identique dans tous les brins. La puissance dissipée dans le bobinage est fonction de la valeur efficace du courant total en régime alternatif, de la résistance en régime continu et d'un facteur liant cette dernière à la résistance en régime alternatif. Ce facteur tient compte de la fréquence et de certains paramètres géométriques de la bobine. Dans l'article [13], Sullivan précise que les méthodes de Dowell et Ferreira divergent à hautes fréquences et propose une approche basée sur la Méthode des Éléments Finis sur laquelle nous reviendrons.

Dans l'article [14], Tourkani décrit un modèle de calcul des pertes dans le bobinage en utilisant la solution analytique des courants créés dans un conducteur rond et en tenant compte d'un champ de fuite variant linéairement à travers le bobinage. La résistance en régime alternatif est calculée en supposant que tous les brins du fil de Litz sont traversés par le même courant.

Nous venons de survoler les principaux modèles analytiques employés pour la modélisation des bobinages multibrins des transformateurs. Une analyse détaillée de chacun de ces modèles nous apprend qu'aucun ne permet de tenir compte finement de la structure interne 3D des fils de Litz. Au passage, nous nous apercevons que certains auteurs appellent fil de Litz des câbles formés à partir d'un seul paquet de brins tandis que d'autres utilisent cette appellation pour des câbles formés de plusieurs paquets de brins, et éventuellement de plusieurs niveaux de paquets. Enfin, nous constatons que tous les modèles sont basés sur l'hypothèse selon laquelle le courant est le même dans tous les brins.

Modélisation des plaques de cuisson

Le développement des plaques de cuisson à induction électromagnétique, qui a vu le jour il y a environ 40 ans [15], est une technologie bien plus récente que celle des transformateurs. La littérature concernant la modélisation des inducteurs utilisés dans ces systèmes est plus restreinte et est dominée par la production des chercheurs espagnols de l'université de Zaragoza.

Dans les publications [16] et [17], Acero présente un modèle analytique de calcul de la résistance de l'inducteur en fil de Litz d'une plaque de cuisson. Nous

retrouvons à nouveau l'hypothèse du courant égal dans tous les brins. Ce modèle ne tient pas compte de la structure interne 3D de l'inducteur.

Réflexion sur l'hypothèse du fil de Litz idéal

Il est intéressant de noter que tous les modèles analytiques cités précédemment sont basés sur une hypothèse récurrente : le courant est le même dans tous les brins. C'est l'hypothèse du fil de Litz idéal qui considère que les brins sont entrelacés de manière à ce que, sur une certaine longueur de câble, chaque brin passe par toutes les positions possibles dans la section du câble [17]. Les auteurs s'appuient généralement sur cette hypothèse pour considérer que, globalement, tous les brins sont tous soumis à un environnement électromagnétique similaire. Par conséquent, ils sont équivalents et donc traversés par le même courant. Ainsi, la modélisation des fils de Litz peut être simplifiée.

Compte tenu de la complexité de la géométrie des inducteurs en fil de Litz et de la difficulté à les modéliser, nous avons cherché à comprendre comment cette hypothèse a été validée. Murgatroyd formule explicitement cette hypothèse pour les fils de Litz en 1989 [18]. L'auteur précise que cette approximation n'est pas rigoureusement vraie au niveau local puisque « l'effet de proximité produit une densité de courant non uniforme dans chaque brin ». Mais, selon lui, cette approximation doit être bonne à l'échelle du brin et aux échelles supérieures. Néanmoins, il n'apporte pas de confirmation de cette affirmation.

En remontant la bibliographie, nous avons constaté que la plupart des auteurs s'appuient sur l'article de Lofti, en 1993, *A High Frequency Model for Litz Wire for Switch-Mode Magnetics* [19]. L'auteur rappelle succinctement la structure et l'intérêt du fil de Litz dans les applications fonctionnant à hautes fréquences.

> « *A more sophisticated type of wire, known as Litzendraht or Litz wire is constructed such that the individual strands possess both azimuthal and radial transposition. This construction offers more benefits, by reducing both external proximity effects and internal skin effects. The transposition is either continuous, discrete or random. The purpose of this type of transposition is to force each strand to occupy all positions of every other strand comprising the wire. This will, at least theoretically, force the impedance of every strand to be equal, resulting in an equal current distribution [1,2,3,4,5]* ».

Parmi les références citées ici dans le texte ci-dessus, l'article de Howe, *The high-frequency resistance of multiply-stranded insulated wire* [20], propose une méthode permettant de calculer la résistance de câbles multibrins en régime harmonique et de déterminer la taille et le nombre de brins appropriés pour la réduire. Il nous semble édifiant de rapporter le propos de cet auteur et de le comparer à l'argumentaire soutenu par Lofti :

> « *Conductors which have to carry high-frequency currents are often made up of a large number of separately insulated fine wires stranded*

> *together, with the object of compelling the current to distribute itself over the whole cross-section of the conductor. This may be done for two different reasons : firstly, to decrease the variation of inductance and resistance with change of frequency ; and, secondly, to decrease the effective resistance of the conductor at high frequencies.*
>
> *To be effective, it is necessary that every wires should occupy in turn the same relative position in the conductor, so that the electromotive force induced in each wire by the magnetic flux should have the same average value over the whole length of the wire. If every separate wire has the same resistance, the same applied [potential difference] and the same induced [electromagnetic field], both in amplitude and phase, they will all necessarily carry the same current, and the total current will therefore be uniformly distributed between all the wires. The usual method to obtaining this similarity in the path of every strand is to make a conductor of three, four or five wires twisted together, and then to twist three such conductors together and so on until the resulting conductor contains the required number of wires* ».

Nous retrouvons exactement les mêmes idées dans les deux articles : l'entrelacement des brins est tel que tous ont le même comportement électromagnétique et tous sont traversés par le même courant. Dans l'article [20], les calculs sont basés sur ce postulat ainsi que sur l'hypothèse tacite que tous les brins sont placés dans un champ magnétique externe uniforme. La méthode est appliquée pour prédire la résistance d'un câble formant un solénoïde à spires jointives. Les résultats obtenus sont en bon accord avec les mesures. Cet article date de 1917.

Limites des modèles analytiques

De nombreuses études, appuyées par des mesures, semblent confirmer la validité de l'hypothèse du fil de Litz idéal. Notons toutefois que celle-ci a été employée et validée dans une configuration particulière présentant une symétrie cylindrique (solénoïde). Nous constatons également que la plupart des modèles analytiques développés pour la modélisation des câbles multibrins s'appuient sur cette théorie. De plus, ceux-ci ont toujours été appliqués à des câbles installés dans des systèmes inductifs présentant une certaine symétrie cylindrique (solénoïde, bobines de transformateurs, inducteurs de plaques de cuisson) et dans certaines conditions favorables (grande épaisseur de peau). Nous nous demandons si cette hypothèse est également valable si le câble suit un chemin quelconque. En effet, dans les installations industrielles de chauffage, les inducteurs suivent des chemins variés et font partie de systèmes qui n'ont pas toujours une géométrie présentant de symétrie particulière.

Dans notre travail, nous souhaitons nous intéresser à l'influence de l'arrangement 3D interne des brins dans les inducteurs en fil de Litz. Les modèles analytiques existants n'offrent pas cette possibilité. De plus, l'hypothèse du fil de Litz

idéal supposant un courant identique dans tous les brins nous paraît trop forte et non validée dans un cas général.

Modèle analytique remarquable

A ce stade de notre revue bibliographique, il convient de démarquer un modèle analytique digne d'intérêt. En effet, le modèle développé par Cerri dans les publications [21] et [22] propose de tenir compte du chemin 3D suivi par les brins d'un inducteur d'une plaque de cuisson. Les brins sont connectés en parallèle et soumis à une différence de potentiel sinusoïdale. Le modèle permet de calculer une matrice tenant compte des auto-inductances et inductances mutuelles des brins. Les inconnues du système linéaire obtenu sont les courants dans chaque brin. La pertinence du modèle est confirmée par comparaison à des mesures.

Cette approche est donc remarquable puisqu'elle ne s'appuie pas sur l'hypothèse du fil de Litz idéal mais tient compte du chemin suivi par les brins. D'ailleurs, les auteurs déclarent que le fil de Litz « assure presque [1] une impédance égale des brins » [22]. Dans les études [21] et [22], les chemins suivis par les brins peuvent être décris analytiquement. Les brins sont modélisés de manière filiforme, le maillage de chaque brin est donc unidimensionnel. Les calculs effectués permettent de modéliser un inducteur de plaque de cuisson contenant jusqu'à 30 brins.

1.2.2 Modèles numériques Eléments Finis

Avantages et inconvénients de la méthode des Eléments Finis pour la modélisation électromagnétique des inducteurs multibrins

Face à la complexité de la géométrie des inducteurs multibrins et des calculs à mener pour simuler leur comportement électromagnétique, de nombreux modèles analytiques ont été mis en place. Néanmoins, avec les progrès de l'informatique, la puissance ainsi que les capacités en mémoire des ordinateurs se sont accrues. Des méthodes numériques telles que la Méthode des Eléments Finis ont été développées.

Cette méthode s'appuie sur un maillage de la géométrie et présente généralement deux avantages essentiels face aux méthodes analytiques. Premièrement, elle permet de calculer les phénomènes entrant en jeux localement, en chaque nœud du maillage. Deuxièmement, elle permet de prendre en compte des systèmes aux géométries bien plus complexes. Cependant, dans le cas de la modélisation électromagnétique d'inducteurs multibrins, la force de cette méthode est aussi sa faiblesse.

En effet, le principal inconvénient de cette méthode réside dans le fait qu'elle nécessite de mailler complètement l'espace dans lequel est placé l'inducteur. Pour assurer une bonne précision des calculs, il est indispensable que les brins ainsi que les espaces entre ceux-ci soient suffisamment maillés. Or, un inducteur multibrins

1. C'est nous qui soulignons.

peut compter plusieurs milliers de brins. A cause de sa géométrie 3D ainsi que de l'importante différence d'échelle entre le diamètre de la section d'un brin (de l'ordre de la dizaine à la centaine de microns) et le diamètre de la section de l'inducteur (jusqu'à plusieurs centimètres), la taille du maillage et, par conséquent, celle du système linéaire associé peuvent devenir extrêmement conséquentes. La modélisation peut donc être limitée par la taille de mémoire disponible sur une machine. Ajoutons également qu'en raison de la complexité de la géométrie, le contrôle de la qualité du maillage, notamment entre les brins, est extrêmement difficile.

Enfin, la boite d'air dans laquelle est incluse la géométrie du système simulé doit être suffisamment grande pour que les résultats de calculs au sein du système ne soient pas perturbés par la condition imposée en frontière de la boite. Des outils comme les boites infinies permettent de modéliser plus rigoureusement les conditions à l'infini, néanmoins le problème du maillage de l'air entre les brins perdure.

Certaines études proposent une approche simplifiée en 2D pour la modélisation du fil de Litz [23], [24]. Néanmoins, en raison de l'approximation 2D, les brins sont considérés comme étant droits et parallèles. Compte tenu des symétries de la géométrie, des approximations peuvent être faites et les études peuvent être menées en 2D en utilisant la symétrie axiale (2D-axi). Cette technique est notamment employée pour la modélisation des transformateurs. Cette approche revient à considérer que les brins suivent des chemins parallèles entre eux et forment des cercles concentriques, centrés sur l'axe de symétrie. Dans les deux cas, l'impact de l'arrangement 3D réel n'est donc pas pris en compte. De plus, quand bien même une modélisation 2D ou 2D-axi réduit drastiquement la taille du maillage et le nombre de degrés de liberté du système linéaire en comparaison à une approche 3D, les difficultés évoquées ci-dessus subsistent malgré tout. Le rapport de taille entre les brins et l'installation complète reste important et le nombre de brins est grand. Ces approches ne présentent donc pas d'intérêt pour une étude précise de l'impact de l'arrangement interne des inducteurs multibrins sur leurs performances énergétiques.

Modèles d'homogénéisation par propriétés équivalentes

Pour palier les problèmes décris précédemment, divers modèles dits *d'homogénéisation* ont été proposés. Ces modèles peuvent être employés lorsque une partie de l'installation à modéliser possède une géométrie présentant une certaine périodicité. Le but de ces modèles est de permettre, par changement d'échelle, de modéliser cette zone en décrivant une géométrie plus simple que la géométrie réelle mais affectée de propriétés physiques dites *équivalentes*, sensées être représentatives de la zone en question. Un exemple simple de modèle d'homogénéisation est proposé dans l'article [24] pour la modélisation en 2D d'inducteurs multibrins. Dans ce modèle, la géométrie réelle du câble multibrins en cuivre est remplacée, pour le calcul en Éléments Finis, par un câble plein de même rayon mais ayant une conductivité électrique proportionnelle à la densité de cuivre dans le câble multibrins.

Dans l'article [13], Sullivan utilise le principe de l'homogénéisation pour éva-

luer l'impact de certains paramètres géométriques (écart entres les spires, écarts entre les brins, diamètre des brins) sur les pertes dans un bobinage de transformateur. Il considère un pavage régulier infini dans un repère 2D cartésien formé de disques de diamètre d, égal au diamètre des brins, et espacés, bords à bords, d'une distance h selon l'axe (Ox) et d'une distance v selon l'axe (Oy). Les centres de quatre disques forment un rectangle de dimensions $(h + d) \times (v + d)$. Les brins sont en cuivre, ne sont pas alimentés en courant et sont supposés soumis à un champ magnétique externe uniforme. La géométrie choisie pour les calculs Éléments Finis est basée sur un motif élémentaire du pavage. Ce domaine forme un rectangle de dimensions $(h+d)\times(v+d)$ dont les sommets sont chacun situés au centre d'un disque. Dans chaque coin de la géométrie se trouve donc un quart de disque en cuivre. L'espace les séparant est isolant. Des conditions de symétrie et d'antisymétrie sont définies sur les frontières du domaine afin de prendre en compte la périodicité de la structure. L'auteur montre que cette approche donne de meilleurs résultats à haute fréquence que les méthodes de Dowell [7] et Ferreira [10]. Dans l'article [25], Sullivan améliore cette méthode et calcule une perméabilité magnétique équivalente, toujours selon une approche 2D.

Dans l'article [26], les auteurs proposent un modèle analogue d'homogénéisation par propriétés équivalentes (perméabilité magnétique et conductivité électrique) pour la modélisation de transformateurs planaires. Ils précisent que leur méthode peut être employée pour modéliser des installations dont certaines portions de géométrie présentent une périodicité, et ce, en 2D comme en 3D. A partir de calculs Éléments Finis menés sur le motif élémentaire, des propriétés équivalentes sont déterminées et un nouveau calcul Éléments Finis est lancé. Cette fois, le calcul est effectué sur la géométrie complète de l'installation en remplaçant la zone périodique de la géométrie par une géométrie non périodique affectée de ces propriétés.

Un autre modèle en perméabilité magnétique et conductivité électrique équivalentes est proposé dans l'article [27] pour la modélisation 2D de fil de Litz dans un système de transfert d'énergie sans fil. Dans ce cas, le fil de Litz est supposé idéal.

Nous avons mis en évidence les limites de la Méthode des Éléments Finis pour la modélisation en 3D de câbles en fil de Litz. Nous avons présenté quelques études et modèles d'homogénéisation alternatifs basés sur la détermination de propriétés équivalentes. Ces modèles sont essentiellement employés pour la modélisation de bobines de transformateurs. Néanmoins, ils présentent certaines limites.

La principale limite réside dans le fait qu'ils sont adaptés pour la représentation de géométries présentant une périodicité. Ceci est effectivement le cas dans le cadre de la modélisation de 2D du bobinage d'un transformateur. En revanche, compte tenu de l'arrangement 3D des brins et de leur nombre dans un inducteur en fil de Litz, il n'est pas évident qu'un tel câble présente une géométrie interne périodique. Si périodicité il y a et si le motif élémentaire peut être déterminé, il faut, de surcroît, que les calculs Éléments Finis sur ce dernier puissent être menés. En effet, le motif serait alors toujours une géométrie 3D représentant une portion de câble et les limites en termes de maillage présentées dans la partie 1.2.2 pourraient subsister.

Ceci explique notamment le fait que de nombreuses études ont été effectuées avec l'approximation 2D. Dans ce cas, les modèles peuvent être utilisés, mais l'influence de l'arrangement 3D est complètement masquée puisque cette approche revient à considérer que la géométrie est invariante par translation (dans le cas d'une modélisation 2D) ou par rotation (modélisation 2D-axi).

Modélisation 3D

Nous l'avons dit, la modélisation 3D des câbles en fil de Litz n'est pas chose aisée. C'est pourquoi cette approche est quasi inexistante dans la littérature. Citons toutefois le travail exposé en 2009 dans la publication [28] pour la modélisation en Éléments Finis de machines à aimants permanents. Dans cet article, les auteurs étudient divers arrangements du bobinage en fil de Litz et les pertes associées. Un modèle analytique permettant le calcul des courants dans chaque brin est également proposé. Cette étude montre notamment que le courant n'est pas toujours réparti de manière homogène entre les brins.

1.2.3 Etudes récentes

Il convient de terminer cette revue bibliographique en citant brièvement quelques publications récentes parues entre 2013 et 2014.

Dans l'article [29], les auteurs reprennent et comparent les méthodes présentées dans les publications [30] et [11] pour le calcul de la résistance en fonction de la fréquence d'un fil de Litz utilisé dans le cas d'un bobinage. L'étude permet de déterminer certains critères à prendre en compte dans la conception du fil. Les modèles sont également comparés à une modélisation Éléments Finis 2D ainsi qu'à des mesures. Les résultats donnés par tous les modèles correspondent aux mesures à basse fréquence. A haute fréquence, seul le modèle Éléments Finis reste cohérent tandis que les autres divergent largement.

Dans l'article [31], une étude basée sur des calculs Éléments Finis 2D évalue les pertes dans un inducteur planaire en spirale formé à partir de fil de Litz. Des mesures permettent de valider les simulations. Une autre étude en 2D est présentée dans l'article [32] dans laquelle les auteurs cherchent à évaluer la résistance d'un inducteur constitué d'une spire de forme ovale. Notons que les brins ont un diamètre de l'ordre du millimètre et que l'étude est menée pour des fréquences inférieures à 200 Hz. L'épaisseur de peau est donc grande devant la taille des brins. Enfin, dans la publication [33], un modèle semi-analytique basé sur l'estimation, grâce à un calcul en Éléments Finis 2D, d'une perméabilité magnétique équivalente est proposé afin d'estimer les pertes cuivre dans une bobine de transformateur.

Nous constatons que les quelques publications présentées ci-dessus adoptent toutes une approche 2D pour la modélisation des fils de Litz. Cela s'explique notamment par le fait que les géométries considérées présentent une symétrie axiale autorisant cette simplification.

Comme nous l'avons vu jusqu'ici, aucun des nombreux modèles présentés dans la bibliographie ne permet de prendre en compte l'influence de l'arrangement interne d'un inducteur multibrins sur les pertes en son sein – exception faite de l'étude présentée dans l'article [28] (voir partie 1.2.2). Néanmoins, certains auteurs commencent à s'intéresser à cette question.

Ainsi, dans l'article [34], les auteurs décrivent une méthode de prise en compte de la géométérie des brins grâce à une série de calculs Éléments Finis en 2D. Ils supposent que, en raison du torsadage, le projeté orthogonal sur une section du câble du chemin d'un brin dans le câble forme une ellipse. Ainsi, sur une certaine longueur de câble, les brins échangent leurs positions dans la section du câble : ceux se trouvant initialement sur l'extérieur du câble passent à l'intérieur et *vice versa*. Pour modéliser ce brassage, ils suggèrent d'effectuer plusieurs simulations en tenant compte de ces permutations dans les positions des brins, puis de calculer la résistance du câble en moyennant les résultats de ces simulations. Les auteurs proposent un calcul complet en Éléments Finis de la résistance d'un inducteur en fil de Litz similaire aux inducteurs de plaque de cuisson en considérant également l'influence des connecteurs situés aux extrémités de l'inducteur. Des mesures permettent de valider les résultats du modèle. Remarquons que les auteurs montrent que la distribution du courant est « très inhomogène » entre les brins. Cet article a donc le mérite de proposer une prise en compte du torsadage des brins. Notons, toutefois, que les brins sont supposés suivre des chemins relativement simples dans le câble, ce qui n'est probablement pas le cas dans tous les types de câbles multibrins. De plus, l'approche 2D tend toujours à masquer certains effets 3D de l'arrangement des brins.

Dans l'article [35], les auteurs proposent une méthode analytique permettant de déterminer les paramètres optimaux pour la conception d'un fil de Litz pour le bobinage d'un transformateur. Cette méthode hérite directement des nombreux travaux précédents des auteurs et ne tient pas compte de la géométrie 3D des brins. En revanche, dans la publication [36], ils s'intéressent directement à cette question. Par le biais de la méthode PEEC (*Partial Element Equivalent Circuit*), divers arrangements de fils de Litz sont modélisés et leurs comportements en fréquence sont calculétudiés. Les calculs sont effectués pour des portions de câble droites. La comparaison à des mesures montre la bonne précision de la modélisation. L'objet de cette publication correspond donc exactement au questionnement auquel nous avons avons travaillé et que nous exposons dans le présent manuscrit.

1.3 Le calcul hautes performances : une histoire de machines

Les outils que nous avons développé pour la modélisation d'inducteurs multibrins n'auraient pu voir le jour sans l'utilisation du calcul parallèle. L'avènement du calcul parallèle est le fruit de nombreuses innovations, tant du point de vue des

composants matériels que des langages de programmation et des logiciels. Dans cette partie, nous proposons de considérer brièvement les grandes lignes de l'évolution des capacités de calcul des machines jusqu'aux structures les plus modernes de calcul intensif ainsi que les principaux modèles de programmation mis en œuvre pour les utiliser.

Dans un premier temps, nous décrirons le développement des capacités de calcul parallèle du point de vue des machines. Ensuite, nous exposerons les grandes lignes des principaux modèles de programmation parallèle les plus répandus et nous argumenterons le choix de celui que nous avons adopté. Ces deux premières parties n'ont pas vocation à être exhaustives mais plutôt à introduire de manière intuitive les outils utilisés dans la suite ce manuscrit. Pour une description plus approfondie, mais tout à fait digeste, de l'histoire du calcul parallèle et des modèles de programmation associés, le lecteur pourra se référer aux chapitres 1 et 2 de la thèse de Jérôme Clet-Ortega [37].

1.3.1 Aux origines du calcul parallèle

Pendant longtemps, la puissance de calcul des ordinateurs était principalement fonction de la fréquence de leur unique processeur (ou *CPU* : *Central Processing Unit*). Cette course à la fréquence était nourrie des innovations régulières des fondeurs qui parvenaient à réduire les temps d'exécution de certaines instructions et à améliorer le traitement des instructions et des données. Deux grandes techniques de parallélisme émergèrent. Citons, tout d'abord, la technique du pipeline qui permet l'exécution simultanée d'instructions différentes : on parle alors de *parallélisme d'instructions*. D'autre part, les processeurs à architectures vectorielles permettent de traiter davantage de données grâce à une organisation matérielle particulière rendant possible le traitement simultané de données de même type (opérations sur des tableaux de nombres, par exemple) : dans ce cas, on parle de *parallélisme de données*.

Néanmoins, l'augmentation des pertes thermiques concomitantes à la montée en fréquence des processeurs a mis un sérieux frein aux avancées dans ce domaine. Depuis les années 2000, cette tendance à la course à la fréquence est beaucoup moins marquée et, si aujourd'hui certains processeurs haut de gamme passent la barre des 4 GHz, la majorité des processeurs grand public stagnent entre 2 et 3.5 GHz [38], en fonctionnement normal [2].

Grâce à une utilisation avancée des pipelines [39], le développement du *multithreading* permit de gagner encore en puissance de calcul en « permett[ant] l'exécution simultanée de plusieurs flots d'instructions indépendants sur un même pipeline » [40]. Ainsi, un seul processeur physique supportant deux threads simultanés

2. On parle ici de fréquence maximale de travail indiquée par le constructeur. Il est possible d'augmenter la fréquence de travail d'un processeur au-delà des spécifications standards grâce à la technique de l'*overlocking*, ou "sur-fréquencage", moyennant des dispositifs de refroidissements adaptés et une certaine prise de risque quant au fonctionnement et à la durée de vie des composants.

peut se comporter comme deux processeurs logiques, ou virtuels, du point de vue du système d'exploitation. Néanmoins, dans ce cas, le processeur n'est pas pour autant deux fois plus puissant. Le gain en performance dépend des tâches à effectuer et varie entre 15 % et 30 % [41], [42].

Pour accroître la puissance de calcul de leurs produits, et grâce aux progrès constants dans la miniaturisation des composants électroniques fondamentaux, les fondeurs ont maintenant pris le parti d'implanter davantage de processeurs physiques, ou *cœurs*, sur une même puce. Ainsi, en quelques années, la technologie du processeur monocœur, avec option *multithread*, a laissé la place aux CPU multicœurs, avec une option de *multithreading* [3].

1.3.2 Les supercalculateurs

Nous l'avons vu, l'évolution des techniques destinées à accroitre la puissance des ordinateurs a conduit ces derniers à intégrer des composants destinés à fonctionner en parallèle. Néanmoins, les structures permettant d'exécuter des calculs intensifs existent depuis longtemps. Et la concurrence est rude ! Ainsi, depuis juin 1993 [4], le projet TOP500 [43] publie deux fois par an le classement des 500 supercalculateurs les plus performants au monde. En juin 1993, le plus vieux supercalculateur référencé était un calculateur vectoriel VP-200 de Fujitsu hébergé au National Fusion Research du Japon datant de 1984. Ce calculateur monoprocesseur avait une puissance crête mesurée de 0.4 GFlops (théorique : 0.5 GFlops) [5]. Quant au calculateur en tête du classement ce même mois, il s'agissait d'un CM-5/1024 de Thinking Machines Corporation, achevé la même année, installé au Los Alamos National Laboratory aux États-Unis avec ses 1024 cœurs et une puissance crête mesurée de 59,7 GFlops (théorique : 131 GFlops).

Bien évidemment, les supercalculateurs ont profité largement des innovations apportées aux processeurs ainsi que des progrès réalisés dans les technologies des réseaux. En novembre 2013, le plus puissant calculateur était le chinois Tianhe-2 [6] du National Super Computer Center de Guangzhou, construit par la National University of Defense Technology et achevé la même année. Avec ses 3.12 millions de cœurs de calcul, il affiche une puissance crête théorique de 54.9 PFlops, mesurée à 33.9 PFlops.

3. Par défaut, la plupart des sytèmes d'exploitation n'activent pas la technologie *multithread*. Ce choix est généralement laissé à l'appréciation de l'utilisateur qui peut le paramétrer *via* le *BIOS*.

4. Pour ne pas remonter plus loin dans le temps... En effet, on estime souvent que l'histoire des supercalculateurs débute dans les années 1960, notamment avec Seymour Cray, initialement ingénieur de la société Control Data Corporation et concepteur en 1964 du CDC 6600, considéré comme le premier supercalculateur dévolu au calcul scientifique. En 1972, Seymour Cray fonda la société portant son nom.

5. Flops = *FLoating point Operations Per Second*, unité indiquant le nombre d'opération à virgule flottante réalisables en une seconde par un système informatique. La mesure de la puissance de calcul pour le classement TOP500 s'effectue grâce au *benchmark* LINPACK [44].

6. MilkyWay-2, en anglais.

Pour atteindre de telles performances, les supercalculateurs sont construits avec une infrastructure particulière. Ils sont constitués d'une grappe de machines plus petites et plus communément appelée *cluster*. Chaque machine est appelée *nœud*. Un cluster est essentiellement constitué d'un nœud maître, de nœuds esclaves, de nœuds de stockage, le tout connecté par un réseau haut débit. Le nœud maître, ou *frontale*, est connecté à Internet et permet aux utilisateurs de soumettre leurs calculs, ou *jobs*. Ces *jobs* sont placés en file d'attente par un outil logiciel, l'ordonnanceur de tâches, ou *scheduler*, qui est chargé de répartir la charge de travail de la file sur les nœuds esclaves à sa disposition. Comme leur nom l'indique, ce sont les nœuds esclaves qui effectuent les calculs à proprement parler. L'ensemble des données, qu'elles soient importées par les utilisateurs en tant que données à traiter par les *jobs* ou qu'elles soient générées par ceux-ci, sont hébergées sur les nœuds de stockage qui offrent une mémoire disque conséquente. Quant à l'interconnexion, elle est généralement constituée de deux types de réseaux : un réseau très-haut débit interconnectant les nœuds esclaves afin de minimiser les temps de communication entre ces derniers, et un réseau plus « lent », moins coûteux, connectant les nœuds esclaves, le nœud maître et les nœuds de stockage. La figure 1.1 présente un schéma simplifié de l'infrastructure de base d'un cluster.

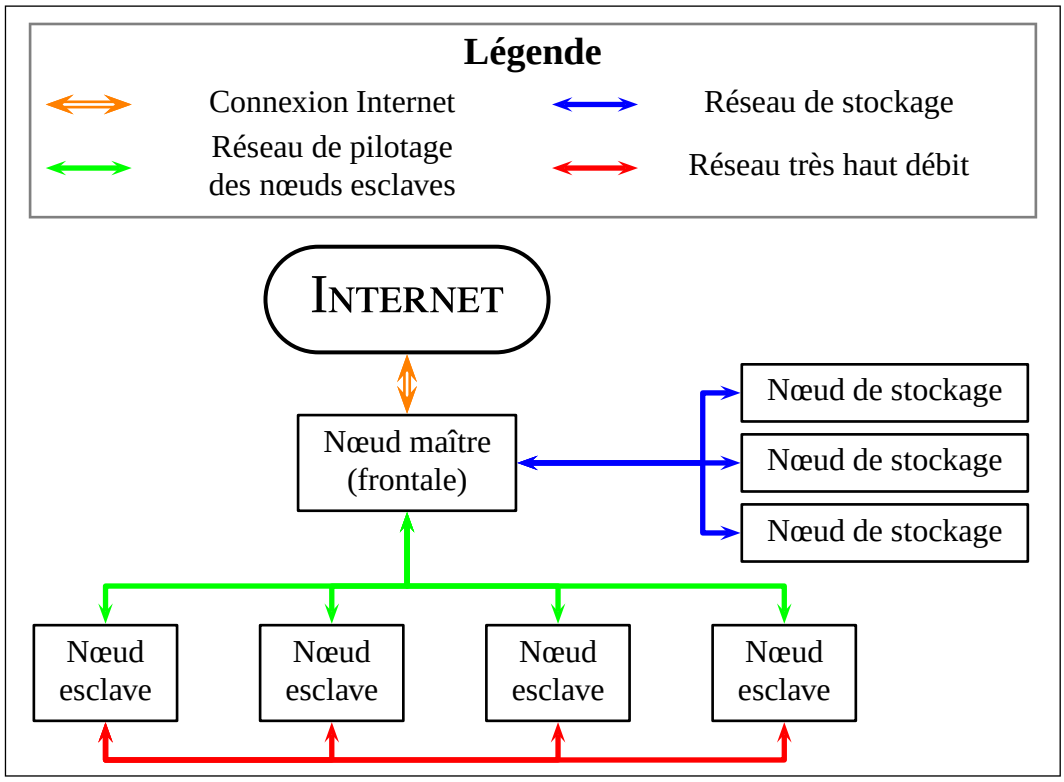


FIGURE 1.1 – Schéma simplifié de l'infrastructure d'un cluster

Si les supercalculateurs ont pendant longtemps profité largement des innovations apportées aux CPU, d'autres innovations non-négligeables et plus récentes ont permis d'atteindre les performances citées plus haut.

1.3.3 Les accélérateurs matériels

Les CPU ne sont pas les seuls composants d'un ordinateur contenant des processeurs. Les CPU sont des unités de calcul « à tout faire », mais il existe d'autres processeurs plus spécialisés.

Ainsi, à la différence des CPU, dont le travail est essentiellement dévolu à l'exécution des programmes, les GPU (ou *GPU* : *Graphics Processing Unit*, processeurs graphiques) ont été à l'origine conçus, comme leur nom l'indique, pour gérer l'affichage à l'écran et sont implantés sur les cartes graphiques. L'architecture matérielle d'un GPU diffère grandement de celle d'un CPU en ce qu'elle est intrinsèquement hautement parallèle. Si, à l'heure actuelle, la majorité des CPU ont un nombre de cœurs inférieur à 20, les GPU en comptent plusieurs centaines. Si l'augmentation de la résolution des écrans a nécessité la conception de cartes graphiques plus puissantes, c'est surtout la forte croissance du marché du jeu vidéo et les efforts menés pour améliorer la qualité des rendus graphiques qui ont dopé le développement des GPU. En effet, et pour ne citer que cet exemple, la quantité de calculs demandée pour générer des images 3D animées en haute résolution est conséquente. Ajoutons à cela l'impératif de la fluidité de l'image. Les calculs doivent donc être réalisés dans un minimum de temps d'où le recours à une architecture massivement parallèle.

L'évolution de ce matériel ainsi que son excellent rapport performances/coût ont favorisé l'intérêt de la communauté scientifique pour les GPU, ce qui a conduit à l'avènement du *GPGPU* (*General-Purpose computing on Graphics Processing Units*) qui consiste à détourner les capacités de calculs des GPU de leur usage initial pour des usages beaucoup plus larges. Devant les possibilités offertes par les GPU, la plupart des constructeurs cherchent désormais à les rapprocher physiquement des CPU en faisant un effort d'intégration. Ainsi, les GPU jouent de plus en plus le rôle d'accélérateurs de calculs. En 2010, AMD dévoilait le premier *APU* (*Accelerated Processing Unit*), prénommé Llano, résultat de l'intégration d'un CPU et d'un GPU sur une même puce [45]. Depuis, d'autres architectures ont été présentées comme Haswell d'Intel, en 2013 [46] ou Kaveri d'AMD au CES 2014 [47], pour ne citer que les innovations les plus récentes des deux géants du domaine.

Aujourd'hui, les GPU font partie intégrante des systèmes de calcul intensif[7]. Certains systèmes de cluster intègrent en partie des nœuds formés de GPU. Notons au passage que les GPU ne sont pas les seuls systèmes d'accélérateur existants. En effet, si un CPU est capable de traiter à peu près n'importe quel type d'instruction, sa polyvalence a pour prix une certaine lenteur de traitement. C'est pourquoi il existe des circuits appellés *coprocesseurs* qui sont spécialisés dans le traitement de certaines tâches et donc plus efficaces que le CPU lui-même. Ainsi, le GPU, dans son utilisation standard, peut être considéré comme un coprocesseur graphique. Il existe des coprocesseurs arithmétiques. Les coprocesseurs jouent donc également un rôle important dans les systèmes *HPC*. A titre d'exemple, le supercalculateur

7. L'abréviation anglaise *HPC* pour *High Perfomance Computing* est couramment employée.

Tianhe-2 est constitué de 16000 nœuds comptant chacun deux processeurs Intel Xeon E5-2692 à 12 cœurs et trois coprocesseurs Intel Xeon Phi 31S1P de 57 cœurs chacun [48].

1.3.4 Les limites des supercalculateurs

Nourri de toutes les innovations décrites précédemment, le monde du calcul intensif a vu la construction de systèmes toujours plus puissants pour répondre aux besoins grandissants en capacité de calcul provenant tant du monde de la recherche académique que de l'industrie. Ainsi, en 20 ans, la puissance de calcul théorique des supercalculateurs a augmenté d'un facteur 10^8. Néanmoins, l'acquisition de telles infrastructures constitue un investissement conséquent. En effet, outre les installations dévolues directement aux calculs (nœuds frontaux et de calculs, réseaux, nœuds de stockage) une telle machine intègre également des systèmes de secours destinés à accroître la tolérance aux pannes et des systèmes de refroidissement indispensables au bon fonctionnement des composants. Ainsi, rien que d'un point de vue purement matériel, la facture est importante. A titre d'exemple, les deux calculateurs les plus puissants figurant au TOP500 de novembre 2013, le chinois Tianhe-2 et l'américain Titan de Cray, avoisinent chacun les 100 millions de dollars US [49], [50], sans compter le budget à réserver à la consommation énergétique et aux nécessaires opérations de maintenance. De plus, de par sa structure, l'évolution d'un supercalculateur n'est pas chose aisée.

A l'aspect financier, s'ajoute une autre contrainte : un supercalculateur est un outil très spécifique. En effet, chaque supercalculateur est unique. Cette spécificité en fait à la fois une force et une faiblesse. Une force, par la puissance de calcul offerte, puisqu'il est optimisé dans ce but. Une faiblesse, en ce qu'un supercalculateur est bâti autour d'une architecture propre, piloté par un système de logiciels propre et avec son propre mode de fonctionnement. De ce fait, il peut être difficile de fédérer les capacités de calculs de plusieurs de ces machines. Remarquons toutefois que, face à cette contrainte forte, des solutions alternatives ont vu le jour et, aujourd'hui, si cette spécificité peut encore exister sur certaines machines, des techniques éprouvées de mise en commun des ressources ont été élaborées.

1.3.5 Le parallélisme 2.0

Certains domaines de recherche sont particulièrement demandeurs en puissance de calcul et en espace de stockage car producteurs de grandes quantités de données et le recourt à une solution de type supercalculateur n'est alors pas toujours une solution viable en raison d'un coût trop élevé. Des solutions alternatives ont donc été envisagées en prenant le contrepied du fonctionnement des supercalculateurs : au lieu de centraliser le travail sur une très grosse machine, certes performante, celui-ci est réparti entre plusieurs machines aux prétentions plus modestes. C'est

la logique du calcul partagé [8]. Ce nouveau paradigme dans le monde du *HPC* doit son apparition au formidable développement d'Internet, depuis le début des années 1990, ainsi qu'aux développements connexes de la technologie des réseaux. On peut distinguer deux grandes approches de calcul réparti.

La première est appelée *Volunteer computing* et fait appel, par définition, à la contribution volontaire de n'importe quel internaute. C'est le cas, par exemple, de la plate-forme open-source BOINC [51]. Le principe général d'un projet BOINC est d'envoyer aux contributeurs des paquets de données à traiter. Côté contributeur, le logiciel BOINC traite les données qu'il a reçues, puis les renvoie. Un nouveau paquet de données lui est alors envoyé, et ainsi de suite. L'astuce consiste à exploiter les ressources que le contributeur n'utilise pas (cycles de CPU inutilisés, temps de veille de la machine,...), sans le gêner dans l'usage personnel de son ordinateur. Si, en termes de matériel, les projets de ce type ne se basent pas sur une architecture optimisée pour le calcul intensif, c'est bien sur le nombre important de machines disponibles que repose l'intérêt d'une telle approche [9].

Le calcul sur grille, ou *Grid computing* constitue un autre paradigme. On peut faire remonter la genèse des grilles de calcul aux travaux de Foster et Kesselman [54] qui en jettent les bases. Une *grille* est un ensemble de machines connectées *via* un réseau et mettant en commun leurs ressources. La grille est pilotée par l'intermédiaire d'un *scheduler*. Contrairement au *Volunteer computing*, le *Grid computing* cherche à exploiter aux maximum l'ensemble des ressources disponibles. Une grille de calcul constitue donc une structure pensée *HPC*, avec une attention accrue portée aux performances des machines et des réseaux. A titre d'exemple, le projet WLCG est l'un des plus importants projets de grille de calcul existants [10].

Ainsi, quelle que soit l'approche employée, le calcul partagé permet de palier les limites inhérentes aux supercalculateurs traditionnels. Tout d'abord, côté matériel, les ressources d'un supercalculateur sont fixes et difficiles à faire évoluer. En revanche, les structures de calcul distribué présentent deux avantages majeurs. Premièrement, elles sont conçues pour être tolérantes à l'*hétérogénéité* des infrastructures qui leur sont connectées : ordinateurs personnels ou centres de calculs (clusters,...). Les réseaux reliant les machines sont également hétérogènes (Internet classique, réseaux plus haut-débit). Autre point fort, la *scalabilité* du système : il supporte davantage l'ajout de nouvelles machines ou le retrait de machines obsolètes ou en panne. De par leur conception, les systèmes de calcul répartis ont donc la capacité d'évoluer et d'inclure des machines aux caractéristiques variées.

Enfin, côté investissement, un système distribué peut être plus attrayant qu'un supercalculateur. Dans le cas d'une grille, si l'investissement global peut être conséquent, il est plus facilement réparti entre les partenaires qui peuvent être nombreux

8. Ou encore calcul distribué, ou réparti.

9. En juin 2013, la puissance moyenne journalière des calculs réalisés *via* BOINC atteignait 7.369 PFlops grâce à près de 440 000 ordinateurs [52], [53].

10. Le projet *Worldwide LHC Computing Grid* est dévolu à l'analyse des données issues des collisions de particules à hautes énergies produites au LHC (Large Hadron Collider) [55], [56].

et qui, de surcroît, apportent chacun leurs propres moyens de calcul existants. Dans le cas des structures de type *Volunteer computing*, il est d'autant plus réduit, l'essentiel du système reposant sur le volontariat des contributeurs ainsi que sur des réseaux déjà construits (Internet essentiellement).

Notons toutefois que les systèmes de calculs répartis sont essentiellement efficaces pour effectuer des calculs hautement parallélisables, c'est-à-dire ayant une faible interdépendance. C'est le cas, par exemple, pour le traitement des événements issus de la chaine d'acquisition du LHC, chaque événement étant indépendant des autres. Par contre, un programme contenant une proportion non-négligeable de calculs parallélisables mais interdépendants (inversion de matrice, par exemple) sera traité plus efficacement par un supercalculateur que par un système distribué, en raison de la plus grande rapidité du réseau entre ses nœuds.

1.3.6 Classification des machines

Nous l'avons vu, l'évolution des techniques a conduit à l'élaboration de systèmes de calculs toujours plus performants et variés. Toutes ces architectures peuvent être classées suivant la classification de Flynn [57], certes ancienne, mais toujours valable. Cette classification répartit les architectures des machines en fonction de deux critères : les types des flux de données traitées, et les types des flux d'instructions appliquées aux données. Un flux de données peut être de deux types : unique, *Single Data stream*, ou multiples, *Multiple Datas stream*. De même, les types de flux peuvent être de type unique, *Single Instruction stream*, ou multiples, *Multiple Instructions stream*. La figure 1.2 donne un aperçu de la classification de Flynn.

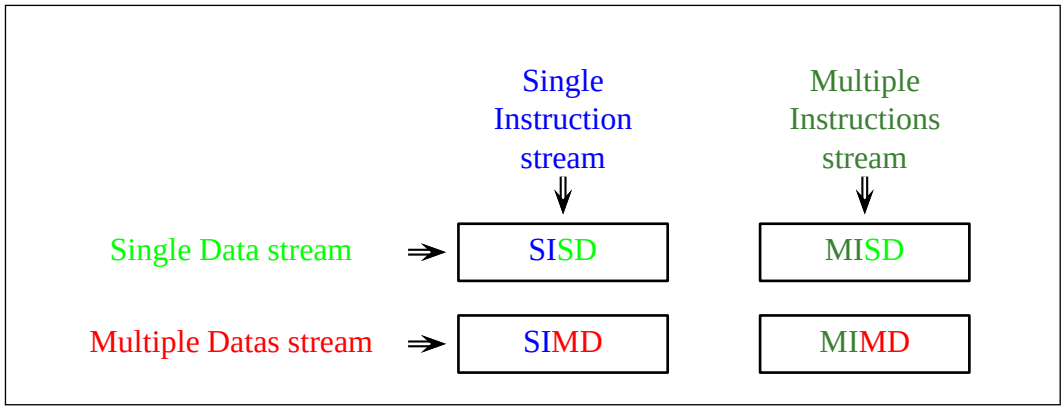


FIGURE 1.2 – Classification de Flynn

Le mode SISD (*Single Instruction on Single Data*) correspond au fonctionnement d'un ordinateur séquentiel : une instruction exécutée à la fois sur une case mémoire à la fois.

Dans le mode MISD (*Multiple Instructions on Single Data*) la même donnée est traitée simultanément par plusieurs flux d'instruction. Ce mode correspond à des

architectures particulières. Il est couramment admis que le pipeline est une architecture MISD [58], [58], [59], [60]. On peut toutefois noter que certains réfutent cette idée car, si les étages d'un pipeline sont exécutés simultanément, chaque étage traite une donnée différente des autres étages [61]. Pour ces derniers, les systèmes de contrôles redondants, comme les contrôleurs de vol d'avion ou de fusée, par exemple, correspondent davantage à une architecture MISD. En effet, dans ces systèmes, une donnée unique est traitée simultanément par plusieurs calculateurs *a priori* identiques. En théorie, les résultats délivrés par les calculateurs doivent donc être identiques. La comparaison de ces sorties permet donc de palier la possible défaillance d'un de ces systèmes de calcul et la fiabilité du contrôleur est alors renforcée.

Les machines SIMD (*Single Instruction on Multiple Datas*) permettent d'appliquer simultanément la même instruction sur une série de données et correspondent typiquement au fonctionnement des GPU.

Enfin, le mode MIMD (*Multiple Instructions on Multiple Datas*) est le plus généraliste : des instructions différentes sont appliquées simultanément à des données différentes. C'est le mode de fonctionnement des ordinateurs multiprocesseurs ainsi que des structures plus importantes évoquées précédemment comme les supercalculateurs et les grilles de calculs.

Pour résumer, dans le domaine du calcul intensif, la majorité des architectures utilisées sont de type SIMD (GPU,...) ou MIMD (systèmes multicœurs,...). On trouve des structures hybrides comprenant des systèmes de ces deux types : certains clusters proposent à la fois des nœuds de calculs à base de CPU multicœurs et des nœuds à base de GPU.

1.3.7 Principaux modèles de programmation parallèle

Pour pouvoir profiter des capacités de calcul offertes par les architectures parallèles, il a fallu élaborer des modèles de programmations adaptés. Ces modèles de programmation ont pour objectif de permettre le partage des calculs en tâches réparties entre les différentes entités de calcul disponibles (cœurs de processeurs, réseau de stations de travail, clusters,...) et de piloter les échanges de données entre ces tâches. Le choix d'un modèle de programmation pour l'élaboration d'un programme parallèle dépend donc, entre autres choses, de l'architecture matérielle sur laquelle ce programme sera exécuté.

De part leur nature, les architectures SIMD comme les GPU nécessitent des modèles de programmation spécifiques. Ainsi, pour ne citer que les deux plus grands acteurs du domaine, Nvidia et AMD proposent respectivement les interfaces de programmation CUDA (*Compute Unified Device Architecture*) [62] et CAL [63] pour l'utilisation de leurs cartes graphiques. La portabilité d'un code n'est donc pas assurée par ces outils. Notons toutefois que des efforts sont menés pour remédier à cela. Le consortium Khronos Group [64] propose le standard OpenCL [65] qui supporte différentes architectures tant de type SIMD que MIMD. Néanmoins, l'utilisation de

ce type d'approche reste ardue car elle nécessite toujours de tenir compte de manière forte de l'architecture matérielle. Nous allons donc nous concentrer par la suite sur les modèles de programmation adaptés aux architectures de type MIMD. En effet, ces derniers sont d'une utilisation plus souple que ceux requis par les architectures SIMD et présentent une portabilité plus importante.

Bien qu'il en existe davantage, la plupart des solutions logicielles de calcul numérique actuelles sont basées essentiellement sur l'un des deux modèles de programmation suivants : OpenMP ou MPI. Il existe aussi des solutions hybrides employant à la fois OpenMP et MPI.

Le standard OpenMP (*Open Multi Processing*) [66] propose une interface de programmation [11] permettant la création d'un code parallèle ou la parallélisation d'un code préexistant. Le parallélisme est implémenté en insérant des directives de compilation qui permettent, à l'exécution du code, de paralléliser certaines parties du programme entre un nombre de tâches, ou *threads*, défini par l'utilisateur [67]. Outre la portabilité, l'intérêt d'OpenMP réside essentiellement dans la simplicité de son utilisation, particulièrement appréciable lorsqu'il s'agit de paralléliser un code existant, car il ne nécessite généralement pas de modifier ce dernier en profondeur. Il est toutefois important de noter que le modèle de programmation offert par OpenMP est adapté pour des architectures à mémoire partagée : par exemple, une station de travail, dans laquelle les différents processeurs multicœurs ont accès à l'ensemble la même mémoire RAM. Il n'est donc pas possible d'utiliser un code parallélisé avec OpenMP sur un cluster, puisque chaque nœud dispose de sa propre mémoire, à moins d'avoir recours à un outil de virtualisation, mais les performances peuvent en être affectées. Ceci est également le cas sur tout autre système à mémoire distribuée, comme un réseau de stations de travail. Dans ce cas, l'emploi de la bibliothèque MPI est plus indiqué. C'est d'ailleurs vers elle que nous nous sommes tournés.

1.3.8 Focus sur MPI

Les bases de la bibliothèque MPI (*Message Passing Interface*) ont été présentées en 1993 [68]. Elle a depuis été enrichie et la version 3.0 du standard MPI est disponible depuis 2012 [69]. Le standard MPI présente de nombreux atouts. Premièrement, il propose un ensemble de fonctionnalités permettant la communication entre plusieurs processus. Ces processus peuvent travailler tant sur une machine à mémoire partagée que sur une architecture à mémoire distribuée. Ceci constitue donc un avantage important du standard MPI par rapport à OpenMP. En effet, grâce à cela, il est possible de concevoir un code de calcul parallèle capable de fonctionner sur plusieurs machines simultanément et ainsi d'accroitre à la fois le nombre de processeurs et la quantité de mémoire utilisable en additionnant les ressources de ces machines.

11. Couramment dénommée également par l'abréviation API pour *Application Programming Interface*.

Deuxièmement, cette librairie a été implémentée pour la plupart des systèmes d'exploitation et la plupart des architectures. Elle est donc facilement portable et peut fonctionner sur des architectures hétérogènes. On trouve ainsi plusieurs implémentations optimisées pour des systèmes d'exploitation et des technologies variées [70]. La plupart de ces implémentations sont issues de l'une ou l'autre des deux implémentations principales de MPI : Open MPI [71] et MPICH2 [72]. Dans notre cas, nous avons choisi MPICH2 qui implémente la version 2.0 du standard MPI.

Enfin, MPI permet de gérer des algorithmes parallèles comptant un très grand nombre de processus (plusieurs centaines). La scalabilité d'un code, c'est-à-dire sa capacité à être exécuter sur un nombre très varié de processus, est ainsi assurée.

Fonctionnement général d'un programme parallélisé avec MPI

Avant d'entrer dans le cœur de la programmation parallèle *via* MPI, nous décrivons dans les grandes lignes le fonctionnement d'un programme conçu avec cette bibliothèque. Nous allons l'illustrer avec un exemple simple.

Soit un programme appelé `Sum.exe` utilisant MPI et qui permet de calculer la somme des éléments d'un tableau. Quel que soit le système d'exploitation, l'exécution de ce programme peut être lancée *via* une ligne de commande par la commande suivante [12] :

```
mpiexec -n 8 ./Sum.exe
```

L'option `-n`, ou `-np`, permet d'indiquer le nombre de processus MPI qui vont exécuter `Sum.exe`, ici huit processus. Au lancement de cette commande, huit processus exécutant chacun *sa propre copie* de `Sum.exe` sont donc créés. Ceci est à noter : le programme est bien exécuté huit fois simultanément et chacun des processus utilise sa propre mémoire. Ce dernier point peut poser certains problèmes quand le programme nécessite un espace mémoire important puisque, suivant la programmation réalisée, la quantité de mémoire demandée peut être proportionnelle au nombre de processus lancés.

Maintenant, comment les calculs sont-ils répartis entre les processus ? Le principe de base de la parallélisation *via* MPI est, comme son nom l'indique, de mettre en place un contexte de communication, ou *communicateur*, permettant l'échange de messages entre les processus appartenant à ce contexte. Un communicateur par défaut regroupant l'ensemble des processus MPI est créé de manière automatique, au lancement du programme. Ce communicateur porte le nom **`MPI_COMM_WORLD`**. Il est possible de créer des sous-communicateurs. Au sein d'un communicateur, chaque processus est identifié par un rang, un indice qu'il est le seul à porter dans ce communicateur. Il est donc possible d'indiquer à chaque processus ce qu'il doit faire en utilisant son rang comme indicateur de la portion de programme qu'il doit

12. Le lancement du programme *via* un logiciel ordonnanceur de tâches spécifique (*scheduler*), comme dans le cas d'un cluster par exemple, requiert une procédure propre à ce logiciel mais qui ne sera pas abordée dans ce manuscrit.

exécuter. On peut donc, si besoin, demander à tous les processus d'exécuter exactement les mêmes calculs ou, à l'inverse, leur demander d'exécuter chacun un code totalement différent.

Mais comment les processus communiquent-ils ? MPI dispose d'un ensemble de fonctions rendant possible les communications entre processus. Elles permettent d'échanger des données entre processus ainsi que des signaux afin de piloter les calculs (synchronisation, signaux d'erreur,...). *A minima*, une communication nécessite un expéditeur, un destinataire, un message et un communicateur. Mais, il existe aussi des communications dites *collectives* qui permettent de communiquer avec tout ou partie d'un communicateur, autrement dit avec plusieurs processus en même temps.

En résumé, dans notre exemple, huit processus MPI exécutant `Sum.exe` sont lancés simultanément. Ceux-ci sont regroupés au sein du communicateur par défaut. Le partage des calculs est paramétré par le rang des processus et les échanges se font grâce aux communications dans le communicateur **`MPI_COMM_WORLD`**.

Nous avons évoqué la possibilité d'exécuter un programme parallèle sur une architecture à mémoire distribuée. Prenons l'exemple d'un réseau de stations de travail sur lequel se trouvent trois machines `machine01`, `machine02` et `machine03`. Pour exécuter le programme `Sum.exe` sur 12 processus avec quatre processus sur chaque machine, il suffit, en principe [13], de lancer la commande suivante :

```
mpiexec -n 12 -machinefile ./machines ./Sum.exe
```

L'option `-machinefile` indique à `mpiexec` la liste des machines sur lesquelles on souhaite lancer le calcul. Ici ce fichier est nommé `machines`. On y retrouve les noms par lesquels les machines sont identifiées sur le réseau ainsi que le nombre de processus que l'on souhaite exécuter sur chacune d'elles. Ce nombre peut être différent suivant les capacités de chaque machine. Le format du fichier est le suivant :

```
machine01 : 4
machine02 : 4
machine03 : 4
```

La numérotation des rangs des processus sera effectuée selon l'ordre d'attribution sur les machines : ici, les processus 0 à 3 sur `machine01`, 4 à 7 sur `machine02`, et ainsi de suite. Notons que, pour que le lancement puisse être effectué tel que décrit précédemment, le programme `Sum.exe` doit être situé sous le même chemin absolu sur chaque machine.

Pour une analyse détaillée du cœur du code du programme dont nous venons de décrire le fonctionnement, on pourra se référer à l'annexe A.

13. Remarque importante : la commande présentée ici ne fonctionne pas systématiquement telle qu'elle sur tous les systèmes. Dans certains cas, l'option `-disable-hostname-propagation` peut être ajoutée pour laisser MPI rechercher automatiquement le nom de la machine depuis laquelle le calcul a été lancé, appelée *hôte*. La commande `mpiexec` propose de nombreuses autres options permettant de paramétrer le lancement et l'exécution du programme.

Vous avez-dit *processus* ?

Il nous paraît utile d'attirer ici l'attention sur un point important. On notera que, jusqu'à présent, dans cette partie consacrée à MPI, nous avons parlé de *processus* et non de *processeur*. Dans le langage courant, il existe visiblement une certaine confusion quant à ce qui différencie les entités désignées par ces termes.

Nous avons effectivement souvent constaté que le terme *processeur* est considéré comme synonyme de *processus*. Ainsi, on entend dire : « J'exécute ce programme en parallèle sur dix processeurs », et, souvent, dans l'esprit du locuteur, le programme est exécuté sur dix processeurs physiques. En fait, ceci constitue un abus de langage. La confusion qui en découle est également renforcée par le vocabulaire employé par la plupart des logiciels commerciaux qui utilisent fréquemment le terme *processeur* là où *processus* serait plus pertinent.

En effet, le terme *processeur* désigne un composant : un cœur d'un CPU (que ce soit un cœur physique ou virtualisé grâce au *multithreading*). En revanche, le terme *processus* se réfère à un programme : tous les programmes tournant sur une machine sont des processus.

Ainsi, plusieurs processus peuvent être exécutés par un seul processeur. Si une machine compte huit processeurs, paralléliser un programme A sur huit processus ne signifie donc pas forcément que les huit processeurs seront utilisés. Tout dépend de la charge de la machine. Si les huit processeurs sont faiblement chargés, alors chacun des processeurs recevra effectivement un processus.

En revanche, si quatre processeurs sont déjà occupés par l'exécution d'un autre programme B, il n'est pas certain que les huit processus de A seront répartis équitablement. Enfin, quand bien même cela serait le cas, chacun des quatre processeurs sur lesquels tourne déjà un processus issus du programme B ne serait pas entièrement occupé par le processus qu'il recevrait du programme A, mais il partagerait son temps de calcul entre ces deux processus. Dans tous les cas, cela ralentirait l'exécution des programmes A et B.

En ce qui nous concerne, nous faisons donc bien la différence entre un *processeur* (entité *hardware*), qui est un cœur de CPU, et un *processus*, un programme (entité *software*).

Quelques remarques sur les communications

Dans un programme parallèle, les communications jouent un rôle clé et ont un impact direct sur les performances. C'est pourquoi il est important de s'attarder sur leur fonctionnement.

Il existe deux types de communications : les unes sont dites *bloquantes*, les autres *non-bloquantes*. Tous les processus impliqués dans une communication bloquante voient leur exécution suspendue, en l'attente d'un signal indiquant que tous ont réalisé leur part de communication. Du point de vue du programme, une communication de ce type a le même effet qu'une barrière de synchronisation appliquée

aux processus qu'elle met en relation. Par exemple, les fonctions de communications **MPI_Send** et **MPI_Recv**, permettant respectivement l'envoi et la réception de données entre un processus émetteur et un processus destinataire, sont toutes deux bloquantes.

Les communications non-bloquantes, à l'inverse, permettent aux processus d'effectuer leur part de communication sans attendre de signal de confirmation des autres processus impliqués. Ainsi, par exemple, les processus émetteurs peuvent effectuer l'envoi des données puis, sans ce soucier de leur réception, poursuivre leur travail. Ce type de communication est particulièrement intéressant car il permet d'utiliser efficacement le temps d'attente qui serait dû à une communication bloquante en effectuant des calculs. On parle alors de *recouvrement* des communications par des calculs. L'exécution du programme se trouve donc accélérée. Néanmoins, une communication non-bloquante peut être employée seulement si le contenu des données envoyées ne risque pas d'être modifié par les calculs recouvrant la communication. Dans le cas contraire, les données que l'on souhaite envoyer risquent d'être corrompues par ces calculs. Pour éviter cela, la fonction **MPI_Wait** permet d'attendre *a posteriori* qu'une communication non-bloquante se termine avant de poursuivre le programme. Les communications non-bloquantes sont distinguées des communications bloquantes par le préfixe "I". La fonction **MPI_Isend**, par exemple, est le pendant non-bloquant de **MPI_Send**.

Quelques règles simples

Nous pouvons établir quelques règles et principes de base pour l'implémentation d'un programme parallèle utilisant MPI. Tout d'abord, pour pouvoir utiliser cette bibliothèque, il faut inclure le *header* `mpi.h`.

La fonction **MPI_Init** permet d'initialiser le contexte de communication MPI. Il est donc essentiel d'effectuer cette étape avant toute communication, sans quoi le programme retourne une erreur. De même, l'usage de la fonction **MPI_Finalize** permet la fermeture propre du contexte.

Chaque processus peut effectuer un code différent des autres. Néanmoins, la concordance des communications doit être assurée. Par exemple, quand le code d'un processus A prévoit un envoi de données vers un processus B, ce dernier doit implémenter la réception correspondante. Cet aspect relève de la responsabilité entière du programmeur.

La compilation d'un programme C++ implémenté avec MPI doit être effectuée grâce à la commande `mpic++`. Quant à l'exécution du programme, elle est lancée *via* la commande :

```
mpiexec -n [nbr_procs] [other_MPI_options] program_name
[program_options]
```

1.4 Conclusion

Dans ce chapitre, nous avons rappelé l'essentiel des modèles existants dans le domaine de la modélisation de câbles multibrins. Nous avons constaté que de nombreux modèles analytiques ont été développés. Ces modèles sont employés pour calculer les pertes dans des câbles ayant une géométrie présentant une certaine symétrie axiale (bobines de transformateurs, inducteurs de plaques de cuisson,...). De plus, une grande majorité de ces modèles s'appuie sur l'hypothèse du fil de Litz idéal qui stipule que les brins sont arrangés de telle sorte que le courant se réparti uniformément entre eux.

Des modèles numériques utilisant la Méthode des Éléments Finis ont également été présentés. Ceux-ci s'appuient sur le calcul de propriétés électriques équivalentes afin de simplifier la description de la géométrie et réduire la taille du maillage. Ces propriétés sont sensées porter l'influence de la géométrie complexe de laquelle elles sont déduites. Nombre de ces modèles d'homogénéisation sont basés sur des calculs en 2D ou 2D-axi. De plus, la portion de géométrie à partir de laquelle sont calculées les propriétés équivalentes doit présenter une structure périodique au motif facilement identifiable.

Cette revue bibliographique nous a permis de conclure que peu de modèles tiennent compte de l'arrangement 3D de câbles multibrins. Nous insistons sur le fait qu'une forte proportion de ces modèles, ainsi que les mesures permettant de les valider, ont été établis pour des géométries particulières (bobines de transformateurs, inducteurs de plaques de cuisson,...). Or, les inducteurs de type industriels présentent des géométries très différentes. Nous n'avons pas trouvé de modèle de câble adapté à la description de ce type d'inducteur.

Seul le modèle présenté dans les articles [21] et [22] tient compte de l'arrangement 3D des brins dans un cas très particulier dans lequel le chemin des brins peut être décrit de manière analytique. Nous n'avons pas trouvé de modèle permettant de décrire, analytiquement ou numériquement, les chemins que suivent les brins dans un cas général. Nous avons toutefois noté que des publications récentes abordent la modélisation 3D de fil de Litz dans des cas simples.

Afin de parvenir à réaliser la modélisation électromagnétique d'inducteurs multibrins, nous avons donc développé des solutions innovantes qui ont abouties à la conception d'un logiciel de calcul parallèle. Dans la suite de ce manuscrit, nous décrirons le modèle numérique de calcul des chemins des brins que nous avons établi. Ensuite, nous détaillerons la méthode numérique permettant d'effectuer les calculs électromagnétiques que nous avons implémentée. Cette méthode sera détaillée pour deux modèles d'inducteurs : un modèle dit *volumique*, permettant de réaliser des calculs locaux sur des câbles à faible nombre de brins (<60 brins) et un modèle dit *filiforme*, destiné à réaliser des calculs globaux pour la modélisation de câbles comptant un grand nombre de brins (>1000 brins).

Dans ce chapitre, nous avons également établi un historique montrant l'essentiel des innovations ayant permis le développement du calcul intensif. Nous avons pré-

senté la bibliothèque MPI et les grandes lignes du fonctionnement d'un programme parallèle implémenté grâce à celle-ci. Nous avons ainsi posé les bases des outils que nous avons employés dans l'implémentation de notre logiciel de modélisation des inducteurs multibrins.

Chapitre 2

Modélisation électromagnétique des inducteurs multibrins

La modélisation électromagnétique d'inducteurs multibrins se heurte à plusieurs problèmes de taille.

Tout d'abord, ces objets présentent une structure interne particulièrement complexe. La géométrie d'un brin est impossible à décrire analytiquement ou à construire à l'aide d'éléments géométriques simples. La description de la géométrie par la plupart des logiciels commerciaux standards n'est donc pas chose aisée, si tant est qu'elle soit possible.

Outre ce premier point, la modélisation électromagnétique d'inducteurs multibrins se confronte à un second problème tout aussi ardu : le choix de la méthode numérique. Comme nous l'avons évoqué dans la section 1.2.1, les méthodes analytiques ne sont utilisables que dans certains cas spécifiques et pour des géométries simples présentant des propriétés particulières, notamment en termes de symétries.

En ce qui concerne la Méthode des Eléments Finis (voir partie 1.2.2), sa principale limite réside dans le fait qu'elle requiert le maillage de l'espace entre les parties électromagnétiquement actives – conductrices – de la géométrie. Si le rapport d'échelle entre les divers objets de la géométrie est important, le maillage peut rapidement devenir conséquent. Ceci se vérifie notamment dans le cas des inducteurs multibrins. Ces structures comptent plusieurs centaines, voire plusieurs milliers de brins d'un diamètre de l'ordre de quelques dizaines de microns et espacés de quelques microns seulement, contre un diamètre total de l'inducteur de l'ordre de quelques centimètres. Le maillage d'une telle géométrie est donc extrêmement volumineux et, de surcroît, il est particulièrement difficile d'en contrôler la qualité.

Si une partie de la géométrie présente une structure périodique (voir partie 1.2.2), des études ont montré la possibilité de mettre en place des modèles d'homogénéisation performants permettant de remplacer par une géométrie plus simple cette partie de la géométrie réelle en prenant en compte certaines de ces caractéristiques. Ainsi, la taille du maillage généré est considérablement réduite. Malheureuse-

ment, là encore, ces modèles ne sont exploitables que si la géométrie de la structure périodique est relativement simple.

L'intérêt majeur des méthodes de type intégrale est de nécessiter uniquement le maillage des parties actives. Ainsi, la suppression du maillage de l'air et des autres parties isolantes présente un avantage évident en termes de taille et de qualité du maillage.

Au cours du travail de thèse, nous avons implémenté un logiciel nativement parallèle dévolu à la modélisation électromagnétique 3D locale d'inducteurs multibrins, aussi appelé fil de Litz. Le logiciel se nomme ModeLitz [73].

Dans ce chapitre, nous commencerons par décrire la méthode de calcul des chemins des brins. Ensuite, nous présenterons la méthode intégrale que nous avons employée pour la modélisation volumique 3D d'inducteurs multibrins. Puis, nous présenterons notre travail d'implémentation de la méthode dans le logiciel ModeLitz. Enfin, nous nous attarderons sur la validation du logiciel et l'étude de ses performances.

2.1 Description de la géométrie d'un inducteur multibrins

Comme évoqué dans la partie 1.1, la description de l'arrangement interne d'un inducteur multibrins constitue en soit un problème de modélisation.

2.1.1 Procédé de construction

Un inducteur en fil de Litz peut compter plusieurs milliers de brins agencés par paquets. Ces brins peuvent être isolés individuellement ou non isolés. Pour commencer, un certain nombre de brins sont torsadés ensemble pour former un premier paquet. Puis, plusieurs de ces paquets sont eux-mêmes torsadés pour former un paquet plus gros. En répétant ce processus avec chaque nouveau paquet, on obtient le câble complet. Le câble est donc constitué de plusieurs niveaux de paquets. On notera N ce nombre. Chaque niveau est construit en respectant deux paramètres caractéristiques, le pas et le sens de torsadage des sous-paquets qu'il contient, choisis et contrôlés par le fabricant. La dernière étape du procédé consiste à enrubanner le câble dans une ou plusieurs couches d'isolant. La figure 2.1 montre un exemple concret d'inducteur multibrins comptant environ 20 000 brins.

Dans ce manuscrit, nous nommons un câble en décrivant son agencement interne. Ainsi, un câble 4×7×7 désigne un câble constitué de 4 paquets constitués de 7 sous-paquets, chacun de ces sous-paquets comptant 7 brins, soit 196 brins au total. Pour se repérer, nous indexons les niveaux des paquets en partant de l'indice 0. Cet indice est associé au premier paquet construit, soit le paquet de 7 brins. L'indice

FIGURE 2.1 – Exemple d'inducteur multibrins comptant environ 20 000 brins en cuivre.

le plus élevé correspond au dernier niveau de paquet formé, c'est-à-dire le câble complet.

La figure 2.2 donne un schéma en coupe de l'agencement du câble 4×7×7. Ce câble compte donc trois niveaux :

— le niveau 0, les paquets comptant 7 brins ;
— le niveau 1, comptant 7 paquets de 7 brins ;
— le niveau 2, formé de 4 paquets de niveau 1.

Nous avons donc la répartition suivante : 28 paquets de niveau 0, 4 paquets de niveau 1 et 1 paquet de niveau 2. La figure 2.3 donne un aperçu des niveaux de paquets dans un inducteur 6×6×9×60, soit 19 440 brins.

2.1.2 Principe de la description géométrique

Le procédé de construction d'un fil de Litz s'explique de manière assez intuitive. En revanche, la description de la géométrie résultante dans un logiciel de calcul n'est pas chose aisée. En effet, il n'existe pas de formule mathématique permettant d'exprimer le chemin que suit un brin au sein d'un tel agencement. Par *chemin* d'un brin ou d'un paquet, nous entendons le chemin que suit le centre du brin ou du paquet en question.

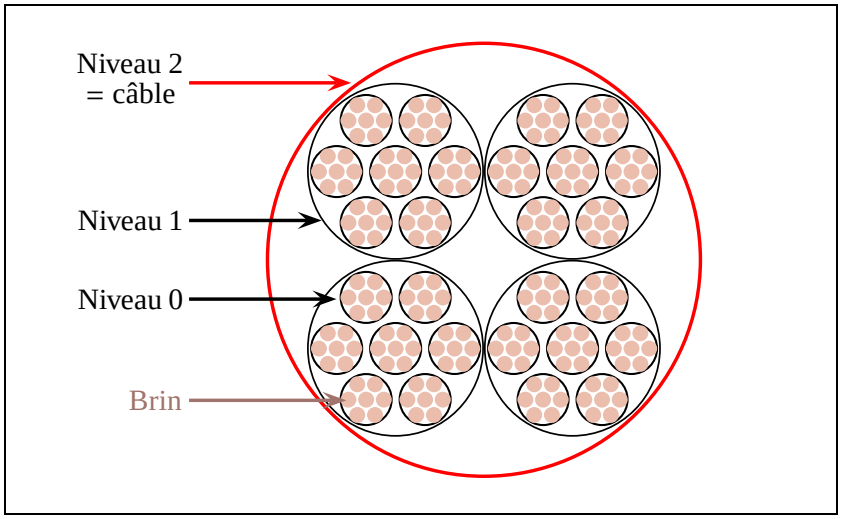


FIGURE 2.2 – Schéma en coupe de l'agencement d'un câble multibrins 4×7×7.

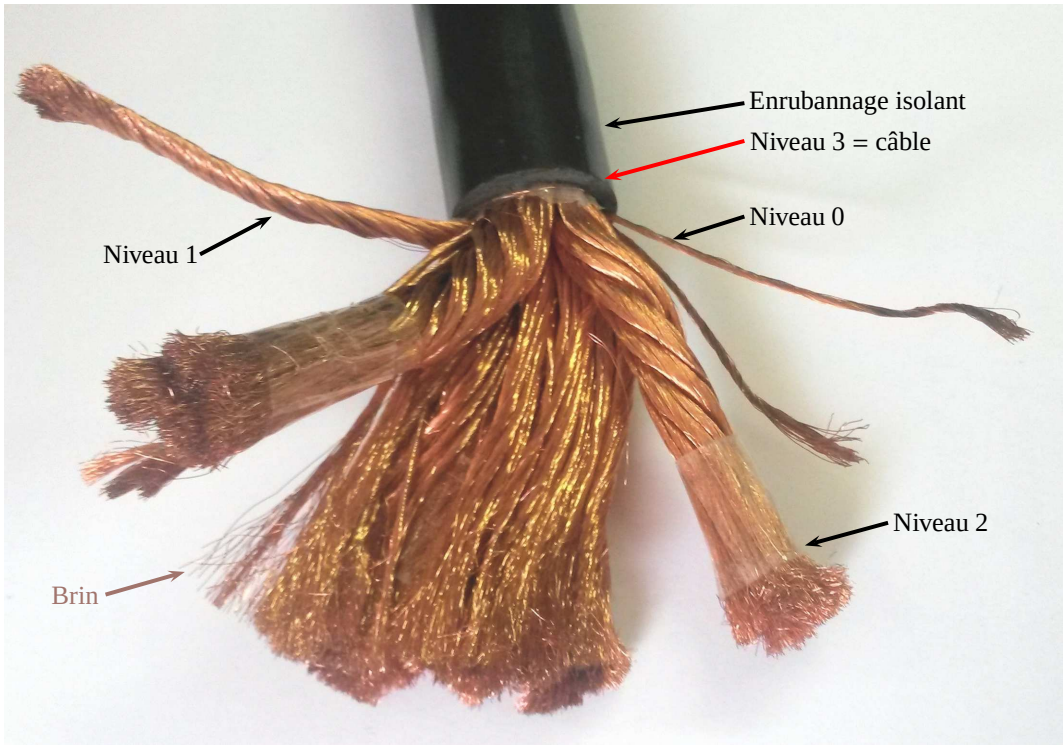


FIGURE 2.3 – Niveaux des paquets dans un inducteur 6×6×9×60.

Nous avons implémenté un algorithme de calcul des chemins des brins présents dans un câble multibrins. Précisons de suite que ce calcul ne prend pas en compte les contraintes mécaniques générées par les torsadages successifs au cours de l'élaboration du câble ainsi que par l'enrubannage final. Notre méthode s'inspire directement du procédé de construction décrit précédemment. Nous en présentons

d'abord le principe avant de la décrire plus en détail.

En sortie de la machine de torsadage, les brins sont tout d'abord agencés en un premier paquet de niveau 0 [1]. Nous prenons comme hypothèse que, lorsque le chemin de ce paquet est selon un axe droit, les brins suivent des chemins en hélice, de même axe et respectant les paramètres de torsadage de ce niveau. Nous considérons ce principe également vrai lors de l'assemblage de chaque niveau de paquet. Ainsi, un paquet de niveau $n + 1$, dont le chemin est selon un axe droit, contient des sous-paquets de niveau n dont les chemins sont des hélices, de même axe et respectant les paramètres de torsadage choisis pour le niveau $n + 1$.

Il en résulte que, dans un paquet de niveau 1 suivant un axe droit, les chemins des paquets de niveau 0 suivent des hélices. Par conséquent, les chemins des brins constituant ces paquets sont alors vus comme des « hélices » enroulées autour d'une courbe, laquelle, en l'occurrence, se trouve être une hélice. Notre algorithme permet de calculer le chemin suivi par une « hélice » enroulée autour d'une courbe quelconque. Dit autrement, nous calculons le chemin que suit une hélice dont on aurait tordu l'axe selon une courbe quelconque.

Cet algorithme est, par essence, utilisable pour le calcul des chemins de n'importe quel niveau, y compris du niveau du câble, le chemin de ce dernier étant supposé être une courbe connue par ailleurs, soit analytiquement, soit par approximation polynômiale.

Nous l'avons vu, les chemins des paquets de niveau n héritent des caractéristiques de torsadage du chemin du paquet de niveau $n + 1$ auquel ils appartiennent. Nous utiliserons les mots *père* et *fils* pour désigner respectivement les paquets et les sous-paquets. Ainsi, un paquet de niveau $n + 1$ est père des sous-paquets qu'il contient, ses fils, qui sont de niveau n. Pour calculer le chemin d'un paquet fils, il faut donc logiquement connaître préalablement le chemin que suit son paquet père.

La figure 2.4 met en évidence la relation existant entre les différents niveaux de paquets dans le cas du câble 4×7×7 décris précédemment. Les données de base nécessaires à l'algorithme de construction d'un inducteur multibrins peuvent être regroupées sous forme de tableau. Le tableau 2.1 donne un exemple pour le câble 4×7×7. Le sens de torsadage représente le sens de rotation de l'hélice formée par un chemin et prend deux valeurs : soit +1 pour une hélice tournant dans le sens trigonométrique, soit -1 pour le sens antitrigonométrique.

Indice du niveau	2	1	0
Nombre de paquets du niveau	4	7	7
Sens de torsadage	1	-1	1
Pas de torsadage (m)	0.55	0.035	0.025

TABLE 2.1 – Exemple de données de base pour la construction d'un inducteur 4×7×7.

1. Individuellement, un brin n'est pas considéré comme étant un paquet, bien que l'algorithme de calcul que nous allons décrire soit le même pour calculer les chemins des brins et des paquets de tous les niveaux.

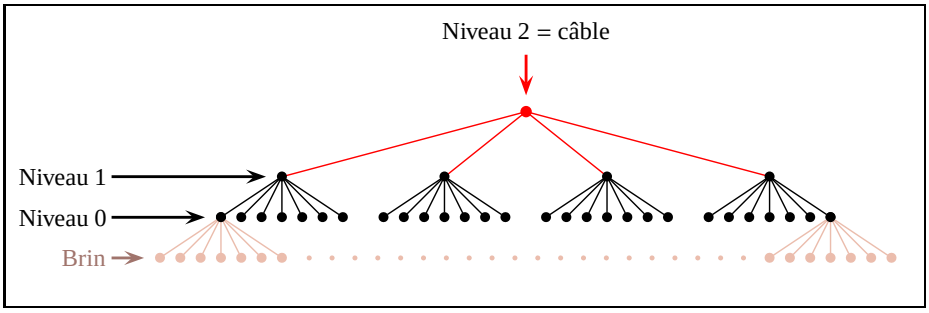


FIGURE 2.4 – Principe d'agencement des paquets et des brins pour le câble 4×7×7. Chaque paquet de niveau 0 contient sept brins.

Dans le texte, lorsque nous donnerons les paramètres de construction d'un câble comptant plusieurs niveaux de paquets, nous donnerons des jeux de valeurs selon la règle suivante : la première valeur concerne le niveau le plus élevé, la dernière concerne le niveau 0. Ainsi, le câble présenté dans la table 2.1 peut être décrit en écrivant que « le câble 4×7×7 compte trois niveaux avec les sens de torsadage (+1, −1, +1) et les pas (0.55 m, 0.035 m, 0.025 m) ». En abrégeant, nous pouvons également parler du câble 4×7×7 (0.55 m, 0.035 m, 0.025 m) (+1, −1, +1) [2].

2.1.3 Calcul de l'agencement interne des paquets

Pour pouvoir calculer les chemins de tous les paquets, il faut au préalable connaître le diamètre de chacun d'eux. Ce diamètre dépend des paramètres de torsadage.

Nous commençons par le niveau 0. A partir des paramètres de torsadage de ce niveau et connaissant le rayon des brins, nous cherchons à agencer les brins de façon à obtenir un paquet avec une compacité maximale. Nous avons conçu un algorithme « d'optimisation » simple nous permettant de déterminer l'agencement des brins correspondant le mieux aux spécifications de ce niveau (nombre de brins objectif, pas et sens de torsadage), dans le cas où le paquet suit un chemin droit. Cet agencement est alors formé de brins en hélice assemblés en couches concentriques complètement remplies. Cet agencement est calculé en respectant un critère de compacité – le paquet doit être le plus compact possible – et en cherchant à obtenir un nombre de brins le plus proche de celui demandé. Pour ce faire, nous testons divers arrangements en changeant le nombre de brins dans la couche 0 : de un seul brin jusqu'à cinq brins.

Pour calculer chacun de ces arrangements, nous nous plaçons dans un plan de coupe orthogonal au paquet. Dans ce plan, l'image de la coupe d'un brin est approximée par une ellipse dont la longueur du demi-grand axe dépend du rayon du

2. Pour simplifier encore, nous écrirons seulement 4×7×7 (0.55 m, 0.035 m, 0.025 m) quand nous insisterons sur les pas de torsadage utilisés et seulement 4×7×7 (+1, −1, +1) quand nous nous intéresserons aux sens de torsadage.

brin, du pas de torsadage du paquet ainsi que du rayon de l'hélice formée par la couche à laquelle appartient le brin. Le demi-petit axe de l'ellipse est égal au rayon du brin. En fonction de la forme des ellipses obtenues, nous calculons le nombre minimum de couches à remplir de manière à être le plus proche du nombre de brins objectif. Les couches sont construites telles qu'elles puissent se toucher. Ainsi, deux brins de deux couches voisines sont, au minimum, séparés par une distance égale à deux fois l'épaisseur de l'isolant.

Au final, l'algorithme fournit un ensemble d'arrangements et choisit celui qui compte le nombre brins le plus proche du nombre objectif.

La figure 2.5 donne un aperçu de divers assemblages de brins en hélice résultant de cet algorithme. Dans ce cas, par exemple, si le nombre de brins souhaité dans un paquet de niveau 0 est huit, alors la configuration la plus proche respectant les paramètres de torsadage fixés est l'assemblage (b). Celui-ci est constitué de deux couches de brins hélicoïdaux : la couche centrale, d'indice 0, comptant deux brins et la couche 1, comptant sept brins. En fonction du nombre de brins visé, le nombre de couches remplies peut être plus important.

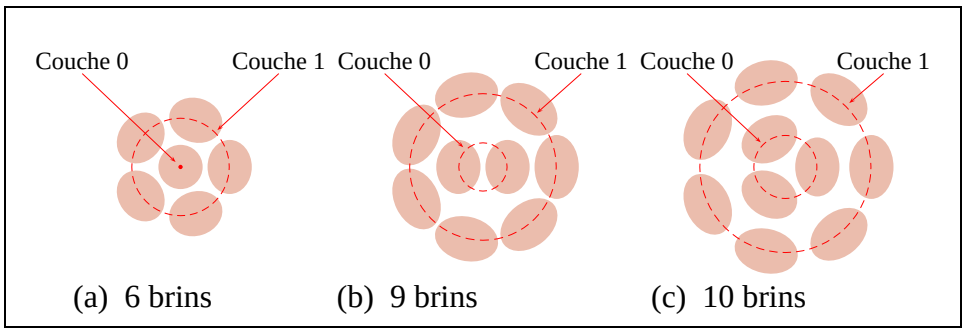


FIGURE 2.5 – Résultats de l'algorithme de calcul des possibilités d'arrangement d'un paquet de brins.

Une fois l'arrangement optimal connu, le diamètre de la section du paquet de niveau 0 peut être calculé.

L'algorithme précédent est ensuite appliqué pour le paquet de niveau 1. Dans ce cas, à partir des paramètres de torsadage de ce niveau et du diamètre d'un paquet de niveau 0, calculé précédemment, l'algorithme détermine l'agencement interne optimal. Celui-ci correspond à l'arrangement qui compte le nombre de sous-paquets de niveau 0 le plus proche du nombre objectif fixé pour le paquet de niveau 1. Nous pouvons alors calculer le diamètre de la section du paquet de niveau 1. Cette procédure est réitérée pour chaque niveau de paquet.

Finalement, nous obtenons les arrangements de tous les niveaux de paquets. Nous connaissons donc le nombre de constituants (brins ou sous-paquets) ainsi que le nombre de couches que compte chaque niveau de paquet.

2.1.4 Algorithme de calcul des chemins

Nous présentons maintenant l'algorithme utilisé pour le calcul des chemins des paquets et des brins. Nous parlerons donc de paquets et de sous-paquets. Précisons que, pour cet algorithme, dans le cas du calcul des chemins des brins, ces derniers peuvent être vus comme des sous-paquets du niveau 0. Rappelons que le principe de cet algorithme est de calculer le chemin que suit une « hélice » dont « l'axe » suit une courbe quelconque.

Pour ce faire, nous introduisons le repère de Frenet.

Soit une courbe C paramétrée par l'abscisse curviligne s et décrite par une fonction f de classe C^2, telle que $f(s) = (x(s), y(s), z(s))$. On appelle $\mathcal{F}(s)$, le repère de Frenet à l'abscisse s. Ce repère est un repère orthonormé local formé des vecteurs $\vec{N}(s)$, $\vec{B}(s)$ et $\vec{T}(s)$ appelés respectivement vecteur normal, vecteur binormal et vecteur tangent à la courbe C. La direction du vecteur $\vec{N}(s)$ passe par le centre de courbure local.

Les vecteurs $\vec{N}(s)$ et $\vec{T}(s)$ se calculent simplement par dérivation de la fonction f respectivement selon les formules (2.1) et (2.2).

$$\vec{N}(s) = \frac{\dfrac{d^2 f(s)}{ds^2}}{\left\| \dfrac{d^2 f(s)}{ds^2} \right\|} \tag{2.1}$$

$$\vec{T}(s) = \frac{df(s)}{ds} \tag{2.2}$$

Quant au vecteur $\vec{B}(s)$, il se déduit de $\vec{N}(s)$ et $\vec{T}(s)$ par (2.3).

$$\vec{B}(s) = \vec{T}(s) \wedge \vec{N}(s) \tag{2.3}$$

Nous allons maintenant décrire le principe de calcul des chemins fils. Nous suivons trois étapes, représentées de manière imagée par les figures (a), (b) et (c) de la figure 2.6.

Soit $\mathcal{R}$ le repère absolu, de coordonnées (x, y, z). Soit $C_{père}$ le chemin père d'un paquet. Soit $\mathcal{F}(s) = (A(s), \vec{N}(s), \vec{B}(s), \vec{T}(s))$, le repère de Frenet d'origine $A(s)$ lié au chemin $C_{père}$. $A(s)$ est le point d'abscisse curviligne s sur ce chemin (voir figure 2.6(a)).

Soit C_{fils} le chemin du paquet que l'on veut construire. L'objectif est de calculer la position dans le repère absolu $\mathcal{R}$ du point P appartenant au chemin C_{fils} (voir figure 2.6(c)).

Au voisinage du point $A(s)$, le chemin $C_{père}$ est approximé par une droite. Le chemin C_{fils} forme donc localement une hélice d'axe $C_{père}$ de rayon r et de pas p. Dans le repère cylindrique $\mathcal{R}_{\mathcal{H}} = (r_{\mathcal{H}}, \theta_{\mathcal{H}}, z_{\mathcal{H}})$, un point de cette hélice a pour

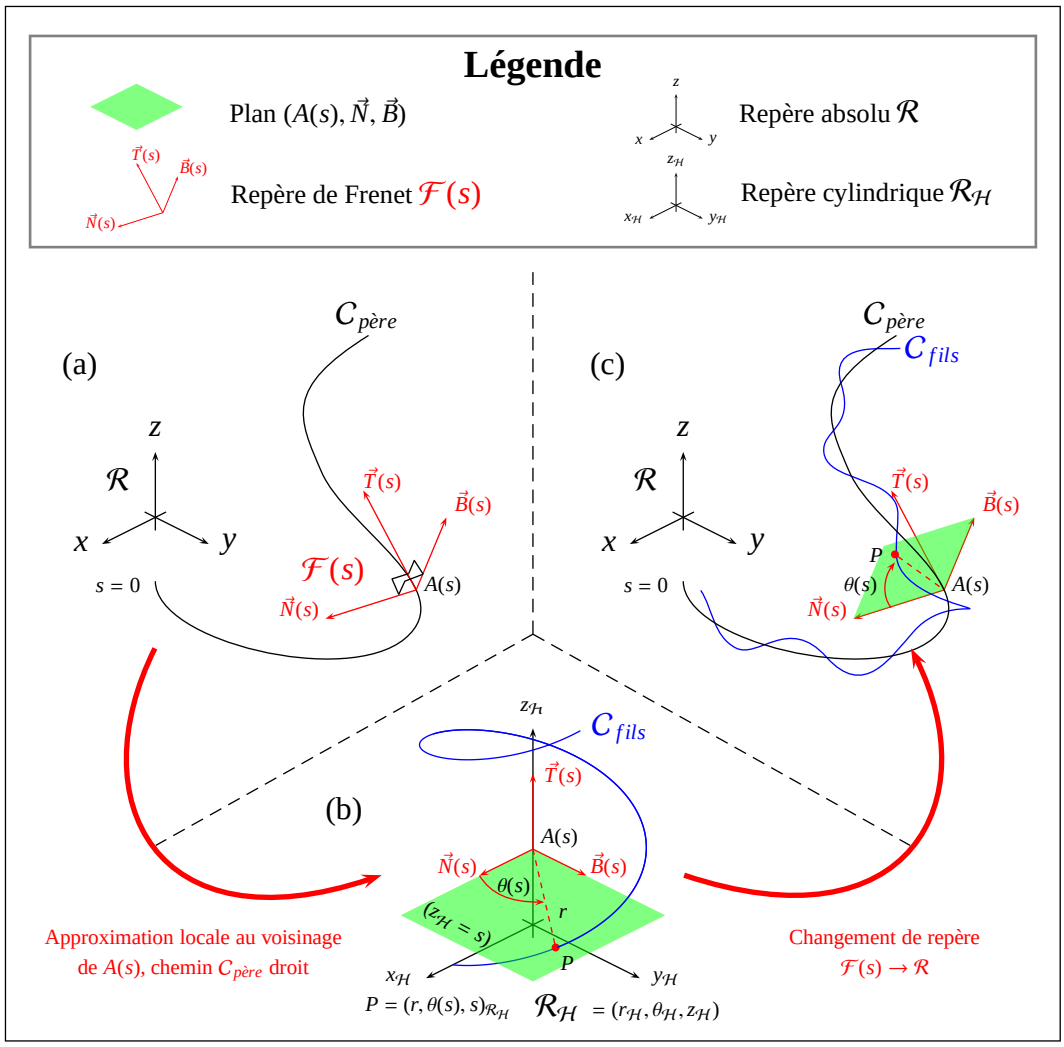


FIGURE 2.6 – Principe de construction d'un chemin fils.

coordonnées $(r, \theta, p\frac{\theta}{2\pi})$. Dans ce repère, $A(s)$ est situé en $(0, 0, s)$. Pour déterminer les coordonnées de P dans $\mathcal{R}_\mathcal{H}$, il suffit donc de calculer l'angle $\theta(s)$ correspondant à l'altitude $z_\mathcal{H} = s$, soit $\theta(s) = s\frac{2\pi}{p}$. Ainsi, le point P a pour coordonnées $(r, s\frac{2\pi}{p}, s)$ (voir figure 2.6(b)).

Dans le repère $\mathcal{F}(s)$, le point P est donc situé dans le plan $(A(s), \vec{N}(s), \vec{B}(s))$ et a pour coordonnées cartésiennes $(r\ cos(\theta(s)), y\ sin(\theta(s)), 0)$. Nous calculons alors les coordonnées du point P dans le repère $\mathcal{R}$ par le changement de repère $\mathcal{F}(s) \rightarrow \mathcal{R}$ (voir figure 2.6(c)).

En faisant varier l'abscisse s le long de $C_{père}$ et en réitérant la procédure décrite ci-dessus, nous obtenons la discrétisation du chemin C_{fils} sous forme d'un ensemble de points. En utilisant un lissage par spline cubique, nous sommes alors capables de calculer les coordonnées d'un point quelconque le long de ce chemin. Le chemin C_{fils} est alors complètement décrit (voir figure 2.6(c)).

Pour construire les chemins des paquets fils du paquet courant, dont nous venons de calculer le chemin, il suffit d'appliquer à nouveau cet algorithme en prenant cet fois le chemin C_{fils} comme paquet père. Ainsi, cet algorithme permet de calculer les chemins de tous les niveaux de paquets ainsi que les chemins des brins.

La figure 2.7 montre le chemin suivi par deux brins enroulés autour d'un chemin courbe. Ceci correspond à un câble 1×2 dont le chemin suit une hélice d'axe *z*. La figure 2.8 montre deux vues d'un câble 2×2 suivant un chemin en hélice également d'axe *z*.

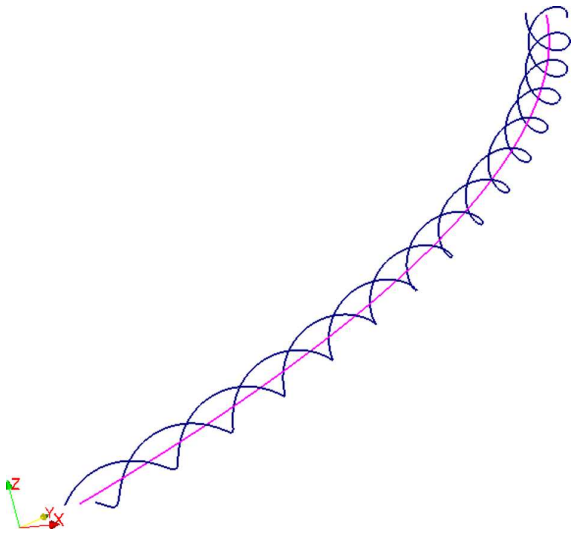
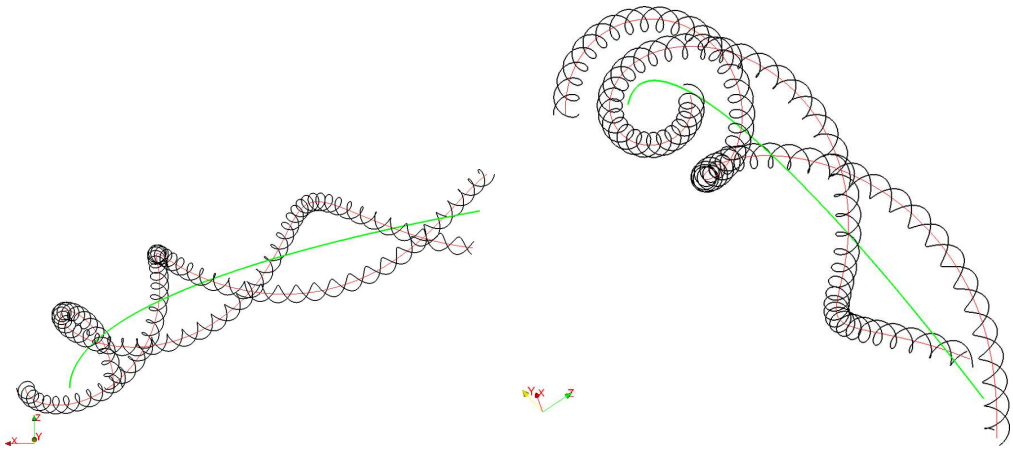


FIGURE 2.7 – Chemins de deux fils (courbes bleues) enroulés autour d'un chemin en hélice (courbe rose).



(a) (b)

FIGURE 2.8 – Chemins des constituants d'un câble 2×2. Courbe verte = chemin du câble, courbe rouge = chemin d'un paquet de deux brins, courbe noire = chemin d'un brin

2.2 Présentation de la méthode intégrale

La méthode intégrale que nous exposons dans cette partie permet d'effectuer une modélisation électromagnétique 3D volumique adaptée à des matériaux linéaires et non magnétiques. Elle s'inspire de la méthode présentée dans l'article [74] qui est une méthode intégrale 3D surfacique adaptée aux calculs d'induction dans le cas où l'ordre de grandeur de la taille des objets modélisés est grand devant l'épaisseur de peau.

2.2.1 Equations de la méthode

La présente méthode est basée sur les équations (2.4) et (2.5), traduisant respectivement la loi d'Ohm locale et la conservation du courant électrique.

$$\vec{J} = -\sigma\vec{\nabla}V - \sigma\frac{\partial\vec{A}}{\partial t} \tag{2.4}$$

$$\vec{\nabla}\cdot\vec{J} = 0 \tag{2.5}$$

avec :
- $\vec{J}$, la densité de courant électrique ;
- V, le potentiel scalaire électrique ;
- $\vec{A}$, le potentiel vecteur magnétique ;
- σ, la conductivité électrique du matériau.

Nous allons utiliser cette méthode pour la modélisation d'inducteurs multibrins en cuivre, soumis à une différence de potentiel sinusoïdale de fréquence inférieure à 3 MHz. Dans ces conditions, nous pouvons utiliser l'approximation quasi statique et l'approximation harmonique [75]. Ainsi, en introduisant la formulation complexe, l'équation (2.6) est équivalente à (2.4) et l'équation (2.7) est équivalente à (2.5).

$$\vec{\boldsymbol{J}} = -\sigma\vec{\nabla}\boldsymbol{V} - \boldsymbol{i}\omega\sigma\vec{\boldsymbol{A}} \tag{2.6}$$

$$\vec{\nabla}\cdot\vec{\boldsymbol{J}} = \boldsymbol{0} \tag{2.7}$$

où :
- $\boldsymbol{i}$, est le nombre complexe ($\boldsymbol{i}^2 = -1$) ;
- $\omega = 2\pi f$, est la pulsation associée à la fréquence f ;
- les grandeurs complexes sont notées en gras.

Le potentiel vecteur $\vec{\boldsymbol{A}}$, créé en un point quelconque, est calculé grâce à la loi de Biot et Savart (2.8).

$$\vec{\boldsymbol{A}} = \frac{\mu_0}{4\pi}\iiint_{\Omega}\frac{\vec{\boldsymbol{J}}}{r}\mathrm{d}\Omega \tag{2.8}$$

avec :

— Ω, le volume total de tous les objets actifs ;
— $\mu_0 = 4\pi \cdot 10^{-7}\text{H.m}^{-1}$, la perméabilité magnétique ;
— $\vec{\boldsymbol{J}}$, la densité de courant en un point source quelconque situé dans le volume Ω ;
— r, la distance entre le point de calcul et le point source.

La méthode est basée sur les équations (2.6), (2.8) et (2.7) et a pour inconnues les grandeurs complexes $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$.

2.2.2 Conditions aux limites

Selon qu'un objet est soumis à une alimentation électrique (un inducteur, par exemple) ou pas (un induit), deux jeux de conditions aux limites s'appliquent.

Soit un objet de type inducteur soumis à une différence de potentiel $\Delta V = V_B - V_A$. Les conditions aux limites appliquées à cet objet sont résumées sur la figure 2.9.

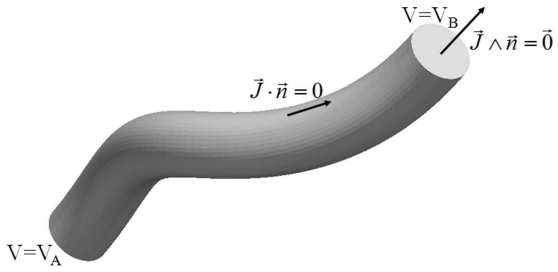


FIGURE 2.9 – Conditions aux limites appliquées à un conducteur soumis à une différence de potentiel $\Delta V = V_B - V_A$.

Dans ce cas, deux conditions de Dirichlet sur $\boldsymbol{V}$ (2.9) sont appliquées.

$$\begin{cases} \boldsymbol{V} = V_A & \text{à la première extrémité du conducteur} \\ \boldsymbol{V} = V_B & \text{à la seconde extrémité} \end{cases} \tag{2.9}$$

En ce qui concerne $\vec{\boldsymbol{J}}$, la direction de la densité de courant est également contrainte afin d'être perpendiculaire à la section du conducteur en entrée et en sortie de celui-ci par (2.10), avec $\vec{n}$ la normale à la surface.

$$\vec{\boldsymbol{J}} \wedge \vec{n} = \vec{0} \tag{2.10}$$

Sur les autres faces du conducteur, la condition d'isolation est assurée par (2.11).

$$\vec{\boldsymbol{J}} \cdot \vec{n} = \vec{0} \tag{2.11}$$

Dans le cas d'un objet de type induit, seulement deux conditions aux limites sont nécessaires. Premièrement, la condition d'isolation (2.11) est appliquée à toutes les

faces de l'objet. Deuxièmement, et pour assurer l'unicité de la solution, le potentiel électrique doit être fixé en un noeud de l'objet *via* une condition de Dirichlet. Le choix du noeud ainsi que de la valeur imposée sont arbitraires. En général, la condition (2.12) est adoptée.

$$V = 0 \tag{2.12}$$

Pour résumer, la méthode intégrale que nous avons décrite précédemment est basée sur les équations (2.6), (2.8) et (2.7) ainsi que des conditions aux limites suivantes :

— (2.9), (2.10) et (2.11) pour les objets de type inducteur ;
— (2.11) et (2.12) pour les objets de type charge.

Pour finir, ajoutons que la condition de radiation de Sommerfeld, qui impose un champ nul à l'infini, est intrinsèquement portée par l'équation (2.8).

2.2.3 Structure du maillage

Avant d'aborder la discrétisation des équations de base de la méthode, une parenthèse sur la structure du maillage est nécessaire. Cette présentation permettra de faciliter la compréhension de la partie 2.2.4. Nous exposerons un exemple de maillage sur lequel s'appuie la méthode intégrale.

Tout comme dans la publication [74], deux maillages volumiques, constitués d'éléments de Lagrange d'ordre 1, sont créés : un pour l'inconnue $\boldsymbol{V}$ et un pour l'inconnue $\vec{\boldsymbol{J}}$. Ces maillages sont construits en quinconce l'un par rapport à l'autre. Le maillage $\boldsymbol{V}$ est créé en premier. Puis, c'est au tour du maillage $\vec{\boldsymbol{J}}$. Chaque nœud $\vec{\boldsymbol{J}}$ est situé au centre d'un élément $\boldsymbol{V}$. L'équation (2.6) est calculée sur chaque nœud du maillage $\vec{\boldsymbol{J}}$. Quant à l'équation (2.7), nous la calculons autour de chaque nœud du maillage $\boldsymbol{V}$. La construction de ces maillages décalés permet d'assurer la continuité du terme $\overrightarrow{\nabla}_i \boldsymbol{V}$, en un nœud i du maillage $\vec{\boldsymbol{J}}$, puisque le gradient peut être calculé par interpolation de l'inconnue $\boldsymbol{V}$ sur l'élément $\boldsymbol{V}$ contenant ce nœud. La figure 2.10 montre, le principe de construction des maillages $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$ en 2D.

Le principe de construction du maillage en 3D est le même. Le maillage $\boldsymbol{V}$ est d'abord créé par extrusion d'un maillage 2D. Par exemple, le maillage d'un brin d'un câble multibrins est extrudé à partir du maillage en section, c'est-à-dire le maillage d'un disque. Puis, le maillage $\vec{\boldsymbol{J}}$ est construit en plaçant chaque nœuds au centre d'un élément $\boldsymbol{V}$. Cette démarche est valable tant pour les nœuds $\vec{\boldsymbol{J}}$ internes que pour ceux qui sont placés sur la surface de l'objet. Ces derniers sont placés aux centres des faces des éléments $\boldsymbol{V}$ qui sont tangentes à la surface de l'objet.

La figure 2.11 donne un aperçu de l'imbrication des deux maillages. Les nœuds $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$ sont respectivement situés aux sommets des éléments des maillages $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$.

Trois types de nœuds $\vec{\boldsymbol{J}}$ sont définis : $\boldsymbol{J}_3$, $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$. Les nœuds $\boldsymbol{J}_3$ sont les nœuds internes aux volumes des objets et portent les trois degrés de libertés correspondant à l'inconnue $\vec{\boldsymbol{J}}$, soit, respectivement, $\boldsymbol{J}_x$, $\boldsymbol{J}_y$ et $\boldsymbol{J}_z$. Les nœuds $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$ permettent de tenir compte des conditions aux limites.

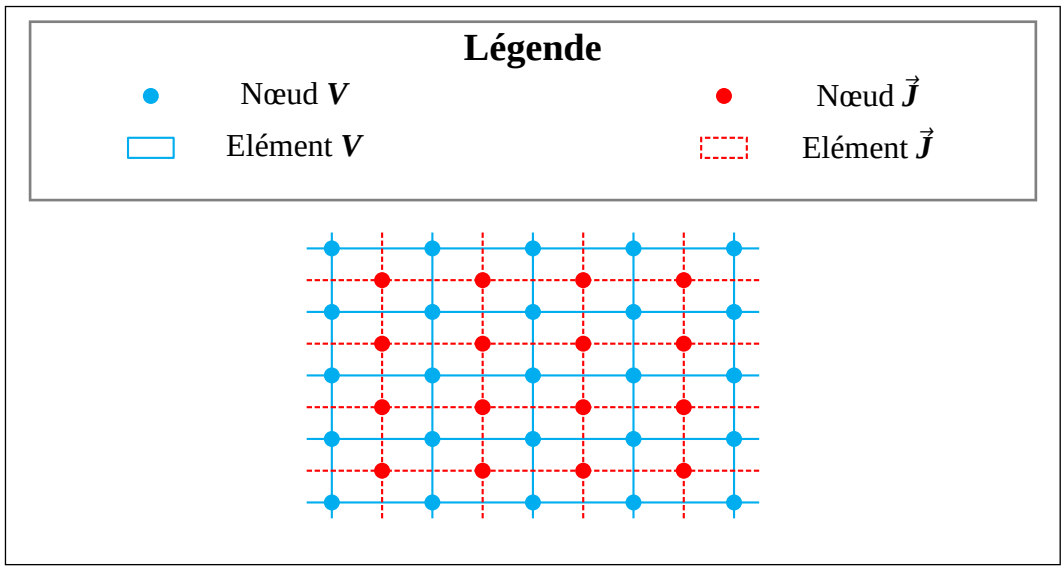


FIGURE 2.10 – Principe des maillages décalés en 2D.

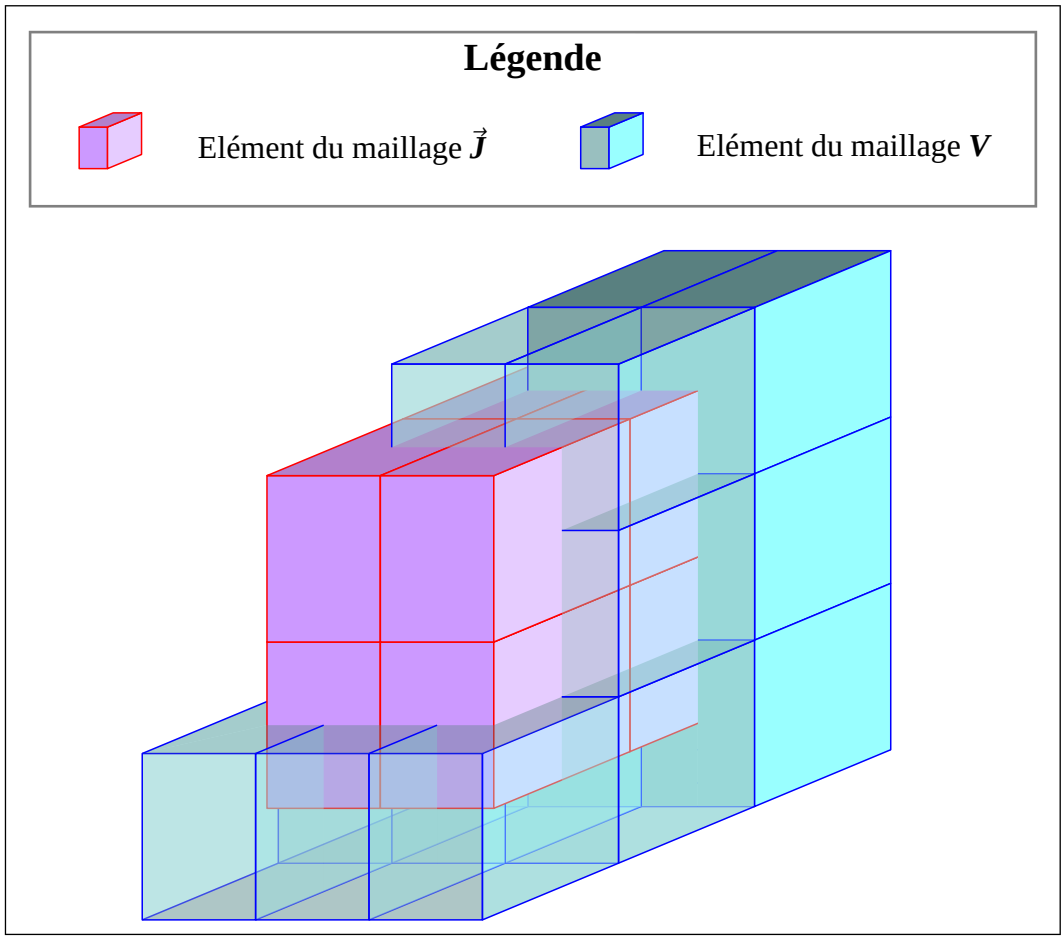


FIGURE 2.11 – Ecorché d'un maillage 3D.

Les nœuds J_2 supportent la condition d'isolation (2.11). L'inconnue $\vec{J}$ est définie comme tangente à la surface d'un objet par projection de ces composantes sur les

vecteurs $\vec{t_1}$ et $\vec{t_2}$, tangents à la surface. Les nœuds $\boldsymbol{J}_2$ portent donc deux degrés de liberté : $\boldsymbol{J}_{t1}$ et $\boldsymbol{J}_{t2}$.

Quant aux nœuds $\boldsymbol{J}_1$, ils permettent d'imposer la condition de normalité (2.10) aux extrémités d'un objet de type inducteur. L'inconnue $\vec{\boldsymbol{J}}$ est donc définie comme colinéaire à la normale $\vec{n}$ à ces faces. Les nœuds $\boldsymbol{J}_1$ portent donc un seul degré de liberté et sont situés sur les faces sur lesquelles une condition de Dirichlet de type (2.9) est imposée. Les différents types de nœuds $\vec{\boldsymbol{J}}$ existants et leurs positions dans le maillage sont résumés par la figure 2.12.

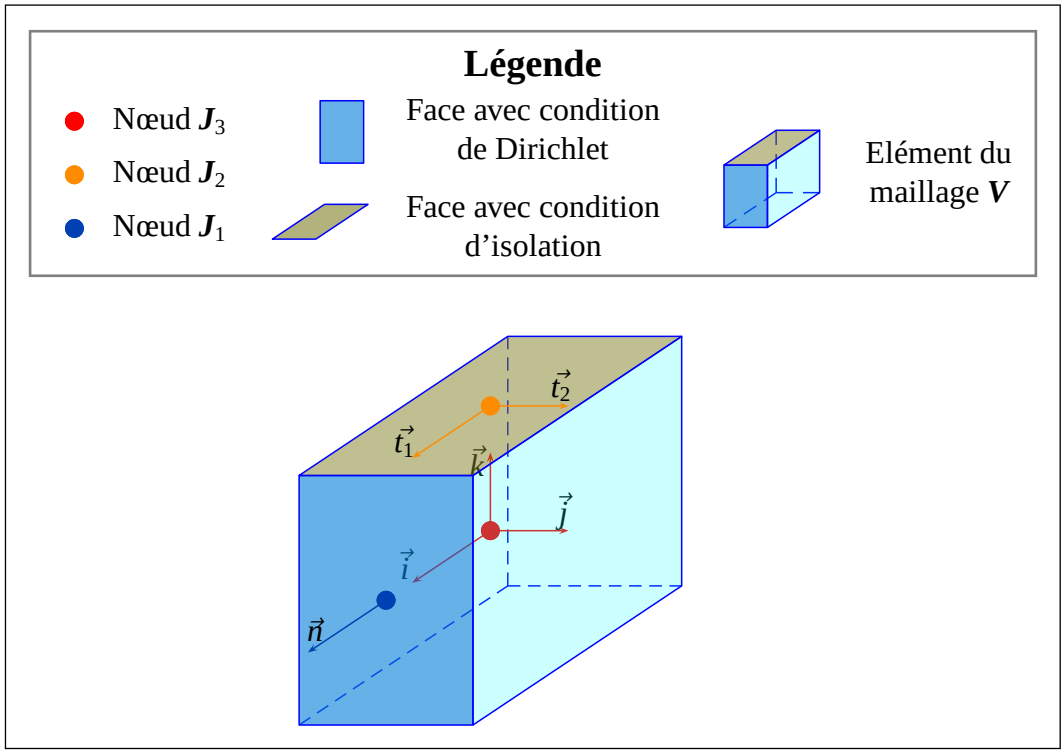
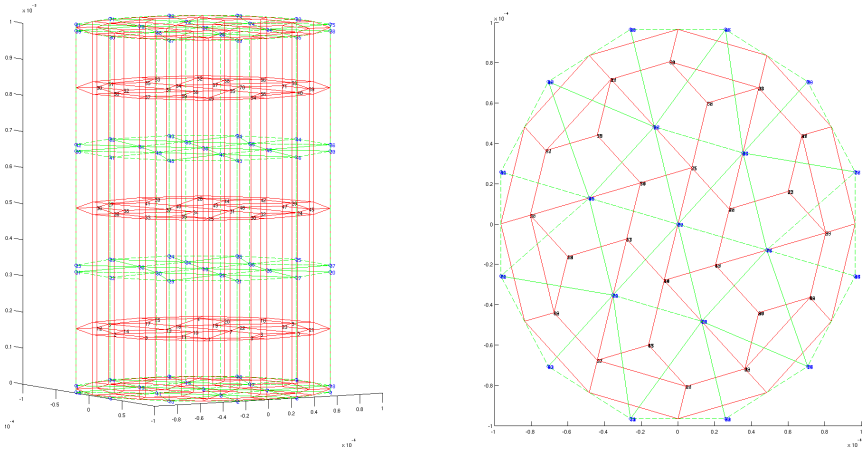


FIGURE 2.12 – Les différents types de nœuds $\vec{\boldsymbol{J}}$ et leurs positions dans le maillage en frontière d'un objet de type inducteur.

La figure 2.13 propose plusieurs vues du maillage d'un cylindre. Le maillage des brins d'un inducteur multibrins est réalisé de manière similaire. En premier lieu, le maillage $\boldsymbol{V}$ est construit par extrusion du maillage de la section du brin. L'extrusion est propagée le long d'un chemin calculé par la méthode décrite dans la partie 2.1. Ensuite, le maillage $\vec{\boldsymbol{J}}$ est construit comme nous l'avons décrit précédemment.

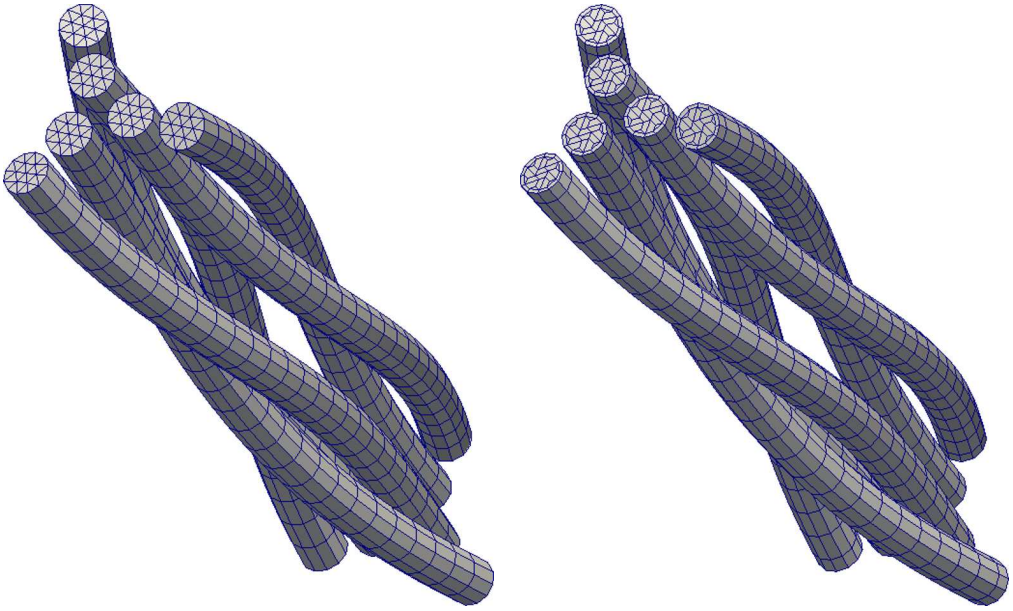
L'équation (2.6) n'a de sens que dans les objets conducteurs. C'est pourquoi la présente méthode ne nécessite pas le maillage des objets non-conducteurs, en particulier de l'isolant et de l'air entre les brins. La figure 2.14 présente les maillages $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$ d'un inducteur à six brins.



(a) Vue 3D.

(b) Vue du maillage d'une base du cylindre.

FIGURE 2.13 – Différentes vues du maillage d'un cylindre. En vert, le maillage $\boldsymbol{V}$, en rouge le maillage $\vec{\boldsymbol{J}}$.



(a) (b)

FIGURE 2.14 – Maillages $\boldsymbol{V}$ (a) et $\vec{\boldsymbol{J}}$ (b) d'un inducteur comportant trois paquets de deux brins. Paraview®[76].

2.2.4 Discrétisation des équations

Nous présentons ici la discrétisation des équations (2.6), (2.7) et (2.8) inspirée de l'article [74]. Pour exprimer chacun des termes de ces équations, nous utilisons les outils de la méthode des Éléments Finis : polynômes d'interpolation de Lagrange

et méthode de Gauss pour l'intégration numérique.

Discrétisation de la loi d'Ohm

En chaque nœud i du maillage $\vec{J}$, la loi d'Ohm locale (2.6) s'écrit (2.13).

$$\vec{J}_i = -\sigma_i \vec{\nabla}_i \boldsymbol{V} - \boldsymbol{i}\omega\sigma\vec{A}_i \tag{2.13}$$

avec :

— σ_i, la conductivité électrique du matériau au nœud i [3] ;

— $\vec{\nabla}_i V$, le gradient du potentiel électrique au nœud i.

Le potentiel vecteur $\vec{A}_i$, calculé en un noeud i du maillage $\vec{J}$, s'obtient par la discrétisation de l'équation (2.8) sous la forme suivante (2.14) :

$$\vec{A}_i = \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \iiint_{\Omega_e} \frac{\vec{J}}{r_i} \mathrm{d}\Omega_e \tag{2.14}$$

avec :

— N_{EJ}, le nombre total d'éléments du maillage $\vec{J}$;

— e, l'index d'un élément du maillage $\vec{J}$;

— Ω_e, le volume de l'élément e ;

— r_i, la distance entre le point i et un point situé dans l'élément e.

L'intégration de (2.14) est effectuée sur l'élément de référence associé à l'élément réel. Sur un élément e, la densité $\vec{J}$ s'exprime grâce aux polynômes d'interpolations selon (2.15).

$$\vec{J} = \sum_{n=1}^{NBN_e} \alpha_n(u,v,w)\vec{J}_n \tag{2.15}$$

avec :

— NBN_e, le nombre de nœuds de l'élément e ;

— n, l'indice d'un nœud de l'élément ;

— (u, v, w), les coordonnées d'un point dans l'élément de référence ;

— α_n, le polynôme d'interpolation lié au nœud n.

L'élément de volume $\mathrm{d}\Omega_e$ s'exprime à partir de (2.16).

$$\mathrm{d}\Omega_e = \det Jac \; \mathrm{d}u \; \mathrm{d}v \; \mathrm{d}w \tag{2.16}$$

où Jac est la matrice jacobienne, qui traduit la transformation qui lie l'élément réel et l'élément de référence, s'écrivant comme (2.17).

3. Dans la suite de ce manuscrit, et sauf indication explicite, nous prendrons $\sigma_i = \sigma_{Cu} = 5.997 \cdot 10^7 \mathrm{S.m}^{-1}$, la conductivité électrique du cuivre.

$$Jac = \begin{bmatrix} \dfrac{\partial x}{\partial u} & \dfrac{\partial x}{\partial v} & \dfrac{\partial x}{\partial w} \\ \dfrac{\partial y}{\partial u} & \dfrac{\partial y}{\partial v} & \dfrac{\partial y}{\partial w} \\ \dfrac{\partial z}{\partial u} & \dfrac{\partial z}{\partial v} & \dfrac{\partial z}{\partial w} \end{bmatrix} \tag{2.17}$$

Les coordonnées réelles (x, y, z) et les coordonnées (u, v, w) sont respectivement reliées par les relations (2.18), (2.19) et (2.20).

$$x = \sum_{n=1}^{NBN_e} \alpha_n(u, v, w)\ x_n \tag{2.18}$$

$$y = \sum_{n=1}^{NBN_e} \alpha_n(u, v, w)\ y_n \tag{2.19}$$

$$z = \sum_{n=1}^{NBN_e} \alpha_n(u, v, w)\ z_n \tag{2.20}$$

Nous pouvons donc écrire (2.21) :

$$\vec{A}_i = \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \iiint_{\Omega_e} \frac{1}{r_i} \sum_{n=1}^{NBN_e} \left[\alpha_n(u, v, w)\vec{J}_n\right] \det Jac\ \mathrm{d}u\ \mathrm{d}v\ \mathrm{d}w \tag{2.21}$$

La somme sur les nœuds de l'élément e et l'intégrale sur ce même élément étant commutatives, nous pouvons réécrire (2.21) en (2.22)

$$\vec{A}_i = \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \sum_{n=1}^{NBN_e} \left[\iiint_{\Omega_e} \frac{1}{r_i} \alpha_n(u, v, w) \det Jac\ \mathrm{d}u\ \mathrm{d}v\ \mathrm{d}w \right] \vec{J}_n \tag{2.22}$$

Le calcul de l'intégrale sur l'élément de référence est effectué numériquement grâce à la méthode de Gauss sur les éléments de type quadrilatère et hexaèdre et à la méthode de Hammer sur les éléments de type triangle [77]. Pour les éléments de type prisme, les coordonnées des points d'intégrations sont calculées par extrusion des points de Hammer sur la base triangulaire et suivant les coordonnées des points de Gauss selon la troisième dimension. Ainsi, l'intégrale sur l'élément de référence correspondant à l'élément e s'écrit comme (2.23).

$$\iiint_{\Omega_e} \frac{1}{r_i} \alpha_n(u, v, w) \det Jac\ \mathrm{d}u\ \mathrm{d}v\ \mathrm{d}w = \sum_{k=1}^{NPI_e} \frac{1}{r_{ik}} \alpha_n(u_k, v_k, w_k) \det Jac_k\ w_k \tag{2.23}$$

avec :

— NPI_e, le nombre de points d'intégration de l'élément e ;
— k l'indice d'un point d'intégration ;
— $\alpha_n(u_k, v_k, w_k)$, le polynôme α_n calculé aux coordonnées de référence du point d'intégration k ;
— Jac_k, la matrice jacobienne calculée au point k ;
— w_k, le poids d'intégration associé au point k ;
— r_{ik}, la distance entre le nœud i du maillage $\vec{\boldsymbol{J}}$ et le point d'intégration k, calculée grâce à (2.24).

$$r_{ik} = \sqrt{\sum_{c=x,y,z} \left[\left(c_i - \sum_{m=1}^{NBN_e} \alpha_m(u_k, v_k, w_k)\, c_m \right)^2 \right]} \tag{2.24}$$

avec c référençant successivement les coordonnées réelles x, y et z. Ainsi, c_i correspond aux coordonnées réelles du nœud i et c_m à celles du nœud m.

Ainsi, la loi de Biot et Savart se discrétise suivant l'expression (2.25).

$$\vec{\boldsymbol{A}}_i = \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \left[\sum_{n=1}^{NBN_e} \left[\sum_{k=1}^{NPI_e} \frac{1}{r_{ik}} \alpha_n(u_k, v_k, w_k) \det Jac_k \; w_k \right] \vec{\boldsymbol{J}}_n \right] \tag{2.25}$$

L'intégration du terme $1/r_{ik}$ pourrait laisser craindre un problème de divergence, mais il n'en est rien. En effet, l'intégrale de $1/r$ sur un volume fini est convergente [4]. De plus, la distance r_{ik} n'est jamais nulle car les nœuds et les points d'intégration ne sont jamais confondus.

Nous allons aborder à présent la discrétisation du terme $\overrightarrow{\nabla}_i \boldsymbol{V}$ de la loi d'Ohm. Nous rappelons que les maillages $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$ sont décalés afin d'assurer la continuité du gradient du potentiel électrique aux nœuds $\vec{\boldsymbol{J}}$. Le nœud i du maillage $\vec{\boldsymbol{J}}$ est donc situé au centre d'un élément j du maillage $\boldsymbol{V}$. Le terme $\overrightarrow{\nabla}_i V$ est donc calculé par le gradient de l'interpolation sur cet élément $\boldsymbol{V}$ selon l'expression (2.26).

$$\overrightarrow{\nabla}_i \boldsymbol{V} = \begin{bmatrix} \sum_{m=1}^{NBN_j} \left(\frac{\partial \alpha_m}{\partial u} \frac{\partial u}{\partial x} + \frac{\partial \alpha_m}{\partial v} \frac{\partial v}{\partial x} + \frac{\partial \alpha_m}{\partial w} \frac{\partial w}{\partial x} \right) \boldsymbol{V}_m \\ \sum_{m=1}^{NBN_j} \left(\frac{\partial \alpha_m}{\partial u} \frac{\partial u}{\partial y} + \frac{\partial \alpha_m}{\partial v} \frac{\partial v}{\partial y} + \frac{\partial \alpha_m}{\partial w} \frac{\partial w}{\partial y} \right) \boldsymbol{V}_m \\ \sum_{m=1}^{NBN_j} \left(\frac{\partial \alpha_m}{\partial u} \frac{\partial u}{\partial z} + \frac{\partial \alpha_m}{\partial v} \frac{\partial v}{\partial z} + \frac{\partial \alpha_m}{\partial w} \frac{\partial w}{\partial z} \right) \boldsymbol{V}_m \end{bmatrix} \tag{2.26}$$

avec :
— NBN_j, le nombre de nœuds de l'élément j ;
— m, l'indice d'un nœud de l'élément ;
— $\boldsymbol{V}_m$, le potentiel électrique au nœud m de l'élément.

4. Il suffit, pour s'en convaincre, de calculer cette intégrale sur une sphère de rayon fini.

Posons $\overrightarrow{\nabla_{ref}}\ \alpha_m$ le gradient du polynôme d'interpolation α_m sur l'élément de référence selon (2.27).

$$\overrightarrow{\nabla_{ref}}\ \alpha_m = \begin{bmatrix} \dfrac{\partial \alpha_m}{\partial u} \\ \dfrac{\partial \alpha_m}{\partial v} \\ \dfrac{\partial \alpha_m}{\partial w} \end{bmatrix} \tag{2.27}$$

On peut alors exprimer $\vec{\nabla}_i\ \boldsymbol{V}$ sous la forme de (2.28).

$$\vec{\nabla}_i \boldsymbol{V} = \sum_{m=1}^{NBN_j} \left(Jac^{-1}\right)^{\intercal} \cdot \overrightarrow{\nabla_{ref}}\ \alpha_m\ \boldsymbol{V}_m \tag{2.28}$$

avec $\left(Jac^{-1}\right)^{\intercal}$ la transposée de l'inverse de la jacobienne, l'inverse de la jacobienne s'écrivant comme (2.29)

$$Jac^{-1} = \begin{bmatrix} \dfrac{\partial u}{\partial x} & \dfrac{\partial u}{\partial y} & \dfrac{\partial u}{\partial z} \\ \dfrac{\partial v}{\partial x} & \dfrac{\partial v}{\partial y} & \dfrac{\partial v}{\partial z} \\ \dfrac{\partial w}{\partial x} & \dfrac{\partial w}{\partial y} & \dfrac{\partial w}{\partial z} \end{bmatrix} \tag{2.29}$$

Après discrétisation, la loi d'Ohm s'exprime suivant la relation (2.30).

$$\vec{\boldsymbol{J}}_i + \sigma \sum_{m=1}^{NBN_j} \left(Jac^{-1}\right)^{\intercal} \cdot \overrightarrow{\nabla_{ref}}\ \alpha_m\ \boldsymbol{V}_m + \boldsymbol{i}\ \sigma\ \omega\ \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \left[\sum_{n=1}^{NBN_e} \left[\sum_{k=1}^{NPI_e} \frac{1}{r_{ik}} \alpha_n(u_k, v_k, w_k) \det Jac\ w_k \right] \vec{\boldsymbol{J}}_n \right] = 0 \tag{2.30}$$

Discrétisation de l'équation de conservation de la densité courant

L'équation (2.7) traduit la conservation de la densité de courant. A l'instar de la méthode présentée dans l'article [74], cette équation est calculée en utilisant la technique des volumes finis par intégration sur un volume élémentaire autour de chaque nœud du maillage $\boldsymbol{V}$. A chaque nœud $\boldsymbol{V}$ est donc associée une équation (2.7), excepté pour les nœuds portant une condition de Dirichlet. Autour d'un nœud j du maillage $\boldsymbol{V}$, le volume élémentaire choisi est le volume total Ω_j des éléments $\vec{\boldsymbol{J}}$ auxquels appartient ce nœud. Ainsi, l'intégration de (2.7) s'écrit suivant (2.31).

$$\iiint_{\Omega_j} \overrightarrow{\nabla} \cdot \vec{J} \, \mathrm{d}\Omega_j = 0 \tag{2.31}$$

Soit NEJ_j le nombre d'éléments $\vec{J}$ entourant le nœud j. L'équation (2.31) se réécrit alors selon (2.32)

$$\sum_{e=1}^{NEJ_j} \iiint_{\Omega_e} \overrightarrow{\nabla} \cdot \vec{J} \, \mathrm{d}\Omega_e = 0 \tag{2.32}$$

avec :

— e, l'indice parcourant les éléments $\vec{J}$ entourant le nœud j ;

— Ω_e, le volume de l'élément e.

Par le théorème de la divergence, ou théorème de Green-Ostrogradski, l'intégrale volumique se transforme en une intégrale surfacique et on obtient (2.33).

$$\sum_{e=1}^{NEJ_j} \iiint_{\Omega_e} \overrightarrow{\nabla} \cdot \vec{J} \mathrm{d}\Omega_e = \sum_{e=1}^{NEJ_j} \iint_{\Gamma_e} \vec{J} \cdot \overrightarrow{\mathrm{d}\Gamma_e} = 0 \tag{2.33}$$

avec Γ_e la frontière du volume Ω_e et $\overrightarrow{\mathrm{d}\Gamma_e}$ l'élément surfacique correspondant et orienté suivant la normale sortante à la surface de Ω_e.

Soient NBF_e le nombre de facettes d'un élément e, NBN_f le nombre de nœuds $\vec{J}$ de la facette f et Γ_f la surface de cette dernière. La relation (2.33) se décompose alors en (2.34) en interpolant $\vec{J}$ sur les facettes de tous les éléments entourant le nœud j.

$$\sum_{e=1}^{NEJ_j} \sum_{f=1}^{NBF_e} \iint_{f} \left[\sum_{n=1}^{NBN_f} \alpha_n(u,v) \vec{J}_n \right] \cdot \overrightarrow{\mathrm{d}\Gamma_f} = 0 \tag{2.34}$$

L'intégration est effectuée sur l'élément de référence correspondant à l'élément 3D surfacique formé par la facette f. Le calcul du terme $\overrightarrow{\mathrm{d}\Gamma_f}$ s'effectue par (2.35).

$$\overrightarrow{\mathrm{d}\Gamma_f} = \begin{bmatrix} \frac{\partial x}{\partial u} \\ \frac{\partial y}{\partial u} \\ \frac{\partial z}{\partial u} \end{bmatrix} \wedge \begin{bmatrix} \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial v} \\ \frac{\partial z}{\partial v} \end{bmatrix} \mathrm{d}u \, \mathrm{d}v \tag{2.35}$$

avec (u, v) le système de coordonnées lié à l'élément de référence.

De plus, l'intégrale sur l'élément f et la somme sur les nœuds sont commutatives. Aussi, nous pouvons réécrire (2.34) sous la forme (2.36) :

$$\sum_{e=1}^{NEJ_j}\sum_{f=1}^{NBF_e}\sum_{n=1}^{NBN_f}\left[\iint_f \alpha_n(u,v)\begin{bmatrix}\frac{\partial x}{\partial u}\\ \frac{\partial y}{\partial u}\\ \frac{\partial z}{\partial u}\end{bmatrix}\wedge\begin{bmatrix}\frac{\partial x}{\partial v}\\ \frac{\partial y}{\partial v}\\ \frac{\partial z}{\partial v}\end{bmatrix}\mathrm{d}u\,\mathrm{d}v\right]\cdot\vec{\boldsymbol{J}}_n = 0 \tag{2.36}$$

L'intégration est effectuée par la méthode de Gauss en utilisant les points de Gauss sur les éléments de type quadrilatère et les points de Hammer sur les éléments triangulaires [77]. La discrétisation de l'intégrale transforme (2.36) en (2.37).

$$\sum_{e=1}^{NEJ_j}\sum_{f=1}^{NBF_e}\sum_{n=1}^{NBN_f}\left[\sum_{k=1}^{NPI_f}\alpha_n(u_k,v_k)\begin{bmatrix}\frac{\partial x_k}{\partial u}\\ \frac{\partial y_k}{\partial u}\\ \frac{\partial z_k}{\partial u}\end{bmatrix}\wedge\begin{bmatrix}\frac{\partial x_k}{\partial v}\\ \frac{\partial y_k}{\partial v}\\ \frac{\partial z_k}{\partial v}\end{bmatrix}w_k\,\mathrm{d}u\,\mathrm{d}v\right]\cdot\vec{\boldsymbol{J}}_n = 0 \tag{2.37}$$

avec :

- NPI_f, le nombre de points d'intégration associés à l'élément surfacique f ;
- k, l'indice d'un point d'intégration ;
- $\alpha_n(u_k, v_k)$, le polynôme α_n calculé aux coordonnées de référence du point d'intégration k ;
- w_k, le poids d'intégration associé au point k.

2.2.5 Structure du système linéaire

Les équations discrétisées (2.30) et (2.37) permettent de construire un système linéaire. Nous construisons ce système nœud par nœud. Le calcul des lignes correspondant à la loi d'Ohm sont obtenues par projection de l'équation discrétisée (2.30). Comme nous l'avons vu dans la partie 2.2.3, nous avons trois types de nœuds $\boldsymbol{J}$. Ainsi, trois cas se présentent :

- quand l'équation est calculée en un nœud $\boldsymbol{J}_3$, celle-ci est projetée sur les vecteurs de base du repère cartésien de référence $(\vec{i}, \vec{j}, \vec{k})$, ce qui permet de construire trois lignes du système ;
- quand l'équation est calculée en un nœud $\boldsymbol{J}_2$, celle-ci est projetée sur les vecteurs orthonormés $\vec{t_1}$ et $\vec{t_2}$ tangents à la surface en ce nœud, ce qui permet de construire deux lignes du système ;
- quand l'équation est calculée en un nœud $\boldsymbol{J}_1$, celle-ci est projetée sur le vecteur $\vec{n}$ normal à la surface en ce nœud et correspond à une ligne du système.

Cette construction nœud par nœud a l'avantage de réduire le nombre de calculs car les coefficients intégraux sont calculés une seule fois pour toutes les composantes du nœud $\boldsymbol{J}$ correspondant à chaque équation.

La figure 2.15 montre la topologie de la matrice du système linéaire obtenue par la discrétisation des équations de base de la méthode. Nous appellerons cette matrice $\boldsymbol{M}$. Nous remarquons que la matrice possède trois zones. La majeure partie de la matrice est pleine et a pour origine la partie imaginaire de la loi d'Ohm correspondant au potentiel vecteur $\vec{\boldsymbol{A}}$ (loi de Biot et Savart). Deux autres zones sont creuses car elles correspondent à des calculs menés sur un nombre restreint d'éléments ou de nœuds. En effet, pour chaque nœud $\vec{\boldsymbol{J}}$, la partie de la loi d'Ohm correspondant à la conduction électrique, $\sigma\vec{\nabla}\boldsymbol{V}$, est calculée sur l'élément $\boldsymbol{V}$ portant ce nœud $\vec{\boldsymbol{J}}$. D'autre part, pour chaque nœud $\boldsymbol{V}$, l'équation de la conservation de la densité de courant électrique, $\vec{\nabla}\cdot\vec{\boldsymbol{J}} = 0$, est calculée sur les éléments $\vec{\boldsymbol{J}}$ entourant ce nœud.

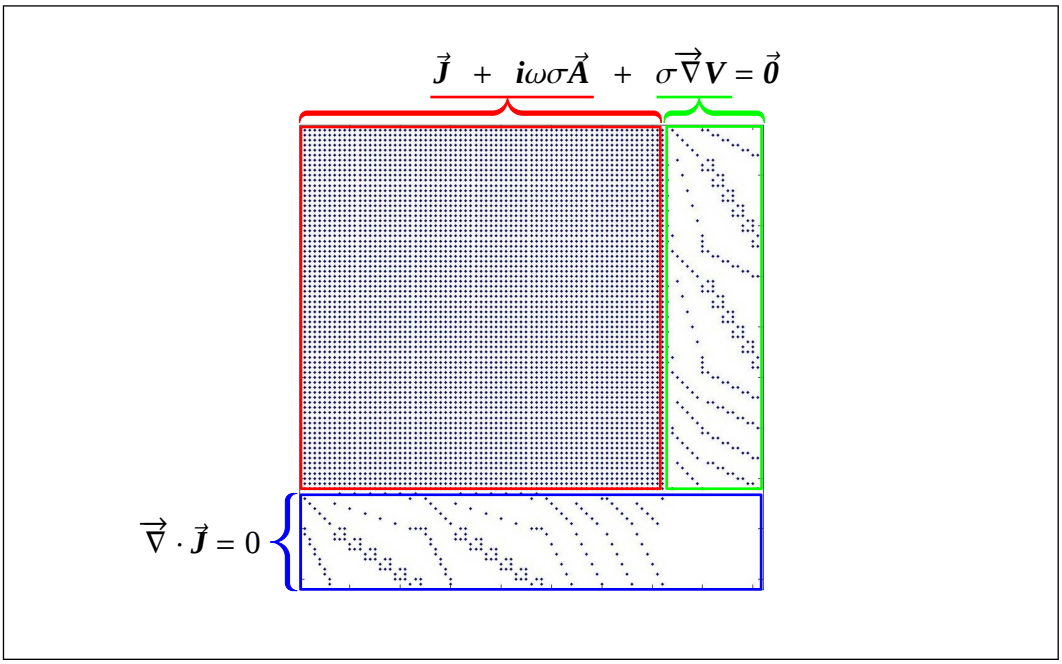


FIGURE 2.15 – Topologie du système linéaire.

Il est important de noter que la matrice n'est pas symétrique. Ceci est notamment dû au terme (2.23) intervenant dans le calcul de la loi d'Ohm.

2.2.6 Points forts et limites de la méthode

Dans la partie 1.2, nous avons évoqué les principales limites rencontrées par les méthodes existantes pour la simulation des inducteurs multibrins. A cause de la difficulté de la représentation géométrique ainsi que des contraintes liées au maillage de ces arrangements, la plupart des méthodes d'homogénéisation recourent à des calculs analytiques ou semi-analytiques ne prenant pas en compte l'arrangement 3D interne des inducteurs. De plus, celles-ci s'appuient souvent sur l'hypothèse d'un fil de Litz idéal, c'est-à-dire dans lequel la densité de courant est uniformément répartie entre tous les brins et qui est non-vérifiée à grand nombre de brins. Enfin, certaines méthodes procédant à une homogénéisation grâce à des calculs Eléments

Finis ne sont applicables que si la géométrie de la zone dont on cherche à simplifier la modélisation peut être construite par répétition d'un motif unitaire. Dans le cas d'un inducteur multibrin, ce motif de base n'est pas trivial ; si tant est qu'il existe systématiquement, quel que soit l'arrangement des brins.

Le principal avantage de la méthode que nous avons présentée dans cette partie consiste en l'absence de maillage de l'air, et plus généralement, des parties non-conductrices. Le nombre de degrés de liberté est ainsi considérablement réduit, en comparaison à d'autres méthodes, comme la méthode des Eléments Finis. De plus, la qualité du maillage peut être nettement améliorée puisque les parties conductrices non-connexes peuvent être maillées indépendamment, sans contrainte de distance. Cela permet d'éliminer complètement la difficulté du maillage de l'espace entre les objets. En effet, dans un maillage de type Eléments Finis, quand des objets sont très proches, le maillage des interstices nécessite un grand nombre d'éléments. De plus, il peut être alors difficile de contrôler la qualité du maillage. A l'inverse, lorsque la distance séparant les objets est plus grande que la taille moyenne de l'objet, l'important rapport entre la taille totale du domaine modélisé et la taille des objets conduit également à un maillage conséquent. Dans la présente méthode, la taille du maillage reste inchangée quelle que soit la distance séparant les objets.

D'autre part, la condition de rayonnement à l'infini étant satisfaite intrinsèquement par l'équation (2.8), il n'est pas nécessaire de construire une surface fermée autour du domaine de calcul sur laquelle définir cette condition. Ceci simplifie la construction géométrique et contribue à réduire la taille du maillage.

Enfin, l'influence de toutes les parties électromagnétiquement actives est calculée de manière forte, ce qui assure une bonne prise en compte des phénomènes. Néanmoins, cela a un coût et la méthode présente trois inconvénients majeurs.

Le premier tient au fait que, comme nous l'avons évoqué précédemment, les équations relevant du calcul des inconnues $\vec{\boldsymbol{J}}$ nécessitent l'intégration de la loi de Biot et Savart sur tous les éléments du maillage $\vec{\boldsymbol{J}}$ et ce, pour chaque inconnue $\vec{\boldsymbol{J}}$. Ces calculs représentent l'essentiel du volume de calculs nécessaires à l'assemblage du système. Ainsi, nous pouvons estimer que le temps de calcul croît comme le carré du nombre d'inconnues $\vec{\boldsymbol{J}}$ et, par conséquent, varie selon une loi en puissance quelque peu inférieure au carré du nombre total d'inconnues. Le nombre d'inconnues $\vec{\boldsymbol{J}}$ est égal à $3 \times N_{J_3} + 2 \times N_{J_2} + N_{J_1}$ avec N_{J_3}, N_{J_2} et N_{J_1} respectivement les nombres de nœuds $\boldsymbol{J}_3$, $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$ du maillage.

Toujours à cause du calcul de la loi de Biot et Savart, le système est plein, et de surcroît non-symétrique, ce qui constitue un deuxième inconvénient. En effet, il est alors nécessaire de stocker la matrice sous forme pleine. Par conséquent, la taille mémoire varie comme le carré du nombre de degrés de liberté qui est égal à $3 \times N_{J_3} + 2 \times N_{J_2} + N_{J_1} + N_V$, avec N_V le nombre de nœuds $\boldsymbol{V}$ portant une inconnue $\boldsymbol{V}$[5]. Le système linéaire requiert donc un important espace de stockage en mémoire.

5. Ce nombre est égal au nombre total de nœuds $\boldsymbol{V}$ moins le nombre de nœuds $\boldsymbol{V}$ soumis à une condition de Dirichlet.

Ainsi, pour donner un ordre de grandeur, un système plein comptant 55 000 degrés de liberté occupe environ 48.4 Go [6].

Enfin, troisième inconvénient : la nature du système linéaire obtenu ne permet pas l'emploi de méthodes de résolution rapides. Divers essais nous ont conduit à choisir la méthode de Gauss-Jordan, avec recherche du pivot maximal. Cette méthode est coûteuse car elle nécessite un nombre de calculs de l'ordre du cube du nombre d'inconnues du système [78].

Parce qu'elle ne requiert que la description et le maillage des objets conducteurs, la méthode intégrale s'impose comme une alternative prometteuse pour la modélisation d'inducteurs multibrins à géométrie complexe. Néanmoins, cette méthode est limitée par l'espace mémoire nécessaire au stockage complet de la matrice. De plus, les temps d'assemblage et de résolution du système sont conséquents.

Principe du calcul multifréquenciel

Dans le cadre de l'étude des inducteurs multibrins, des études multifréquentielles sont incontournables pour suivre l'évolution du comportement des inducteurs en fonction de la fréquence. La méthode intégrale décrite précédemment offre une possibilité intéressante pour accélérer les calculs de ce type d'étude : l'étape de construction du système linéaire est nécessaire uniquement pour le calcul d'une seule fréquence. Nous appellerons cette fréquence *fréquence de référence* ou F_{ref}. Pour obtenir le système linéaire correspondant à une autre fréquence F, il suffit de multiplier la partie imaginaire de chaque terme du système linéaire obtenu pour la fréquence de référence – le terme $\boldsymbol{i}\omega\vec{\boldsymbol{A}}$ (cf. figure 2.15) – par le rapport F/F_{ref}.

Pour effectuer des études en fréquence, nous procédons donc comme suit. A la fin de la construction du système correspondant à la fréquence de référence, nous conservons en mémoire dans un fichier les termes non-nuls du système ainsi que la valeur F_{ref}. Pour effectuer un calcul à une fréquence F, le fichier est lu pour reconstruire le système correspondant en tenant compte du rapport F/F_{ref}.

Bien évidemment, cette procédure n'est possible qu'à la condition de disposer d'un espace de stockage sur disque suffisant, mais elle permet de réduire de manière appréciable le temps de construction du système linéaire. Certes, pour la fréquence F_{ref}, cette étape nécessite de nombreux calculs d'intégration pour évaluer les termes correspondants à la loi de Biot et Savart, mais, pour les autres fréquences, le principe présenté ici permet de remplacer ces calculs coûteux par de simples multiplications.

2.3 Implémentation parallèle de la méthode intégrale

Dans la partie 1.3, nous avons décrit différentes architectures de machines ainsi que le principe de fonctionnement d'un code de calcul parallèle utilisant la biblio-

6. Pour rappel, le système constitue une matrice à coefficients complexes, chaque coefficient occupant un espace de 2×8 octets = 16 octets.

thèque MPI. Précédemment, dans la partie 2.2, nous avons présenté la méthode intégrale que nous avons jugée pertinente pour la modélisation locale d'inducteurs multibrins.

Nous allons maintenant nous pencher sur l'implémentation parallèle de cette méthode dans le logiciel ModeLitz qui a été nativement implémenté en parallèle avec MPI. Nous identifierons les tâches pouvant faire l'objet d'un traitement parallèle puis nous détaillerons leur principe de fonctionnement en paralléle.

2.3.1 Analyse de la méthode

Dans la partie 2.2, nous avons présenté la méthode intégrale que nous utilisons pour le calcul du comportement électromagnétique des inducteurs multibrins. Pour rappel, cette méthode nécessite deux maillages : un maillage $\boldsymbol{V}$, pour le potentiel électrique, et un maillage $\vec{\boldsymbol{J}}$, pour la densité de courant électrique. A chacun de ces maillages nous associons une des deux équations sur laquelle la méthode est basée et que nous rappelons ici.

Soit, pour le calcul de l'inconnue $\vec{\boldsymbol{J}}_i$ portée par un nœud i du maillage $\vec{\boldsymbol{J}}$, ce nœud appartenant à un élément j du maillage $\boldsymbol{V}$, l'équation (2.38) :

$$\vec{\boldsymbol{J}}_i + \sigma \sum_{m=1}^{NBN_j} \left(Jac^{-1}\right)^{\intercal} \cdot \overrightarrow{\nabla_{ref}}\, \alpha_m\, \boldsymbol{V}_m - \boldsymbol{i}\omega\sigma \frac{\mu_0}{4\pi} \sum_{e=1}^{N_{EJ}} \sum_{n=1}^{NBN_e} \left[\iiint_{\Omega_e} \frac{\alpha_n}{r_i} \mathrm{d}\Omega_e \right] \vec{\boldsymbol{J}}_n = 0 \quad (2.38)$$

avec :

- NBN_j, le nombre de nœuds de l'élément j ;
- m, l'index d'un nœud de l'élément j ;
- $\overrightarrow{\nabla_{ref}}\, \alpha_m$, le gradient du polynôme d'interpolation α_m sur l'élément de référence lié à l'élément j ;
- Jac, la matrice jacobienne calculée sur l'élément j ;
- $\boldsymbol{V}_m$, le potentiel électrique au nœud m de l'élément j ;
- N_{EJ}, le nombre total d'éléments du maillage $\vec{\boldsymbol{J}}$;
- e, l'index des éléments du maillage $\vec{\boldsymbol{J}}$;
- Ω_e, le volume de l'élément e ;
- NBN_e, le nombre de nœuds de l'élément e ;
- n, l'index d'un nœud de l'élément e ;
- $\vec{\boldsymbol{J}}$, la densité de courant électrique en un point du volume Ω_e ;
- r_i, la distance entre le point i et le point où se trouve $\vec{\boldsymbol{J}}$.

Pour chaque nœud j du maillage $\boldsymbol{V}$ porteur d'une inconnue, nous avons l'équation (2.39) :

$$\sum_{e=1}^{NEJ_j} \iint_{\Gamma_e} \vec{\boldsymbol{J}} \cdot \overrightarrow{\mathrm{d}\Gamma_e} = 0 \quad (2.39)$$

avec :

- NEJ_j, le nombre d'éléments $\vec{\boldsymbol{J}}$ entourant le nœud j ;

— e, l'index d'un de ces éléments $\vec{\boldsymbol{J}}$;
— Γ_e, la surface orientée entourant l'élément e ;
— $\vec{\boldsymbol{J}}$, la densité de courant en un point de Γ_e.

Les intégrations sur les éléments volumiques et surfaciques 3D sont effectuées numériquement et sont décrites dans la partie 2.2.4.

Dans la partie 2.2.6, nous avons souligné que cette méthode conduit à un algorithme séquentiel qui présente trois inconvénients majeurs : un besoin d'espace mémoire important pour stocker la matrice du système et des temps d'assemblage et de résolution conséquents.

Les équations (2.38) et (2.39) sont linéaires. Sous réserve que chaque processus ait connaissance de l'ensemble du maillage $\vec{\boldsymbol{J}}$, la construction du système peut donc être parallélisée. Comme chaque ligne du système peut être calculée indépendamment, la parallélisation de cette étape peut s'effectuer de manière particulièrement efficace puisqu'elle ne requiert pas de communication. Cela constitue un cas idéal de parallélisation en ce que l'accélération du calcul est alors théoriquement proportionnelle au nombre de processus. Ainsi, le parallélisme est une solution particulièrement appropriée au premier inconvénient cité dans la partie 2.2.6.

D'autre part, grâce aux potentialités offertes par MPI, il est possible de profiter des ressources offertes par des architectures très variées, notamment les réseaux de stations de travail et les clusters, et d'avoir accès à davantage de mémoire RAM. Ceci est particulièrement intéressant pour répondre à la deuxième limitation évoquée précédemment. En effet, la matrice du système peut être divisée entre chacun des processus et donc stockée par morceaux sur chacune des machines.

Pour terminer, nous espérons accélérer également l'algorithme de résolution par la parallélisation. En effet, la recherche du pivot maximal est presque entièrement parallélisable. Chaque processus peut rechercher son propre pivot maximal local, ce qui constitue une étape absolument indépendante entre les processus. Quelques communications sont nécessaires pour connaître la valeur et la position du pivot maximum absolu dans la matrice. Les calculs procédant à la diagonalisation de la matrice peuvent également être parallélisés. Néanmoins, nous ne pouvons espérer accélérer autant la résolution du système que sa construction car l'algorithme nécessite des communications pour intervertir les lignes et les colonnes de la matrice.

Après cette brève analyse, il apparaît que le calcul parallèle permet de lever certaines limites inhérentes à la méthode intégrale, principalement en ce qui concerne les temps de calcul. L'implémentation de cette méthode dans le code ModeLitz est le fruit de la présente réflexion. Les processus sont organisés suivant une topologie maître-esclave, le maître étant le processus de rang 0.

2.3.2 Construction et assemblage du système

Nous voulons exploiter le caractère indépendant des équations (2.38) et (2.39) pour paralléliser la construction du système. De plus, nous souhaitons morceler

la matrice afin de permettre un stockage réparti sur l'ensemble des processus. Il nous faut donc procéder astucieusement pour optimiser l'*équilibrage de charge* des machines.

On parle généralement d'équilibrage de charge en sous-entendant un équilibrage en terme de volumes de calculs : on cherche, autant que faire se peut, à attribuer la même quantité de calculs à chaque processus afin de minimiser les temps d'attente induits par les communications bloquantes et les synchronisations. Nous veillerons donc à cela.

Nous prêterons également attention à l'équilibrage en terme d'espace mémoire. Nous nous efforcerons d'assurer une répartition équitable de la matrice entre les processus tout en conservant une implémentation simple. Nous distinguerons donc l'équilibrage de charge en calculs, ou équilibrage en calculs, de l'équilibrage de charge en mémoire, ou équilibrage en mémoire. En ce qui nous concerne, ce dernier conditionne en partie l'équilibrage de charge en calcul. Nous exposerons donc, dans un premier temps, l'équilibrage en mémoire, puis l'équilibrage en calculs.

Équilibrage de charge en mémoire

Soit N le nombre de degrés de liberté du système et N_P le nombre de processus MPI. Nous partageons équitablement la matrice en la découpant selon les lignes en N_P sous-matrices de dimensions $n \times N$ avec n calculé par la relation (2.40) [7].

$$n = \left\lfloor \frac{N}{N_P} \right\rfloor \tag{2.40}$$

Bien évidemment, le partage ne peut pas être toujours parfaitement équitable et on défini le nombre r par la relation (2.41) [8].

$$r = N \pmod{N_P} \tag{2.41}$$

Nous décidons de confier le stockage des r lignes restantes au processus maître qui possède donc une sous-matrice de taille $(n + r) \times N$. Quant aux esclaves, ils possèdent chacun une sous-matrice de taille $n \times N$. La figure 2.16 montre le principe de répartition de la matrice entre les processus.

La simplicité de cet algorithme de stockage induit évidemment un certain surcoût en mémoire pour le maître comparé aux esclaves. Néanmoins, cela permet de simplifier de manière non négligeable l'implémentation du code exécuté par les esclaves le rendant ainsi plus robuste. De plus, ce surcoût est en général relativement peu conséquent en regard de la taille d'une sous-matrice de dimension $n \times N$. Le cas échéant, il est néanmoins possible de jouer sur le nombre de processus pour équilibrer la charge mémoire.

7. Le résultat du calcul $\left\lfloor \frac{A}{B} \right\rfloor$ est la partie entière du quotient $\frac{A}{B}$.
8. Le résultat du calcul $A \pmod{B}$ est le reste, ou modulo, du quotient $\frac{A}{B}$.

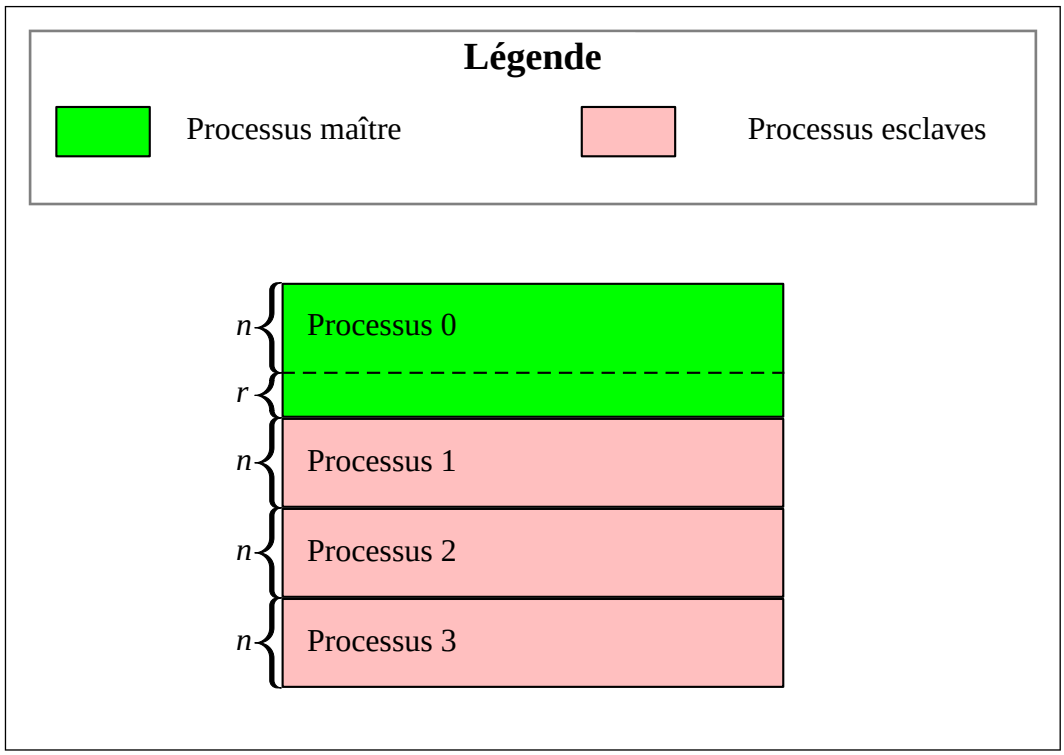


FIGURE 2.16 – Répartition de la matrice entre les processus.

Cet excès de degrés de liberté à gérer par le maître pourrait faire craindre un certain déséquilibre de charge en calcul. Comme nous le verrons plus loin, si cela est en partie vrai pour le solveur, en revanche, ce déséquilibre n'a pas d'impact significatif sur la vitesse d'assemblage du système.

Équilibrage de charge en calculs

L'équilibrage des calculs est directement lié aux différents types de nœuds existants, et donc aux équations qui leur sont associées. Nous décrivons ici la répartition des calculs des équations entre les processus. Il est évident, comme le montrera la figure 2.17, qui résumera cette partie, que les équations de type (2.39) qui sont calculées aux nœuds $\boldsymbol{V}$, constituent une part négligeable du volume de calculs nécessaires à la construction du système. C'est plutôt la partie correspondant à la loi de Biot et Savart dans les équations de type (2.38) qui est la plus gourmande en temps de calcul. Ces équations sont calculées aux nœuds $\vec{\boldsymbol{J}}$.

Comme nous l'avons montré dans la partie 2.2.5, que ce soit pour les nœuds de type $\boldsymbol{J}_2$ ou de type $\boldsymbol{J}_3$, une seule intégration par élément $\vec{\boldsymbol{J}}$ suffit pour calculer les coefficients de Biot et Savart pour chacune des composantes de l'inconnue $\vec{\boldsymbol{J}}$ liée à ces nœuds. Pour alléger l'écriture, nous parlerons par la suite de *traitement d'un nœud*, ou de *calcul sur un nœud*, au sens de calcul des équations portées par ce nœud pour l'assemblage des lignes correspondantes dans le système.

A nombre de nœuds identiques, l'assemblage du système est donc en moyenne

3 fois plus rapide pour le traitement des nœuds $\boldsymbol{J}_3$ que celui des nœuds $\boldsymbol{J}_1$, puisqu'il nécessite 3 fois moins de calculs. Pour la même raison, les calculs sur les nœuds $\boldsymbol{J}_2$ sont 2 fois plus rapides.

Il est donc plus intéressant de faire en sorte que le traitement d'un nœud $\boldsymbol{J}_3$ ou $\boldsymbol{J}_2$ soit effectué complètement par le même processeur plutôt que de répartir ce calcul sur plusieurs processeurs car, dans ce cas, le calcul des coefficients de Biot et Savart doit être effectué plusieurs fois. Nous veillons à cet aspect et répartissons les calculs entre les processeurs en tenant compte des types de nœuds.

Chaque processus construit son propre maillage complet de la géométrie puis évalue le nombre de nœuds de chaque type pour lesquels il réalisera la discrétisation des équations (2.38) et (2.39). Les autres nœuds seront traités par les autres processus.

Les nombres N_{J_3}, N_{J_2} et N_{J_1} représentent respectivement les nombres de nœuds $\boldsymbol{J}_3$, $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$ du maillage et le nombre N_V, le nombre de nœuds $\boldsymbol{V}$ portant une inconnue $\boldsymbol{V}$. Le nombre de degrés de liberté du système est défini par la relation (2.42).

$$N = N_{J_3} + N_{J_2} + N_{J_1} + N_V \tag{2.42}$$

Nous définissons les quantités entières n_{J_3}, n_{J_2}, n_{J_1} et n_V comme, respectivement, les nombres minimaux de nœuds $\boldsymbol{J}_3$, $\boldsymbol{J}_2$, $\boldsymbol{J}_1$ et $\boldsymbol{V}$ dont chaque processus reçoit le traitement. Ces quantités son calculées respectivement par les formules (2.43), (2.44), (2.45) et (2.46).

$$n_{J_3} = \left\lfloor \frac{N_{J_3}}{N_P} \right\rfloor \tag{2.43}$$

$$n_{J_2} = \left\lfloor \frac{N_{J_2}}{N_P} \right\rfloor \tag{2.44}$$

$$n_{J_1} = \left\lfloor \frac{N_{J_1}}{N_P} \right\rfloor \tag{2.45}$$

$$n_V = \left\lfloor \frac{N_V}{N_P} \right\rfloor \tag{2.46}$$

Ainsi, sur l'ensemble des N_P processus, nous répartissons de manière homogène le traitement de n_{J_3} nœuds $\boldsymbol{J}_3$, de n_{J_2} nœuds $\boldsymbol{J}_2$, de n_{J_1} nœuds $\boldsymbol{J}_1$ et de n_V nœuds $\boldsymbol{V}$.

Nous définissons également les quantités r_{J_3}, r_{J_2}, r_{J_1} comme les nombres de nœuds restants, respectivement de type $\boldsymbol{J}_3$, $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$, dont le traitement ne peut être totalement répartis sur l'ensemble des processus. Ces quantités sont calculées respectivement par les formules (2.47), (2.48), (2.49). Nous n'avons pas besoin de définir une quantité similaire pour les nœuds $\boldsymbol{V}$ car nous allons les répartir suivant une logique différente.

$$r_{J_3} = N_{J_3} \pmod{N_P} \tag{2.47}$$

$$r_{J_2} = N_{J_2} \pmod{N_P} \tag{2.48}$$

$$r_{J_1} = N_{J_1} \pmod{N_P} \tag{2.49}$$

Les quantités r_{J_3}, r_{J_2} et r_{J_1} sont des nombres entiers compris dans l'intervalle $[0; N_P - 1]$. Afin d'équilibrer la charge de calcul, nous pouvons donc répartir les nœuds $\boldsymbol{J}_3$, $\boldsymbol{J}_2$ et $\boldsymbol{J}_1$ restants, un par un, respectivement sur r_{J_3}, r_{J_2} et r_{J_1} processus.

Chaque processus possède une sous-matrice de taille $n \times N$, hormis le processus maître pour lequel la sous-matrice a une taille $(n+r) \times N$. Chaque processus esclave alloue donc l'espace mémoire nécessaire au stockage des termes de n équations correspondantes à n degrés de libertés. De même, le maître stocke les termes de $(n + r) \times N$ équations. Répartir le traitement des nœuds entre les processus revient donc à attribuer à chaque processus un certain nombre de degrés de liberté.

Chaque type de nœud porte un certain nombre de degrés de liberté : 3 pour les nœuds $\boldsymbol{J}_3$, respectivement 2 pour les $\boldsymbol{J}_1$, 1 pour les $\boldsymbol{J}_1$ et 1 pour les $\boldsymbol{V}$. A un nœud de type $\boldsymbol{J}_3$ correspondent donc 3 lignes de la matrice, respectivement 2 lignes pour un nœud $\boldsymbol{J}_2$, 1 ligne pour un nœud $\boldsymbol{J}_1$ et 1 ligne pour un nœud $\boldsymbol{V}$ portant une inconnue $\boldsymbol{V}$. Les équations portées par les nœuds $\boldsymbol{J}_3$ sont donc les plus volumineuses en espace mémoire. C'est pourquoi, la répartition de l'ensemble des nœuds restants commence par les nœuds restants $\boldsymbol{J}_3$, puis ce sont les nœuds $\boldsymbol{J}_2$ restants et les nœuds $\boldsymbol{J}_1$ restants.

Afin d'assurer un bon équilibrage des calculs, chacun des r_{J_3} premiers processus (entre les rangs 0 et $r_{J_3} - 1$) reçoit un des r_{J_3} nœuds $\boldsymbol{J}_3$ à traiter. De même, les r_{J_2} nœuds $\boldsymbol{J}_2$ sont attribués, un par un, aux r_{J_2} derniers processus (entre les rangs $N_P - 1$ et $N_P - r_{J_2}$). Quant aux r_{J_1} nœuds $\boldsymbol{J}_1$, ceux-ci sont également attribués, un par un, aux r_{J_1} derniers processus (entre les rangs $N_P - 1$ et $N_P - r_{J_1}$).

A ce stade, chaque processus peut remplir partiellement sa sous-matrice avec les termes des équations liées aux degrés de liberté des nœuds dont le traitement lui a été attribué. Ainsi, à titre d'exemple, les r_{J_3} premiers processus reçoivent chacun $n_{J_3} \times 3 + n_{J_2} \times 2 + n_{J_1} \times 1 + 3$ degrés de liberté.

Le reste des sous-matrices peut être rempli avec les degrés de liberté portés par les nœuds $\boldsymbol{V}$. Ainsi, on attribue à chaque processus un certain nombre de nœuds $\boldsymbol{V}$ à traiter. Ce nombre est calculé de manière à ce que l'ensemble des lignes restantes des sous-matrices correspondent toutes à un degré de liberté. Le traitement des nœuds $\boldsymbol{V}$ excédentaires est attribué au processus maître. En terme d'équilibrage des calculs, cela a peu d'incidence puisque le traitement de ces nœuds est rapide.

La figure 2.17 montre la répartition des nœuds entre quatre processus et la topologie de leurs sous-matrices avec $r_{J_3} = 2$, $r_{J_2} = 2$, $r_{J_1} = 1$.

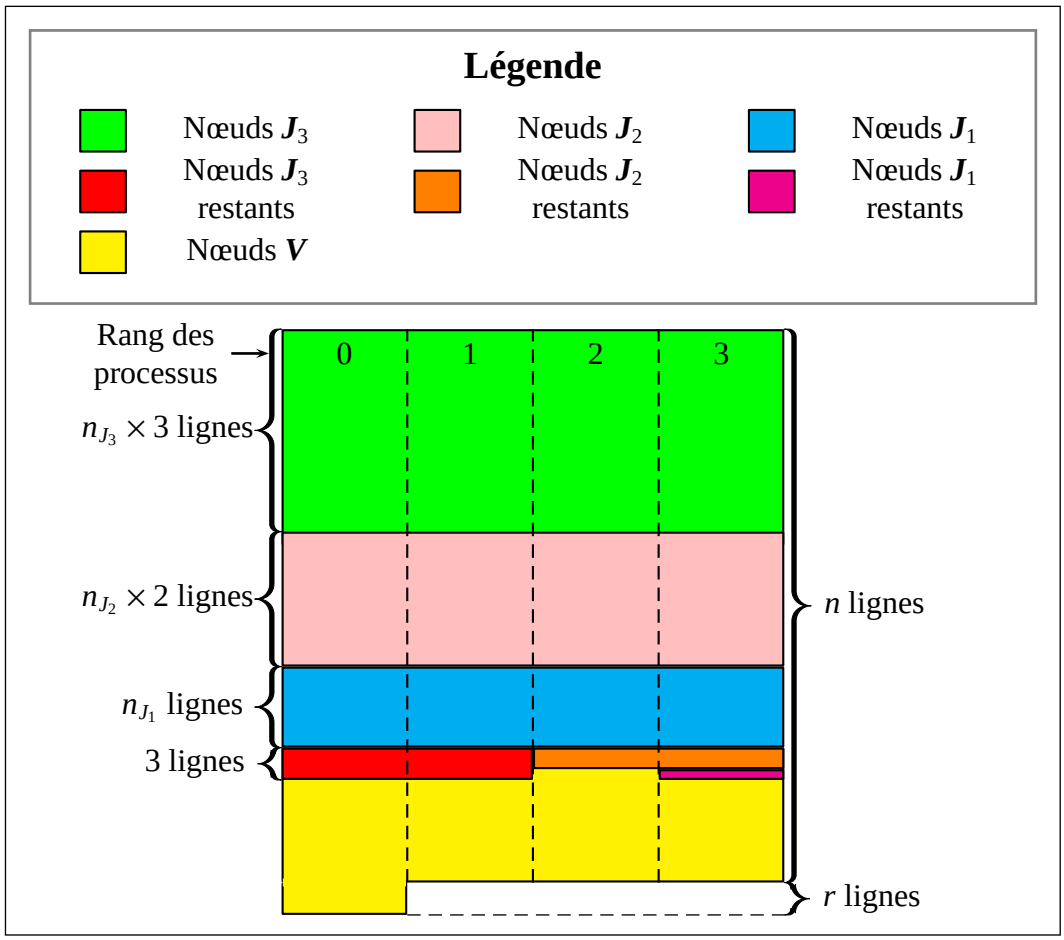


FIGURE 2.17 – Répartition du traitement des nœuds entre quatre processus et topologie des sous-matrices. Le processus de rang 0 est le maître.

En résumé, tous les processus construisent leur propre maillage et reçoivent de manière équitable n_{J_3} nœuds $\boldsymbol{J}_3$, n_{J_2} nœuds $\boldsymbol{J}_2$, n_{J_1} nœuds $\boldsymbol{J}_1$ et n_V nœuds $\boldsymbol{V}$. Pour optimiser l'équilibre de charge en calculs, les r_{J_3} premiers processus reçoivent un nœud $\boldsymbol{J}_3$ supplémentaire à traiter. De même, le r_{J_2} derniers processus reçoivent un nœud $\boldsymbol{J}_2$ supplémentaire et les r_{J_1} derniers processus un nœud $\boldsymbol{J}_1$ supplémentaire.

2.3.3 Résolution

Algorithme de Gauss-Jordan séquentiel

Le système linéaire à résoudre compte N inconnues complexes. Nous pouvons l'écrire sous la forme $\boldsymbol{M} \times \boldsymbol{X} = \boldsymbol{K}$, avec $\boldsymbol{M}$ la matrice du système, $\boldsymbol{X}$ le vecteur d'inconnues, et $\boldsymbol{K}$ le second membre. Pour le résoudre, nous utilisons l'algorithme de Gauss-Jordan avec recherche du pivot maximal à chaque étape du solveur qui conduit à la diagonalisation de la matrice $\boldsymbol{M}$[78]. Au fil de la résolution, $\boldsymbol{M}$ devient la matrice identité et $\boldsymbol{K}$ est modifié. Au final, la solution est contenue dans le vecteur

K. La recherche du pivot maximal n'est pas nécessaire en théorie mais, en pratique, elle permet de réduire l'impact de l'erreur numérique liée au fait que la précision des calculs est limitée par la précision de la machine. Le pivot maximal est le terme qui est le plus grand en valeur absolue. L'algorithme séquentiel de Gauss-Jordan est exprimé dans l'algorithme 1. La figure 2.18 présente la topologie de la matrice avant la $k+1^{ième}$ étape du solver.

Algorithme 1 : Algorithme séquentiel de Gauss-Jordan avec recherche systématique du pivot maximal.

Données : matrices **M** et **K**, nombre de degrés de liberté N
Résultat : Solution du système contenue dans **K**

```
1  pour k=0 à N-1 faire
2      Trouver la position (I_piv, J_piv) du pivot maximal dans la sous-matrice
       M[k : N-1 ; k : N-1];
3      Échanger les lignes k et I_piv;
4      Échanger les colonnes k et J_piv;
       // Diagonalisation du système
5      pour i = 0 à k-1 faire
6          pour j = k+1 à N-1 faire
7              M[i ;j] = M[i ;j] - M[i ;k] × M[k ;j] / M[k ;k];
8          K[i] = K[i] - M[i ;k] × K[k] / M[k ;k];
9      pour i = k+1 à N-1 faire
10         pour j = k+1 à N-1 faire
11             M[i ;j] = M[i ;j] - M[i ;k] × M[k ;j] / M[k ;k];
12         K[i] = K[i] - M[i ;k] × K[k] / M[k ;k];
13     pour j = k+1 à N-1 faire
14         M[k ;j] = M[k ;j] / M[k ;k];
15     K[k ;j] = K[k ;j] / M[k ;k];
```

Parallélisation de l'algorithme de Gauss-Jordan

Comme nous l'avons précédemment évoqué, certaines parties de l'algorithme de Gauss-Jordan sont parallélisables. C'est le cas de la recherche du pivot maximal (voir la ligne 2 dans l'algorithme 1) et des calculs de diagonalisation du système (à partir de la ligne 5), qui ne présentent pas de dépendance à condition que chaque processus connaisse le contenu de la ligne **k** après échange des lignes.

En revanche, il faut procéder à des communications pour déterminer la valeur et le repérage du pivot maximal global ainsi que pour effectuer l'échange des lignes. La matrice étant partagée selon les lignes (voir figure 2.16), l'échange des colonnes ne nécessite pas de communication. Cinq communications donc sont nécessaires.

Premièrement, chaque processus ayant trouvé son propre pivot local, une com-

$$\text{ligne } k \rightarrow \begin{bmatrix} 1 & 0 & 0 & \boldsymbol{M}_{0,k} & \cdot & \cdot & \boldsymbol{M}_{0,N-1} \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & \cdot \\ \vdots & \ddots & 1 & \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & 0 & \boldsymbol{M}_{k,k} & \cdot & \cdot & \boldsymbol{M}_{k,N-1} \\ \vdots & \vdots & \vdots & \cdot & \cdot & \cdot & \cdot \\ \vdots & \vdots & \vdots & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \boldsymbol{M}_{N-1,k} & \cdot & \cdot & \boldsymbol{M}_{N-1,N-1} \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \boldsymbol{K}_k \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$\text{colonne } k \downarrow$$

FIGURE 2.18 – Topologie de la matrice $\boldsymbol{M}$ avant la $k^{ième}$ étape de l'algorithme de Gauss-Jordan. Le terme $\boldsymbol{M}_{k,k}$ est le pivot. Les symboles $\cdot$ représentent les valeurs non-nulles de la matrice, les symboles $\ddots$ et $\vdots$ représentent les zéros.

munication collective **MPI_Gather**[9] est initiée à destination du maître. Celui-ci récupère dans un tableau les valeurs des pivot locaux, selon l'ordre des rangs des processus. Puis, le maître recherche le pivot maximal absolu dans ce tableau et le stocke dans la variable **pivot**. L'indice **rank_piv** de la position du pivot maximal dans le tableau indique le rang du processus qui l'a effectivement trouvé[10].

Deuxièmement, tous les processus procèdent à une nouvelle communication **MPI_Gather** à destination du maître. Celle-ci leur permet de lui communiquer les coordonnées absolues, *i.e.* dans la matrice complète du système, de leurs pivots locaux respectifs. Connaissant le rang du processus qui a trouvé le pivot maximal absolu, le maître peut donc déterminer les coordonnées (**I_piv**,**J_piv**) du pivot.

Afin de contrôler que la matrice est bien inversible, nous définissons par ailleurs une valeur **epsilon**, supérieure à l'erreur machine, et la comparons systématiquement à la valeur absolue du pivot maximal. Avant de poursuivre, le maître procède donc au test (**pivot** < **epsilon**). Si ce test est vrai, alors nous considèrons que le pivot est nul : la matrice est singulière. Dans ce cas, un mécanisme d'envoi d'erreur permet d'interrompre le programme proprement.

Troisième communication, si la matrice est inversible : le maître communique à tous les processus, *via* une communication **MPI_Bcast**, les coordonnées (**I_piv**, **J_piv**) du pivot. A partir de ces informations, chaque processus calcule le rang **rank_piv** du processus qui possède le pivot. Ce calcul est très rapide et permet de gagner du temps en évitant la communication de cette valeur. De la même manière, les processus calculent également le rank **rank_k** du processus qui héberge la ligne **k** et qui va être échangée avec la ligne **I_piv**.

9. Pour plus de renseignements au sujet de cette fonction, voir l'appendice A.

10. C'est parce qu'il est indispensable de connaître le rang du processus qui possède le pivot maximal que nous procédons ainsi. Sinon, si la seule connaissance de la valeur du pivot était nécessaire, l'emploi de l'opération de réduction **MPI_MAX** *via* **MPI_Reduce** serait tout à fait indiqué.

Quatrièmement, *via* une communication **MPI_Bcast** émise par le processus **rank_piv**, le contenu de la ligne **I_piv** de la matrice ***M*** ainsi que le terme correspondant dans le second membre ***K*** est envoyé à tous les processus.

Enfin, cinquièmement, si les rangs **rank_k** et **rank_piv** sont identiques, cela signifie que les lignes **I_piv** et **I_k** sont hébergées par un seul et même processus. L'échange de ces lignes ne requiert donc pas de communication. En revanche, si ce n'est pas le cas, le processus **rank_piv** envoie le contenu de la ligne **I_piv** au processus **rank_k**. Cette communication s'effectue grâce aux fonctions **MPI_Send** et **MPI_Recv**. Ainsi, l'échange des lignes a bien lieu.

Après ces échanges, tous les processus exécutent en parallèle, et de manière totalement indépendante, la diagonalisation de leur propre sous-matrice.

La figure 2.19 (page 82) illustre le diagramme correspondant à l'algorithme que nous venons de décrire.

Une fois la résolution terminée, la solution se trouve dans le vecteur K et est répartie sur l'ensemble des processus. Nous avons choisi de faire réaliser le post-traitement par le processus maître aussi une dernière communication **MPI_Gather** lui permet de réceptionner l'ensemble de la solution.

2.3.4 Post-traitement

Le post-traitement des calculs est effectué par le processus maître et permet de calculer, pour chaque brin d'un inducteur mutibrins :

- le courant total complexe $\boldsymbol{I}_b$ parcourant le brin, par moyenne des courants entrant et sortant aux extrémités du brin. A chaque extrémité, le courant est calculé par intégration de la densité de courant sur la section ;
- l'écart relatif (en %) des courants entrant et sortant du brin, ce qui nous donne un critère de contrôle de la précision du calcul de la loi d'Ohm (2.39) ;
- le courant efficace I_{eff_b} ;
- la puissance Joule perdue $\mathcal{P}_b$;
- la fraction de puissance Joule perdue dans le brin, par rapport à la puissance Joule totale perdue dans l'ensemble de l'inducteur ;
- la résistance électrique.

La puissance Joule totale $\mathcal{P}_b$ dissipée dans un brin b est calculée par intégration sur le volume de cet objet selon la formule (2.50)

$$\mathcal{P}_b = \sum_{e=1}^{NBE_b} \sum_{k=1}^{NPI_e} \frac{\vec{\boldsymbol{J}}_k \cdot \vec{\boldsymbol{J}}_k^*}{2\sigma} \det Jac_k \; w_k \tag{2.50}$$

avec NBE_b le nombre d'éléments $\vec{\boldsymbol{J}}$ que compte le maillage du brin b et $\vec{\boldsymbol{J}}_k^*$ le complexe conjugué de $\vec{\boldsymbol{J}}_k$, la densité de courant au point d'intégration k.

La résistance électrique R_b d'un brin b est calculée grâce à la relation (2.51).

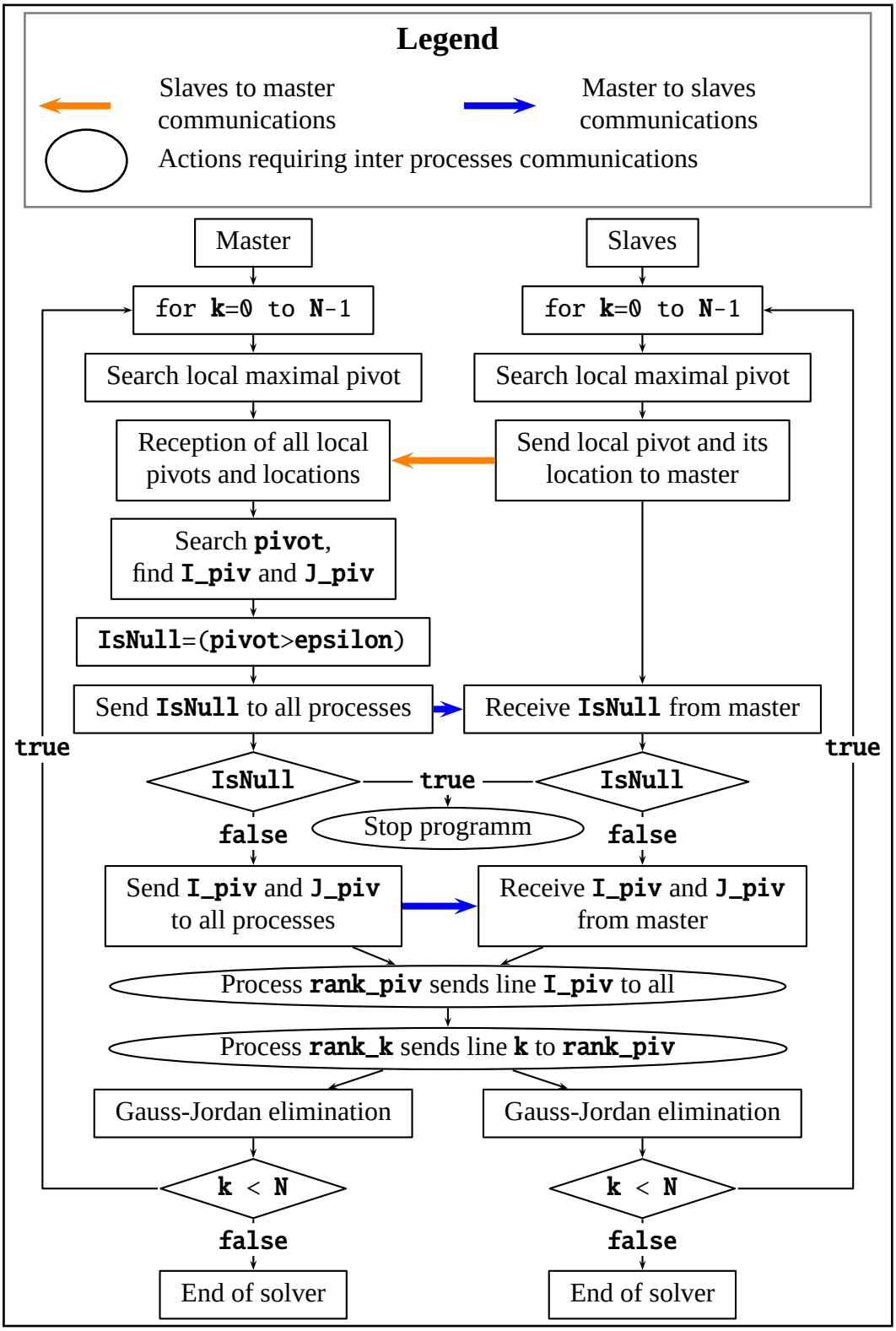


FIGURE 2.19 – Algorithme parallèle de Gauss-Jordan avec recherche du pivot maximal.

$$\mathrm{R}_b = \frac{\mathcal{P}_b}{I^2_{eff_b}} \tag{2.51}$$

A l'échelle de l'inducteur, le logiciel calcule :
— le courant total complexe $\boldsymbol{I}_{induc}$, égal à la somme des courants dans les brins ;
— l'écart relatif (en %) des courants entrant et sortant ;
— le courant efficace ;
— la puissance Joule perdue $\mathcal{P}_{induc}$;
— l'impédance complexe de l'inducteur $\boldsymbol{Z}$;
— la résistance équivalente R_{induc} et l'inductance équivalente L_{induc} déduites de $\boldsymbol{Z}$.

La puissance Joule totale dissipée dans l'inducteur est calculée par la somme des puissances dissipées dans les brins selon la formule (2.52)

$$\mathcal{P}_{induc} = \sum_{b=1}^{N_{brins}} \mathcal{P}_b \tag{2.52}$$

avec N_{brins} le nombre de brins que compte l'inducteur.

L'inducteur étant soumis à une différence de potentiel $\Delta\boldsymbol{V}$, l'impédance complexe de l'inducteur est calculée par la formule (2.53).

$$\boldsymbol{Z} = \frac{\Delta\boldsymbol{V}}{\sum_{b=1}^{N_{brins}} \boldsymbol{I}_b} \tag{2.53}$$

D'autre part, les grandeurs locales sont stockées dans des fichiers au format *vtk*. Nous créons un fichier pour les valeurs associées au maillage $\boldsymbol{V}$ et un fichier pour celles liées au maillage $\vec{\boldsymbol{J}}$. Ainsi, nous pouvons visualiser la répartition des grandeurs suivantes :
— les parties réelles et imaginaires de $\boldsymbol{V}$;
— les parties réelles et imaginaires de $\vec{\boldsymbol{J}}$;
— le module de $\boldsymbol{V}$;
— le module de $\vec{\boldsymbol{J}}$;
— la densité de puissance Joule.

La visualisation de la géométrie, du maillage et des grandeurs locales est réalisée grâce au logiciel Paraview®[76].

Pour décrire de manière plus concrète l'utilisation de ModeLitz, l'annexe C expose un exemple complet minimal de calcul, depuis la description de la simulation jusqu'au post-traitement.

2.4 Validation du logiciel

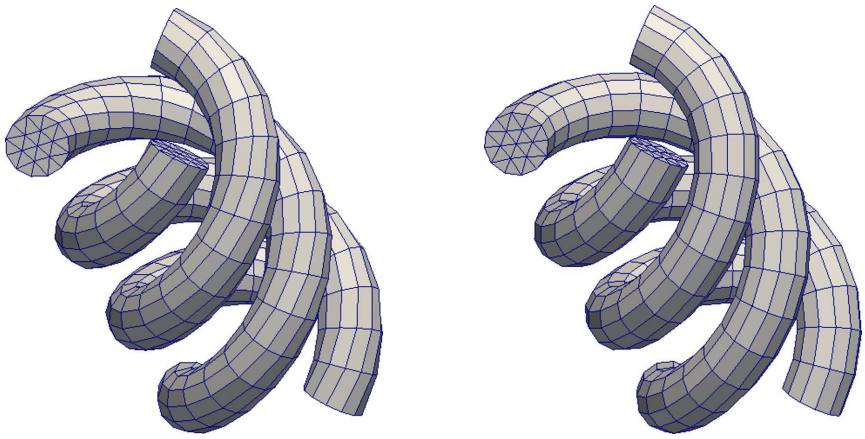
Nous présentons ici les tests que nous avons menés afin de valider l'implémentation du logiciel ModeLitz. Nous commencerons par comparer les résultats de simulations menées à la fois sous ModeLitz et sous le logiciel MIGEN [79]. Puis, nous

étudierons l'impact du maillage sur la qualité des résultats. Enfin, nous analyserons les performances de notre logiciel.

2.4.1 Comparaison avec MIGEN

MIGEN est un logiciel de calculs électromagnétiques 2D et 3D dédié à la modélisation de systèmes inductifs et basé sur la méthode intégrale. Ce logiciel, dont la validation a été attestée par comparaison à des mesures expérimentales, permet notamment la modélisation de certaines géométries classiques d'inducteurs (hélice, spirale,...). En revanche, la gestion des objets géométriques ne permet de pas d'implémenter des objets complexes et composites de type fil de Litz. De plus, le fonctionnement séquentiel de ce logiciel présente les inconvénients de la méthode intégrale évoqués dans la partie 2.2.6.

Nous comparons ci-après les résultats obtenus par MIGEN et ModeLitz sur une géométrie simple. Soit un ensemble de trois brins en cuivre d'un rayon de 25 µm. Les brins sont agencés en hélice de rayon 75 µm (distance entre l'axe de l'hélice et le centre d'un brin). La portion de câble ainsi formée est modélisée sur une longueur égale au pas de l'hélice formée par chacun des brins soit 0.3 mm. La tension aux bornes des brins est fixée à 1 mV et la fréquence d'alimentation à 200 kHz. Les maillages $\boldsymbol{V}$ des trois brins pour les deux logiciels sont présentés sur la figure 2.20.



(a) MIGEN (b) ModeLitz

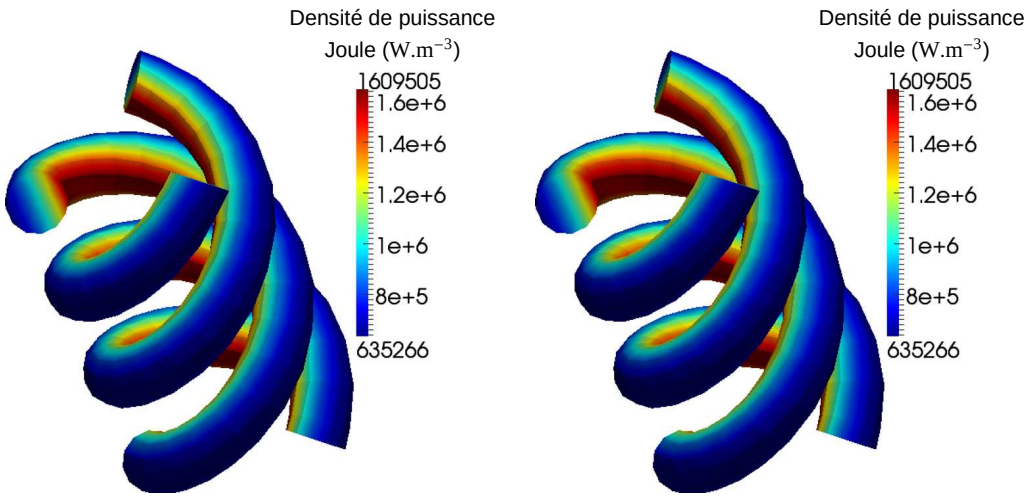
FIGURE 2.20 – Comparaison des maillages construits par MIGEN et par ModeLitz.

Dans un premier temps, nous avons mené un travail d'implémentation visant à permettre à ModeLitz d'effectuer une simulation réalisée sous MIGEN, notamment en chargeant le maillage généré par ce dernier. Nous comparons donc ici les résultats des simulations obtenues par chacun des deux outils et à partir d'un maillage identique, le maillage de la figure 2.20a.

Les deux simulations étant basées sur le même maillage, le système linéaire construit par ModeLitz doit correspondre à celui généré par MIGEN. Nous avons

donc comparé le contenu des systèmes linéaires avant résolution. Nous avons trouvé un accord parfait entre les deux logiciels, la différence entre les termes de chaque matrice étant proche de l'erreur machine. Ceci nous permet donc de conclure à la conformité de l'implémentation parallèle de la méthode intégrale sous ModeLitz.

La figure 2.21 montre que les deux codes calculent une même répartition 3D de la densité de puissance Joule dans les brins. La table 2.2 confirme l'excellente concordance des deux logiciels sur le calcul des grandeurs globales. L'écart relatif entre MIGEN et ModeLitz est de $7{\cdot}10^{-7}$% sur la partie réelle du courant, de $4{\cdot}10^{-5}$% sur la partie imaginaire du courant et de $4 \cdot 10^{-6}$% sur la puissance Joule.



(a) MIGEN (b) ModeLitz

FIGURE 2.21 – Comparaison des résultats fournis par MIGEN (a) et par ModeLitz (b) sur la répartition de la densité de puissance Joule à partir de calculs basés sur le maillage de MIGEN.

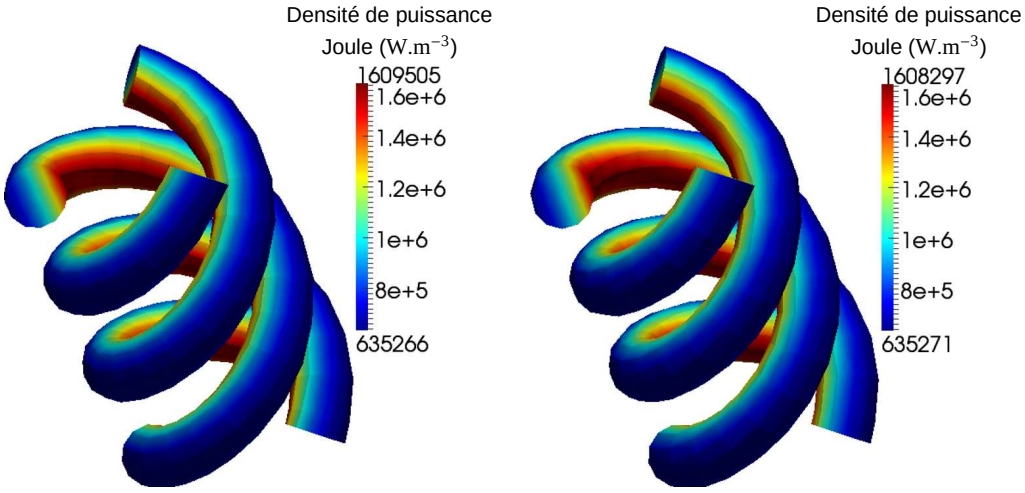
	MIGEN	ModeLitz
Courant, partie réelle (A)	5.6838481154e-2	5.6838482671e-2
Courant, partie imaginaire (A)	-5.1603381674e-3	-5.1603296192e-3
Puissance Joule (W)	2.7123138882e-6	2.7123140090e-6

TABLE 2.2 – Comparaison des grandeurs globales obtenues par MIGEN et ModeLitz à partir de modélisations réalisées sur le maillage de MIGEN.

Nous constatons donc un très bon accord entre MIGEN et ModeLitz quand la simulation est menée en s'appuyant sur le même maillage. Ceci valide complètement l'implémentation de la méthode intégrale et le post-traitement des résultats dans le logiciel parallèle ModeLitz.

Nous comparons maintenant les résultats des calculs menés par chaque logiciel sur son propre maillage (voir la figure 2.20). La figure 2.22 compare la répartition de

la densité de puissance obtenue par les deux logiciels. La table 2.3 met à nouveau en évidence un bon accord entre les deux logiciels sur le calcul des grandeurs globales, les écarts relatifs sur les parties réelles et imaginaires du courant et sur la puissance Joule étant respectivement de 0.02%, de 1.57% et de 1.16%. Les écarts entre les deux logiciels sont ici un peu plus importants car les maillages sont différents.



(a) MIGEN (b) ModeLitz

FIGURE 2.22 – Comparaison de la répartition de la densité de puissance Joule calculée indépendamment par MIGEN (a) et ModeLitz (b).

	MIGEN	ModeLitz
Courant, partie réelle (A)	5.6838481154e-2	5.6826542507e-2
Courant, partie imaginaire (A)	-5.1603381674e-3	-5.2421055419e-3
Puissance Joule (W)	2.7123138882e-6	2.7439411291e-6

TABLE 2.3 – Comparaison des grandeurs globales obtenues par MIGEN et ModeLitz à partir de modélisations réalisées sur leurs maillages respectifs.

La figure 2.23a présente la répartition du module de la densité de courant $|\vec{\boldsymbol{J}}|$ au sein des brins et la figure 2.23b montre la répartition du module du potentiel électrique $|\boldsymbol{V}|$. Nous pouvons remarquer que le courant est tangent aux chemins des brins.

Les tests présentés ci-dessus permettent de vérifier la qualité des calculs menés par le logiciel ModeLitz et de valider son implémentation.

2.4.2 Sensibilité due au maillage

Nous étudions à présent l'impact de la qualité du maillage sur les calculs. Pour ce faire nous réalisons deux études de sensibilité au maillage : l'une sur un inducteur

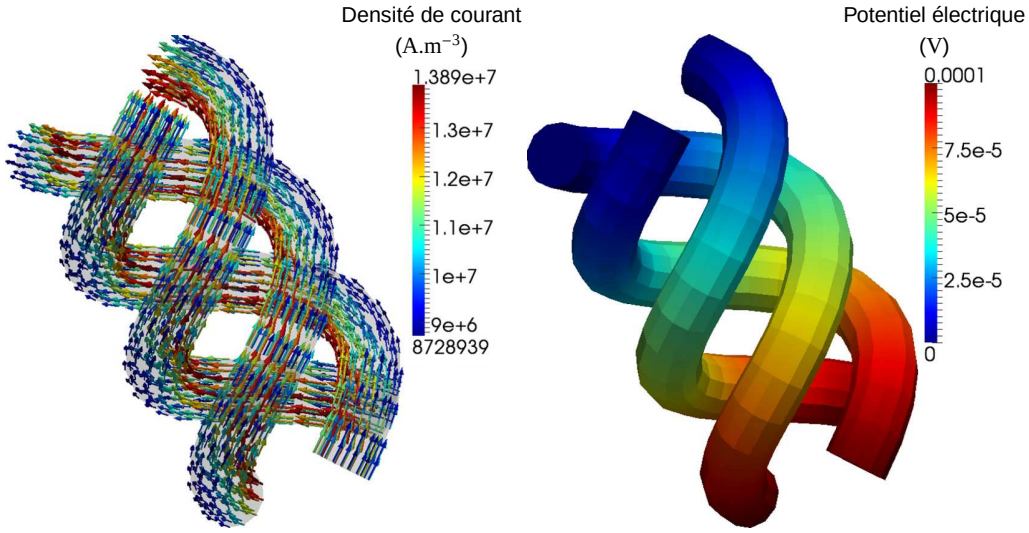


FIGURE 2.23 – Répartition de la densité de courant (module) (a) et du potentiel électrique (module) (b) dans les brins (résultats issus des calculs menés avec ModeLitz).

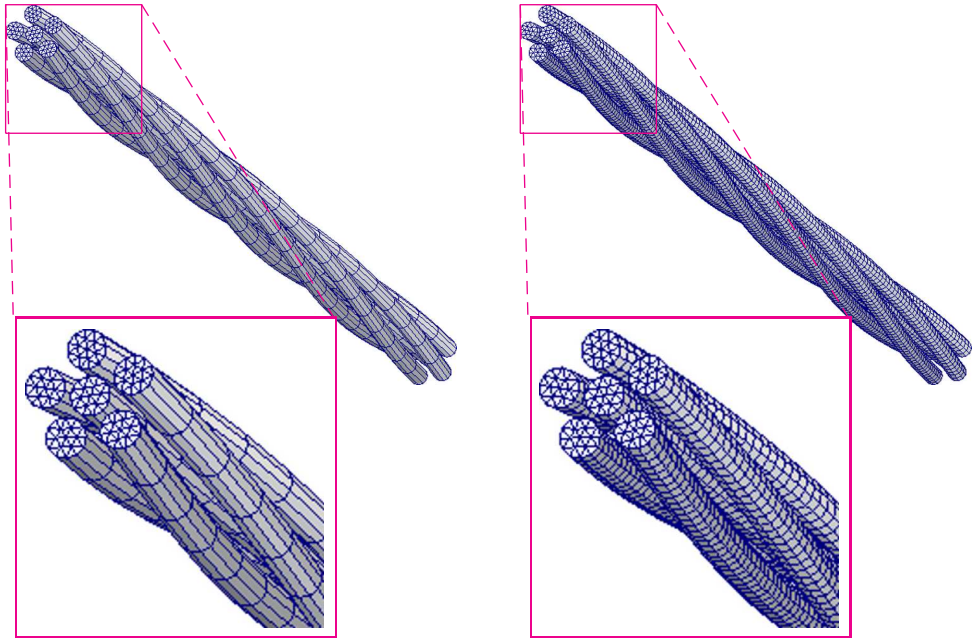
droit à six brins arrangés en hélices (que nous désignerons par **(A)**), une autre sur un inducteur droit en fil de Litz à six brins **(B)**.

Dans ces deux études, les brins sont en cuivre et ont un rayon r de 25 µm. Chaque brin est entouré d'un isolant d'épaisseur 5 µm. Le pas de torsadage des brins de l'inducteur **(A)** est de 4 mm de même que la longueur d'inducteur modélisé. L'inducteur **(B)** est agencé suivant un arrangement 3×2 brins. Au sein d'un paquet, les brins ont un pas de torsadage de 2 mm et chaque paquet de deux brins est torsadé suivant un pas de 4 mm. Chaque niveau de paquet est torsadé suivant le sens trigonométrique (+1). Chaque inducteur est soumis à une tension de 0.1 V.m^{-1}. L'inducteur **(A)** est alimenté à une fréquence de 300 kHz ($\delta/r = 4.75$) et l'inducteur **(B)** à une fréquence de 200 kHz ($\delta/r = 5.81$). L'épaisseur de peau étant grande devant le rayon des brins, l'étude de l'influence du raffinement du maillage de la section des brins n'est pas nécessaire.

Dans les deux cas, nous étudions l'influence de la discrétisation curviligne des brins en faisant varier le nombre de subdivisions le long de leurs chemins. Nous commençons avec un maillage relativement lâche comptant 25 divisions. Pour rendre la compréhension plus aisée, nous raffinons le maillage suivant un coefficient de raffinement c_{raff} : le nombre de divisions est donc égal à $c_{raff} \times 25$. Les figures 2.24 et 2.25 montrent les maillages obtenus suivant divers coefficients de raffinement, respectivement pour l'arrangement en hélice et pour l'arrangement en fil de Litz.

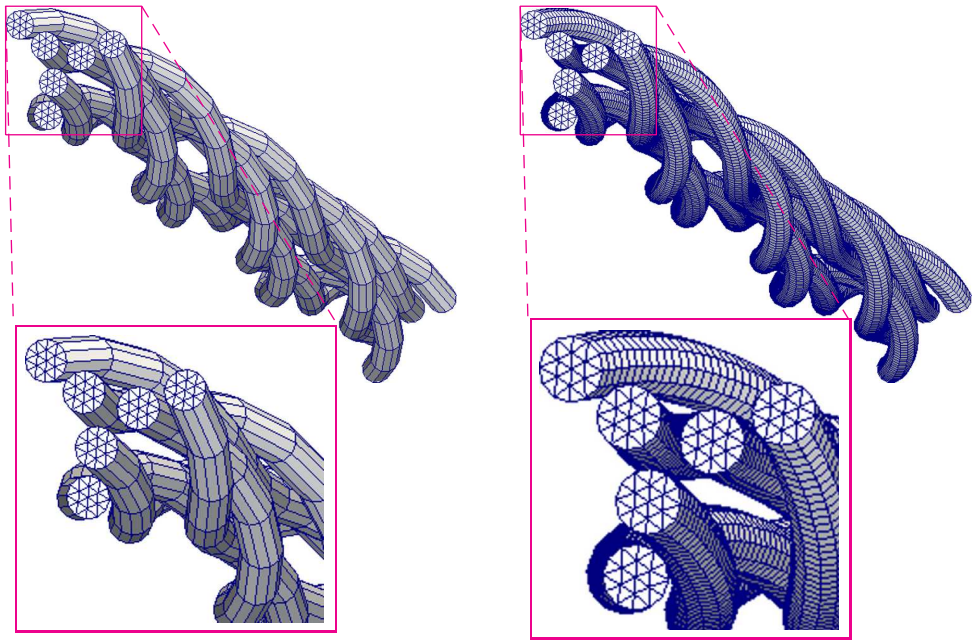
Nous nous intéressons à l'impact du raffinement du maillage sur les calculs. La figure 2.26 montre la convergence de la résistance et de l'inductance équivalentes linéiques en fonction du coefficient c_{raff} pour les deux types d'inducteur.

Nous remarquons une convergence plus rapide des grandeurs calculées dans le



(a) $c_{raff} = 1$ (b) $c_{raff} = 6$

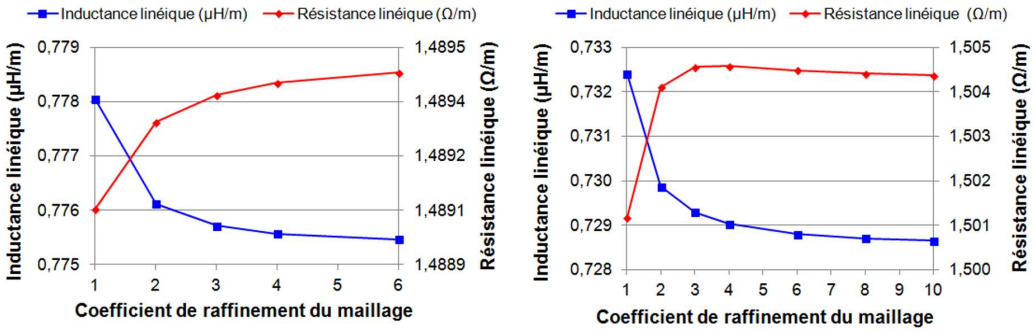
FIGURE 2.24 – Maillages des simulations tests d'inducteur arrangés en hélice **(A)** pour différents coefficients de raffinement.



(a) $c_{raff} = 1$ (b) $c_{raff} = 10$

FIGURE 2.25 – Maillages des simulations tests d'inducteur en fil de Litz 3×2 **(B)** pour différents coefficients de raffinement.

cas de l'inducteur **(B)**. Dans les deux cas, nous constatons que, même si le maillage est relativement lâche, les calculs nous donnent une bonne estimation des ordres de



(a) Inducteur **(A)** (b) Inducteur **(B)**

FIGURE 2.26 – Profils de convergence, en fonction du coefficient de raffinement du maillage c_{raff}, de la résistance et de l'inductance équivalentes linéiques pour l'inducteur à brins en hélice (a) et pour l'inducteur en fil de Litz (b).

grandeurs de la résistance et de l'inductance équivalentes linéiques des inducteurs. Nous estimons que, dans ce cas, un coefficient $c_{raff} = 3$ constitue un bon compromis entre taille du maillage et qualité des résultats. En effet, avec ce coefficient et en considérant que les valeurs « vraies » sont les valeurs convergées, calculées avec le maillage le plus fin, les écarts relatifs sur l'inductance et sur la résistance avec les valeurs convergées sont respectivement de 0.03 % et de 0.004 % pour l'inducteur **(A)**, de 0.09 % et de 0.01 % pour l'inducteur **(B)**. Même dans le cas le moins favorable, c'est-à-dire avec $c_{raff} = 1$, les écarts relatifs sur les inductances et les résistances pour les deux types d'inducteurs sont inférieurs à 1 %.

Dans ModeLitz, le maillage d'un brin peut être piloté soit en fixant le nombre de subdivisions le long de celui-ci, soit en indiquant une longueur de subdivision. Cette longueur peut être définie comme proportionnelle au rayon du brin. Nous pouvons ainsi estimer que la longueur optimale d'élément correspondante au coefficient $c_{raff} = 3$ que nous venons de déterminer est environ égale à 2 fois le rayon du brin [11]. Ceci ne constitue pas une règle absolue mais donne un critère facile à utiliser pour réaliser une simulation. En fonction de la taille du système obtenu et de la qualité des résultats désirée, il peut être possible de modifier quelque peu la taille des éléments sans craindre une dégradation importante de la précision des calculs.

2.4.3 Étude des performances

Protocole

Nous avons réalisé des tests de performances du code ModeLitz sur diverses plate-formes : une station de travail multi-cœurs et deux clusters. Nous avons étudié

11. Pour indication, un coefficient c_{raff} de 1 correspond à une taille d'élément environ égale à 6.5 fois le rayon du brin.

l'influence du parallélisme en faisant varier le nombre de processus et en mesurant le temps d'exécution du programme pour diverses simulations.

Étant donné que le déroulement du programme, pour l'ensemble des processus, est essentiellement rythmé par le processus maître, en particulier dans l'exécution du solveur, nous mesurons le *temps CPU* de calcul du maître. Le temps *CPU*, appelé logiquement *CPU time* par les systèmes d'exploitation, correspond au temps que le processeur a véritablement consacré à l'exécution du processus. Ce temps est toujours inférieur au temps *elapsed time*, le temps pendant lequel l'utilisateur patiente en attendant que le processus se termine. Cette différence tient au fait que, comme nous l'avons évoqué dans la partie 1.3.8, plusieurs processus peuvent être exécutés sur un même cœur.

Ainsi, par exemple, si deux processus particulièrement gourmands se partagent le même cœur, avec le même degré de priorité [12], ce dernier travaille la moitié du temps sur l'un et l'autre moitié du temps sur l'autre. Au final, le temps *elapsed time* pour ces deux calculs est environ deux fois plus important que le temps *CPU*.

En revanche, si un processus gourmand en calcul est exécuté sur un cœur qui n'est pas très chargé, c'est-à-dire que l'ensemble des processus déjà présents l'accaparent peu, ce cœur consacre alors la majeure partie de son temps au processus en question. Dans ce cas, les temps *CPU time* et *elapsed time* sont proches.

Étude de l'influence du parallélisme sur l'accélération des calculs

La figure 2.27 montre l'évolution de l'accélération des calculs en fonction du nombre de processus sur une station de calcul HP-Z800 pour une simulation comptant 17 796 degrés de liberté. Cette station est équipée de deux processeurs Intel® Xeon® X5680 (6 cœurs avec *multithreading*, 3.33 GHz). Avec le *multithreading*, nous avons au total 2 CPU $\times$ 6 cœurs $\times$ 2 threads soit 24 cœurs virtuels. L'espace mémoire occupé par le test avec un seul processus est de 5.3 Go. Ceci correspond à une exécution purement séquentielle.

Nous constatons que l'accélération de l'étape de construction du système suit une loi quasi linéaire (loi en puissance 0.95) en fonction du nombre de processus tant que le nombre de processus est inférieur à 13, ce qui est très proche de l'accélération maximale théorique. Ce comportement était attendu puisque cette étape est entièrement parallélisée. En revanche, il est surprenant de constater qu'à partir d'une parallélisation sur 13 processus, les performances semblent se dégrader et l'accélération de cette étape est sujette à variations. Compte tenu du fait que la machine compte 12 processeurs physiques, nous suspectons un effet inattendu du *multithreading*. Nous reviendrons plus en détail sur ce sujet dans la partie 2.4.3.

Nous observons également que l'augmentation du nombre de processus n'est pas aussi efficace pour accélérer la résolution du système, ceci en raison des nom-

12. Les sytèmes d'exploitation disposent d'un système de priorité permettant de régler la proportion de temps qu'un processeur consacre à chacun des processus qu'il exécute.

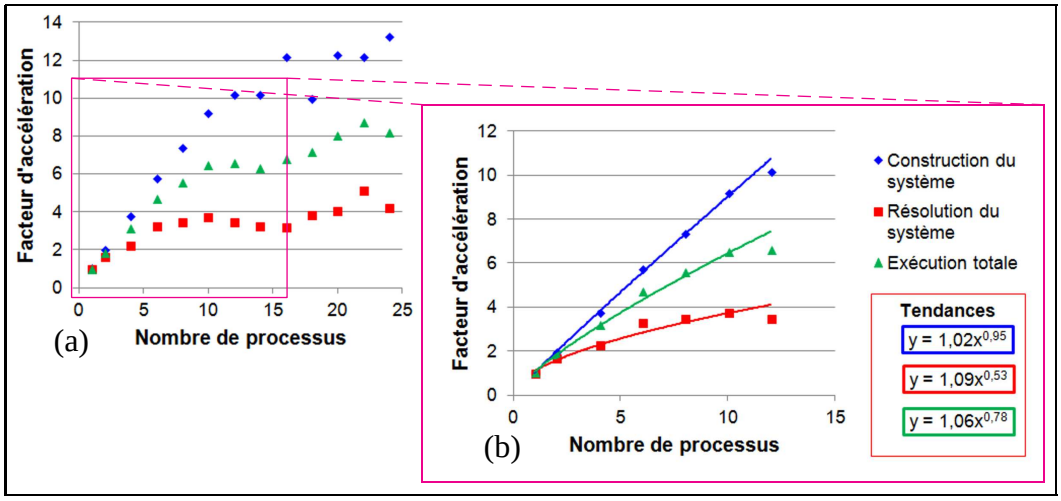


FIGURE 2.27 – Evolution de l'accélération des calculs en fonction du nombre de processus sur la station de travail HP-Z800 de 1 à 24 processus (a) et de 1 à 12 processus (b).

breuses communications que requiert cette étape. Au final, entre un et 12 processus, l'accélération totale suit une loi en puissance 0.78, ce qui est satisfaisant.

Nous avons également réalisé des tests sur le calculateur Ada de l'IDRIS [13]. Ce cluster est constitué de 332 nœuds de calculs comptant chacun quatre processeurs Intel® Xeon® E5-4650 (8 cœurs sans *multithreading* [14], 2.7 GHz) soit 332 nœuds × 4 CPU × 8 cœurs = 10 624 cœurs physiques. La figure 2.28 montre l'impact de la parallélisation, entre 40 et 360 processus, sur l'accélération d'un calcul comptant 54 840 degrés de liberté. Les facteurs d'accélération sont ici calculés en prenant comme référence les temps de calcul du cas parallélisé sur 40 processus. Le test mené avec 40 processus requiert un espace de 95 Go de mémoire vive.

Nous notons encore une fois l'excellente performance de la construction du système qui croît linéairement avec le nombre de processus. Le test parallélisé sur 40 processus a une durée effective de 15 heures 10 minutes environ, ce qui correspond environ à un temps de calcul cumulé de 606 heures 40 minutes [15]. Quant au cas à 360 processus, il a une durée effective de 2 heures 48 minutes, soit en cumulé 1 008 heures. Cette augmentation de la durée cumulée est clairement liée au temps des communications nécessaires à la résolution, comme le montre la non-linéarité de la loi d'accélération de la résolution présentée sur la figure 2.28. L'accélération totale varie selon une loi en puissance 0.797. Les bonnes performances du logiciel sont donc confirmées. De plus, la proximité des résultats entre les deux études précédentes, menées sur deux architectures assez différentes, reflète la bonne portabilité de ModeLitz.

Nous pouvons néanmoins noter que dans l'étude sur le cluster Ada, nous n'avons

13. Institut du Développement et des Ressources en Informatique Scientifique
14. Les processeurs E5-4650 peuvent supporter le *multithreading*, mais l'option n'est pas activée.
15. Calculé simplement en multipliant le temps de calcul par le nombre de processus.

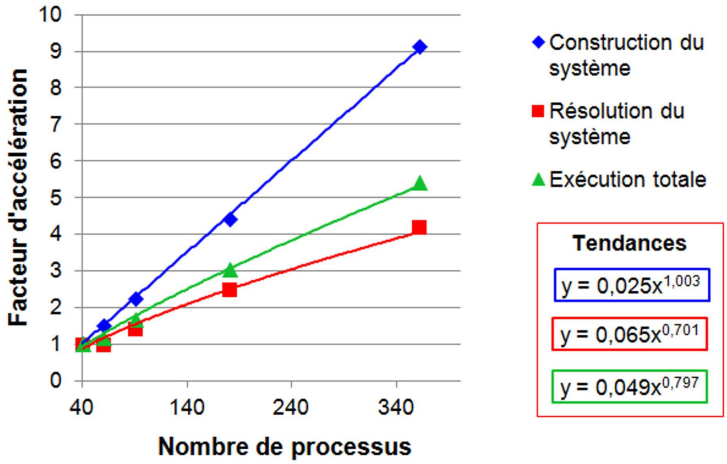


FIGURE 2.28 – Evolution de l'accélération des calculs en fonction du nombre de processus sur le cluster de calcul Ada.

pas observé de dégradation des performances, notamment en ce qui concerne l'étape de construction du système. Or, nous avons précisé que, sur cette machine, le *multithreading* est désactivé. Cette constatation tend à confirmer l'interprétation de la dégradation des performances sur la station de calcul HP-Z800.

Remarques sur l'intérêt du *multithreading*

Le laboratoire EPM disposant de son propre cluster, nous avons mené une étude plus poussée de l'impact du *multithreading* sur les performances du logiciel. Ce cluster est composé de quatre nœuds hébergeant chacun deux processeurs Intel® Xeon® E5-2620 v2 (10 cœurs avec *multithreading*, 2.5 GHz) soit 4 nœuds $\times$ 2 CPU $\times$ 10 cœurs $\times$ 2 threads = 160 cœurs virtuels. Les nœuds sont interconnectés à la fois par un réseau FastEthernet (10-100 Mbits/s) et par un réseau InfiniBand (1000 Mbits/s). Le choix du réseau à emprunter dépend du compilateur employé pour générer l'exécutable du programme. Nous avons utilisé le réseau Ethernet.

La configuration simulée compte 19 866 degrés de liberté et le test lancé sur un processus occupe 6.5 Go de RAM. Pour évaluer l'impact du *multithreading*, nous nous observons essentiellement l'évolution de l'accélération de la construction du système puisque cette étape est entièrement parallélisée.

La figure 2.29 montre l'évolution de l'accélération des différentes étapes du calcul en fonction du nombre de processus, entre 1 et 40 processus, quand le calcul est lancé sur un seul nœud du cluster. Nous constatons que l'accélération de la construction du système est idéale et prédictible (loi linéaire) entre 1 et 19 processus. Au-delà, en revanche, celle-ci devient aléatoire. Il apparaît donc que la technologie *multithread* ne permet pas systématiquement d'accélérer les calculs et semble avoir un impact dès que le nombre de processus choisi atteint le nombre de processeurs physiques du nœud.

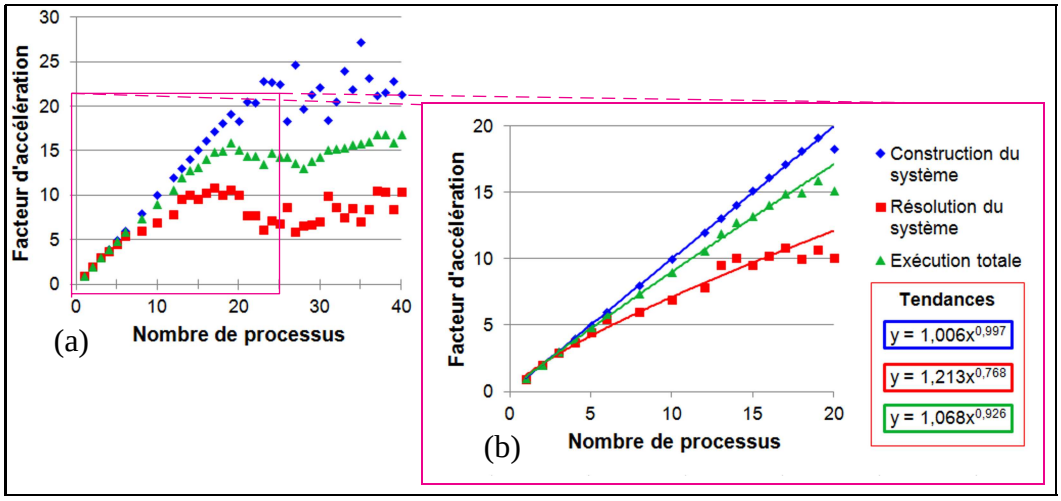


FIGURE 2.29 – Evolution de l'accélération des calculs en fonction du nombre de processus sur un seul nœud du cluster du laboratoire.

Pour vérifier cela, nous avons mené une nouvelle étude avec le même cas test, en utilisant cette fois les quatre nœuds du cluster. Nous avons veillé à charger les nœuds de la manière la plus uniforme possible. Ainsi, pour le test à quatre processus, chaque nœud reçoit un processus. Pour le test à cinq processus, nous distribuons deux processus sur le nœud 1 et un processus sur les autres nœuds, et ainsi de suite jusqu'à 80 processus. Les résultats de cette étude sont rapportés dans la figure 2.30.

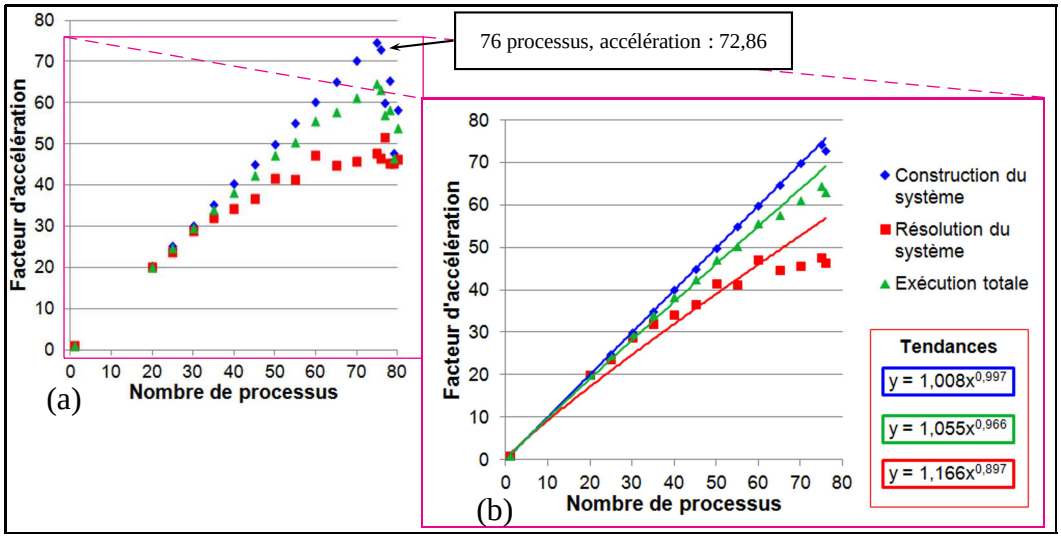


FIGURE 2.30 – Evolution de l'accélération des calculs en fonction du nombre de processus sur les quatre nœuds du cluster du laboratoire. Les processus sont répartis le plus équitablement possible entre les nœuds.

Nous pouvons constater une évolution quasi linéaire de l'accélération de la construction du système jusqu'à 76 processus. Au-delà, les performances sont considérablement dégradées. Pour les cas tests entre 77 et 80 processus, la répartition des

processus sur les quatre nœuds est telle qu'au moins un nœud reçoit 20 processus, c'est-à-dire un nombre égal au nombre de processeurs physiques qu'il possède.

Nous pouvons donc conclure de manière certaine que la dégradation des performances rapportée sur la figure 2.29 est due au *multithreading*. De plus, la variabilité des performances est telle qu'il n'est pas possible de prédire si, pour une parallélisation sur un nombre N_p de processus, le gain en temps de calcul sera meilleur ou moins bon qu'avec $N_p - 1$ ou $N_p + 1$ processus

D'autre part, nous constatons que lorsque le nombre de processus exécutés sur un nœud est égal au nombre de processus physiques de ce nœud, l'influence du *multithreading* commence a se faire sentir. Au cours de cette étude, l'option *multithread* était activée pour tous les nœuds. Il pourrait être intéressant de vérifier si l'accélération de la construction du système pour les cas tests entre 77 et 80 processus est linéaire quand cette option est désactivée.

Néanmoins, ces études nous permettent de constater à nouveau les bonnes performances du logiciel, prédictibles de surcroît, tant que celui-ci est exécuté sur un nombre de processus tel que le recours au *multithreading* n'est pas nécessaire.

Influence de la taille du système linéaire sur les temps de calcul

Nous nous intéressons à présent à l'évolution de la durée des calculs selon la taille du système linéaire. Nous nous basons sur une simulation d'un câble 3×2 brins dont nous raffinons peu à peu le maillage. Avec un maillage raffiné 10 fois, ce calcul compte 68 166 degrés de liberté répartis comme suit :

— 56 880 degrés de liberté correspondants aux composantes des inconnues $\vec{\boldsymbol{J}}$;
— 11 286 degrés de liberté correspondants aux inconnues $\boldsymbol{V}$.

La figure 2.31 montre l'évolution des temps d'assemblage du système et de la résolution en fonction de la taille du système pour des calculs parallélisés sur 72 processus sur le cluster du laboratoire. Nous choisissons ce nombre de processus afin de profiter au maximum du parallélisme tout en évitant la perturbation éventuelle de l'étude par le *multithreading*.

Nous constatons que les temps de construction du système et de résolution varient respectivement en puissance 1.82 et 2.94 du nombre d'inconnues. Ces comportement correspondent aux prévisions faites dans la partie 2.2.6. Plus le nombre de degrés de liberté est important, plus le temps total du calcul tend à évoluer comme le cube de ce nombre en raison des calculs de résolution du système linéaire. Ajoutons que le temps de construction du maillage du calcul le plus gros (maillage raffiné 12 fois, 81 966 degrés de liberté) est de 90 secondes, ce qui est négligeable devant les durées autres étapes de calcul.

Bilan des performances

Les tests de performances du logiciel ModeLitz correspondent aux attentes évoquées initialement, notamment en ce qui concerne l'accélération du temps d'assem-

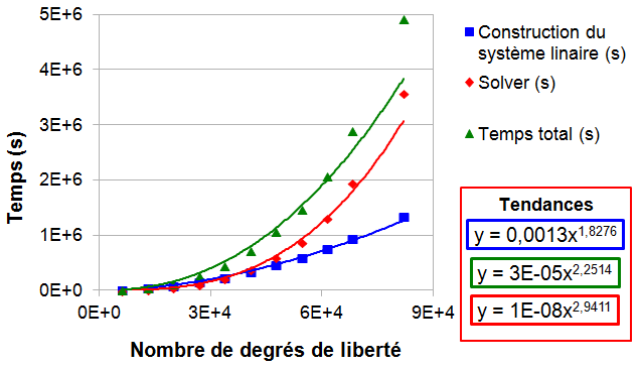


FIGURE 2.31 – Évolution du temps de calcul parallélisé sur 72 processus en fonction du nombre de degrés de liberté du système linéaire.

blage du système. Dans le cas où le choix de l'architecture de la machine et/ou du nombre de processus fixé pour l'exécution du logiciel permettent d'éviter le recours au *multithreading*, nous pouvons prédire une évolution de l'étape de construction du système quasi proportionnelle au nombre de processus. Pour ce qui est de la résolution du système, la réduction du temps de calcul est plus difficile à prédire car elle dépend notamment du nombre et du type de communications nécessaires, facteurs qui varient suivant le système à résoudre et le nombre de processus choisi. De plus, la vitesse des communications dépend du débit réel du réseau connectant les processeurs et du trafic externe au logiciel transitant à travers ce réseau. Néanmoins, l'étude des performances que nous avons menée nous permet d'établir une loi de comportement de l'accélération des calculs en fonction du nombre N_p de processus. Pour la construction du système, l'accélération varie linéairement selon N_p. Pour ce qui est de l'exécution totale, l'accélération suit une loi en puissance proche de $N_p^{0.8}$.

Outre la station de calcul individuelle et les clusters, nous avons pu également exécuter des calculs parallélisés sur plusieurs stations connectées *via* le réseau interne du laboratoire. Ces stations sont assez hétérogènes car pilotées par des systèmes d'exploitation Linux différents et équipées de CPU différents. Il est difficile de mener une étude rigoureuse des performances sur une telle architecture. En effet, ce type de réseau (Ethernet) est utilisé principalement à des fins de connexion des machines au réseau local et à Internet et est donc emprunté simultanément par plusieurs utilisateurs. Une étude de performances ne pourrait donc avoir de sens qu'à la condition d'accaparer temporairement et exclusivement les machines. Néanmoins, nous pouvons affirmer que, sur une architecture de ce type, intermédiaire entre la station individuelle et le cluster, les performances sont moindres en raison essentiellement des transferts *via* le réseau, puisque celui-ci n'est pas optimisé. Toutefois, même sur ce type d'architecture moins efficace, le logiciel fonctionne parfaitement. Cette solution permet de profiter d'un espace mémoire plus important et d'un plus grand nombre de CPU en fédérant les ressources de plusieurs machines.

Les tests de performances sur diverses structures de calcul intensif confirment

donc les attentes portées sur la parallélisation de la méthode intégrale et évoquées dans la partie 2.3.1. De plus, la portabilité et la scalabilité du logiciel ModeLitz sont confirmées.

Les limites du logiciel

Au vu des performances du logiciel, nous pouvons dire que les limites de la méthode intégrale en termes de temps de calcul (voir partie 2.2.6) sont levées grâce au calcul parallèle. En effet, sous réserve de ne pas recourir au *multithreading*, il suffit de disposer de suffisamment de cœurs de processeurs pour réduire d'autant le temps de calcul.

Par contre, l'emploi du parallélisme ne fait pas complètement disparaître le problème de l'espace mémoire nécessaire au stockage du système linéaire. En effet, celui-ci est toujours proportionnel au carré du nombre de degrés de liberté. Par exemple, un système comptant 182 000 degrés de liberté requiert un espace de environ 531 Go. Ainsi, sur ce point, l'apport du parallélisme est de permettre de morceler cet espace et de le répartir sur plusieurs machines. Grâce à cela, il est donc possible d'envisager des calculs importants à condition de disposer d'un espace mémoire total suffisant.

D'autre part, comme nous l'avons décrit dans la partie 2.3.2, chaque processus construit son propre maillage, ce qui permet d'assurer la forte parallélisation de l'étape de construction du système. Mais ceci présente un inconvénient : la taille occupée par le maillage est alors proportionnelle au nombre de processus. Or, nous avons constaté que le maillage généré par le logiciel peut être conséquent. Ainsi, une modélisation comptant environ 182 000 degrés de liberté et lancée sur un seul processus est basée sur un maillage occupant environ 3.35 Go. A grand nombre de processus, la taille totale occupée par tous les maillages entre donc en concurrence avec l'espace nécessaire au stockage du système linéaire ce qui limite les capacités du logiciel.

2.5 Conclusion et perspectives

Dans ce chapitre, nous avons présenté les travaux de développement du logiciel parallèle ModeLitz consacré à la modélisation électromagnétique d'inducteurs multibrins. Nous avons détaillé la méthode de calcul permettant de décrire la géométrie complexe de ces objets. Puis, nous avons exposé la méthode intégrale employée dans ce logiciel. Cette méthode est appropriée car elle permet une modélisation locale 3D de systèmes électromagnétiques à géométrie complexe sans requérir de maillage de l'air et autres parties isolantes, contrairement à la Méthode des Éléments Finis. Il est donc possible de simuler des systèmes dans lesquels les tailles des objets sont très variées et où les espaces entre conducteurs sont de faibles dimensions devant celles-ci. Ceci constitue le principal atout de cette méthode.

Nous avons également présenté les inconvénients de la méthode. D'une part, les temps de calculs pour la construction et la résolution du système linéaire sont importants. D'autre part, le stockage de ce système, qui est une matrice pleine et non symétrique, nécessite un espace mémoire conséquent. Ces deux points induisent de fortes limitations sur l'emploi de cette méthode par un logiciel séquentiel.

Nous avons donc exposé le principe de parallélisation de la construction et de la résolution du système linéaire employés dans ModeLitz ainsi que les avantages attendus par cette implémentation. Nous avons ensuite procédé à la validation du logiciel par comparaison avec le logiciel MIGEN. Enfin, nous avons réalisé des études de performances sur divers systèmes de calcul intensif et mis en évidence les principales limites de l'outil ModeLitz.

L'emploi du calcul parallèle est le facteur clé de la performance de l'outil de modélisation que nous avons développé. Il permet une avancée importante tant en ce qui concerne les temps de calcul que la taille des simulations. En effet, le logiciel MIGEN permet d'effectuer des calculs comptant jusqu'à environ 22 000 degrés de liberté. Un tel calcul a une durée de l'ordre de 100 heures. En raison de l'implémentation séquentielle de ce logiciel, des calculs plus conséquents sont difficilement envisageables en raison de l'augmentation de l'espace mémoire nécessaire (de l'ordre du carré du nombre de degrés de liberté) et du temps de calculs (variant comme l'ordre du cube du nombre de degrés de liberté). La simulation la plus volumineuse effectuée à ce jour avec ModeLitz compte environ 182 000 degrés de liberté, soit 8 fois les capacités standards offertes par MIGEN. Néanmoins, des limites demeurent en raisons des besoins importants en espace mémoire.

Pour palier ces limitations, plusieurs pistes complémentaires peuvent être envisagées. La réorganisation de l'implémentation de certaines parties du code est une première étape incontournable. Ainsi, nous avons identifié que nous pouvons réaliser une économie de mémoire en travaillant sur la méthode de construction des chemins des paquets et des brins. De plus, ce travail permettrait de réduire le temps de calcul nécessaire à la construction des chemins. Nous pensons également qu'il est possible d'obtenir un gain de mémoire en réorganisant le code des classes C++ liées au maillage.

Une autre voie intéressante serait d'organiser un partage du maillage entre les processus. Pour éviter que chaque processus construise son propre maillage, ce qui peut demander beaucoup d'espace mémoire, il pourrait être astucieux de répartir la construction et le stockage du maillage sur tout ou partie des processus, de manière analogue à l'approche adoptée pour le stockage de la matrice du système. Évidemment, cela demanderait d'apporter des modifications à l'algorithme de construction du système linéaire et pourrait entraîner une certaine dégradation des performances pour cette étape (non-quantifiable à ce jour).

D'autre part, le nombre de calculs de l'algorithme de construction du système linéaire que nous avons décrit varie comme le carré du nombre de degrés de liberté liés à l'inconnue $\vec{J}$, car nous calculons la loi de Biot et Savart sur tous les éléments du maillage. Nous pourrions réduire ce nombre en ne tenant compte, dans le calcul

de cette loi, que des éléments appartenant à un domaine restreint (une sphère, par exemple) centré sur chaque nœud $\vec{J}$. Une telle approche entraînerait inévitablement une dégradation des résultats mais celle-ci pourrait être contrôlée par un choix judicieux des dimensions de ce domaine (rayon de la sphère). Ainsi, il pourrait être envisageable d'obtenir une matrice creuse. Il faudrait alors utiliser un algorithme approprié capable de résoudre le système linéaire non symétrique résultant.

Nous pensons qu'une réflexion plus poussée sur l'organisation des processus et sur l'algorithme du pivot de Gauss pourrait permettre d'en améliorer les performances. Compte tenu de la topologie de la matrice, nous avons trouvé une procédure permettant de réduire le nombre de tests nécessaires à la recherche du pivot maximal pour une proportion non-négligeable des étapes du solveur.

Comme nous l'avons montré, ModeLitz est parallélisé grâce à la bibliothèque MPI. Nous envisageons sérieusement une implémentation hybride MPI-OpenMP [80]. En effet, OpenMP fonctionnant sur des systèmes à mémoire partagée, l'emploi de cette approche permettrait de réaliser d'importantes économies de mémoire. De plus, nous avons identifié que la programmation hybride nous permettrait d'accélérer certaines étapes, notamment la résolution, en réduisant le nombre de communications et en permettant un équilibrage de charge en calculs plus dynamique.

En l'état, ModeLitz permet la modélisation électromagnétique des inducteurs multibrins, mais n'inclut pas la possibilité de décrire une charge (pièce à chauffer). Il est essentiel d'implémenter cette possibilité afin d'affiner l'étude du comportement électromagnétique de ces inducteurs en tenant compte de leur environnement.

Enfin, des travaux sont encore à mener pour permettre la description d'inducteurs plus complexes. En effet, l'outil de description des chemins est fonctionnel tant que le chemin d'inducteur suit soit une courbe, soit une ligne droite. En revanche, il est inadapté si celui-ci est constitué d'une succession de parties rectilignes et courbes. Travailler sur ce point permettrait de faire un pas important vers la description de chemins d'inducteurs industriels.

Chapitre 3

Études du comportement électromagnétique d'inducteurs multibrins

Notre objectif est d'étudier l'influence de divers paramètres sur le comportement électromagnétique des inducteurs multibrins : longueur d'inducteur, pas et sens de torsadage et fréquence d'alimentation.

Dans l'article [81], nous nous sommes concentrés sur l'analyse de l'impact du pas de torsadage sur des inducteurs en hélice comptant 18 brins. Nous avons également comparé les comportements d'un câble en hélice (28 brins) et d'un câble à brins parallèles (37 brins). Ces études ont été réalisées sur de faibles longueurs de câble.

Dans ce chapitre, nous étudierons le comportement électromagnétique d'inducteurs à plusieurs niveaux de paquets. Pour ce faire, nous commencerons par nous intéresser au rôle de la longueur modélisée et du pas de torsadage sur le comportement électromagnétique d'un paquet de quatre brins. Puis, nous analyserons l'impact des paramètres de torsadage des brins (pas et sens) ainsi que le comportement en fréquence d'un câble composé de deux niveaux de paquets. Enfin, nous comparerons les performances énergétiques de trois inducteurs comptant 12 brins.

Sauf précision, les études suivantes seront menées sur des inducteurs constitués de brins en cuivre d'un rayon r_{brin} de 25 µm. Chaque brin est recouvert d'un isolant d'épaisseur 5 µm. De plus, le gradient du potentiel électrique le long des inducteurs est fixé à 0.5 V.m^{-1}. Enfin, le chemin suivi par le câble est rectiligne.

3.1 Étude d'un paquet de quatre brins

Avant d'entreprendre l'étude d'inducteurs à plusieurs niveaux de paquets, nous nous intéressons au comportement électromagnétique d'un simple paquet de brins.

Nous choisissons de modéliser un paquet de quatre brins agencés en hélices [1]. Le rayon $r_{câble}$ de ce câble est de 72 µm. La fréquence d'alimentation est fixée à

1. Cet arrangement correspond à un câble 1×4, selon la description donnée dans la partie 2.1.2.

300 kHz, ainsi l'épaisseur de peau est d'environ 120 µm (δ/r_{brin}=4.75). L'épaisseur de peau est donc grande devant la taille des brins.

3.1.1 Influence de la longueur modélisée

Nous nous intéressons ici à l'impact de la longueur d'inducteur. Nous étudions différentes longueurs de câble $l_{câble}$ entre 3 mm et 50 mm, soit un rapport maximal $l_{câble}/r_{câble}$ égal à 694. Pour pouvoir modéliser de telles longueurs d'inducteur, la taille des éléments du maillage est fixée à environ 6.5 fois le rayon d'un brin (voir partie 2.4.2). Le pas de torsadage choisi est de 3 mm.

La figure 3.1 montre la variation de l'inductance linéique et de la résistance de l'inducteur en fonction de sa longueur. Nous remarquons que la résistance évolue linéairement en fonction de la longueur. Le câble a donc un comportement essentiellement résistif, avec une résistance linéique de l'ordre de 2.23 Ω.m^{-1}.

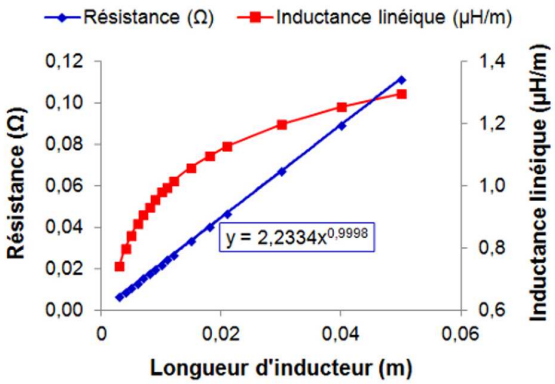


FIGURE 3.1 – Évolution de l'inductance linéique et de la résistance de l'inducteur en fonction de la longueur. F = 300 kHz.

Nous observons également une augmentation de l'inductance linéique. L'origine de cette variation est purement géométrique et est portée par la loi de Biot et Savart (2.8) (voir partie 2.2). En effet, les formes géométriques des objets influent sur leurs inductances propres ainsi que sur leurs inductances mutuelles. Par conséquent, l'augmentation de la longueur des brins tend à accroître ces inductances.

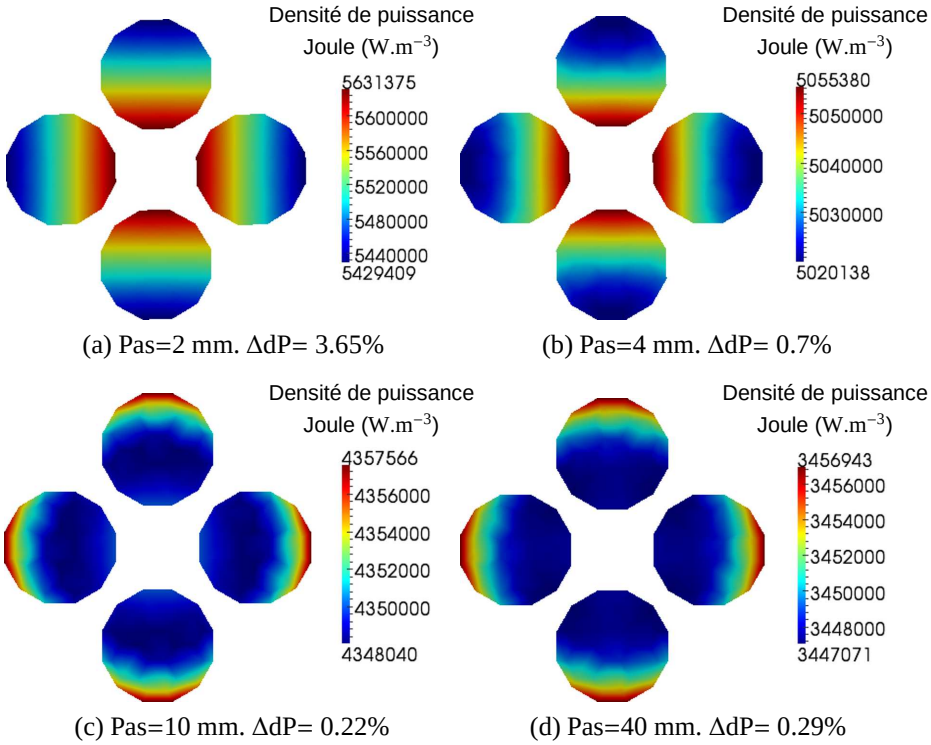
3.1.2 Influence du pas de torsadage

Nous étudions ici l'influence du pas de torsadage sur le comportement électromagnétique du câble 1×4 présenté précédemment. Nous testons des pas de torsadage compris entre 2 mm et 200 mm. A chaque fois, la longueur d'inducteur modélisée est égale au pas. La fréquence choisie est de 300 kHz.

La figure 3.2 présente la répartition de la densité de puissance Joule dans une coupe orthogonale au paquet pour différents pas. Nous définissons la valeur ΔdP

comme l'écart relatif (en %) entre les valeurs extrêmes de la densité de puissance Joule, dP_{max} et dP_{min}, selon la relation (3.1). Cette grandeur nous permet de déterminer si la répartition du courant entre les brins est homogène ou non.

$$\Delta dP = 200 * \frac{(dP_{max} - dP_{min})}{(dP_{max} + dP_{min})} \tag{3.1}$$



(a) Pas=2 mm. ΔdP= 3.65%

(b) Pas=4 mm. ΔdP= 0.7%

(c) Pas=10 mm. ΔdP= 0.22%

(d) Pas=40 mm. ΔdP= 0.29%

FIGURE 3.2 – Répartition de la puissance Joule dans une coupe d'un paquet de quatre brins pour différents pas de torsadage. F = 300 kHz.

Nous pouvons clairement constater que la variation du pas de torsadage a un impact direct sur la répartition de la puissance Joule dans les brins. Pour les pas inférieurs à 4 mm, la densité de puissance Joule, donc le courant, se concentre au centre du câble. Ceci s'explique par le fait que le courant tend à suivre le chemin le plus court possible et ce, malgré les effets de proximité entre brins qui tendent à écarter les courants circulant dans le même sens. Ainsi, à 300 kHz, le câble conserve un comportement résistif, similaire à celui observé à 10 kHz sur la figure 3.3. D'ailleurs, l'écart ΔdP à 300 kHz (3.65%) est très proche de la valeur de ΔdP à 10 kHz (3.99%).

A l'inverse, avec un pas supérieur à 8 mm, le comportement du paquet tend vers celui d'un câble à brins parallèles, le courant étant repoussé en périphérie du paquet. Ceci indique donc que, dans ce cas, les effets de proximité sont prépondérants.

La figure 3.4 montre l'évolution de l'écart ΔdP, calculé comme précédemment sur les coupes de l'inducteur, en fonction du pas de torsadage. Il apparaît que l'écart

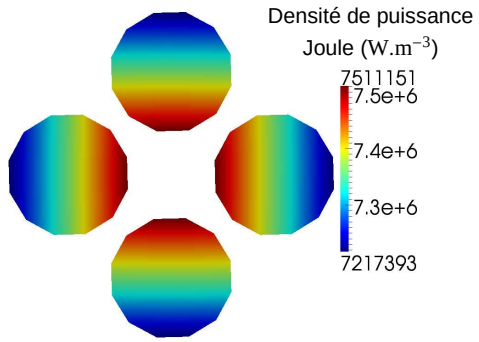


FIGURE 3.3 – Répartition de la puissance Joule dans une coupe d'un paquet de quatre brins torsadés selon un pas de 2 mm et pour une fréquence de 10 kHz. ΔdP= 3.99%

ΔdP minimum est de 0.14 % pour un pas de 7 mm. Ainsi, avec un pas de 7 mm environ, la densité de puissance est répartie de manière quasi-homogène dans une section de l'inducteur. Le courant est donc distribué uniformément entre les brins.

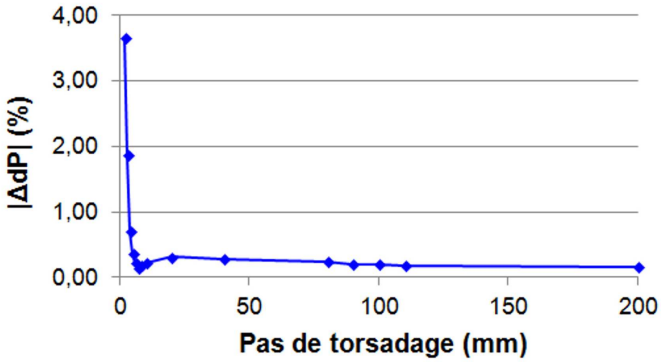


FIGURE 3.4 – Évolution de l'écart ΔdP en fonction du pas de torsadage.

La figure 3.5, montre l'influence du pas de torsadage sur la résistance et l'inductance linéiques du paquet. La figure 3.6, quant à elle, expose la variation du rapport entre la longueur des brins l_{brins} et la longueur de câble $l_{câble}$ en fonction du pas.

L'épaisseur de peau étant grande devant le rayon des brins, la diminution et la convergence de la résistance linéique vers une valeur d'environ 2.224 Ω.m^{-1} s'explique simplement par le fait que plus le pas de torsadage est grand, plus la longueur des brins s'approche de celle du câble, ainsi que le montre la figure 3.6. La hausse de l'inductance linéique, quant à elle, est liée à l'allongement de la longueur utilisée pour la modélisation de l'inducteur, comme évoqué en partie 3.1.1.

Nous retrouvons donc un comportement similaire à celui que nous avions présenté dans l'article [81]. Néanmoins, dans cette publication, compte tenu du nombre important de brins modélisés (entre 20 et 30 brins selon les configurations), nous avions simulé différents pas de torsadage tout en conservant une longueur d'inducteur constante. Dans la présente partie, nous avons adapté la longueur de l'inducteur à son pas de torsadage et avons ainsi pu étendre le domaine d'étude.

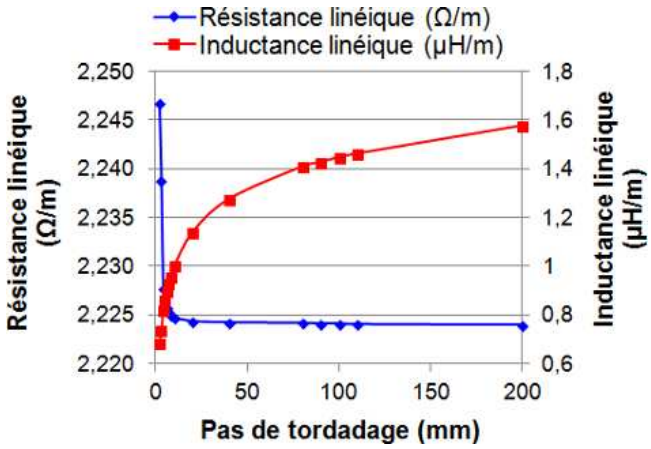


FIGURE 3.5 – Évolution de la résistance et de l'inductance linéiques d'un paquet de quatre brins en fonction du pas de torsadage.

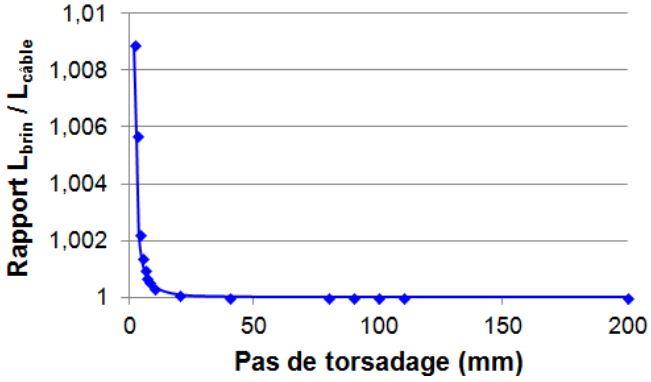


FIGURE 3.6 – Évolution du rapport $l_{brins}/l_{câble}$ en fonction du pas de torsadage.

Dans cette partie, nous avons constaté que le comportement électromagnétique d'un inducteur multibrins est très dépendant de sa géométrie. Nous avons notamment remarqué que l'allongement de l'inducteur conduit à l'augmentation de son inductance linéique. De plus, nous avons montré que le pas de torsadage influe grandement sur la répartition de la puissance dissipée dans l'inducteur. Plus le pas est petit, plus les effets de proximité sont masqués par le caractère résistif des brins. A l'inverse, l'allongement du pas tend à favoriser les effets de proximité.

3.2 Étude d'un câble à deux niveaux de paquets

3.2.1 Influence du sens de torsadage du niveau 0

Dans cette partie, nous nous intéressons à l'impact du sens de torsadage des paquets sur les pertes au sein d'un inducteur multibrins de type fil de Litz. Nous étudions un inducteur rectiligne 3×4 brins alimenté à une fréquence de 300 kHz

(δ/r_{brin}=4.75). La longueur de câble modélisée $l_{câble}$ est de 6 mm. La table 3.1 précise l'organisation du câble étudié. Le pas de torsadage des paquets de niveau 1 est de 4 mm. Ainsi, en moyenne, le rayon du câble $r_{câble}$ est de 162 µm, ce qui correspond à un rapport $r_{câble}/l_{câble}$ égal à environ 37. Chacune des simulations menées pour cette étude compte environ 94 000 degrés de libertés et a nécessité en moyenne 121 heures (temps *CPU*) de calculs sur 36 processus.

Indice du niveau	1	0
Nombre de paquets du niveau	3	4
Sens de torsadage	paramètres d'étude	
Pas de torsadage (m)	0.004	paramètre d'étude

TABLE 3.1 – Paramètres de construction de l'inducteur 3×4.

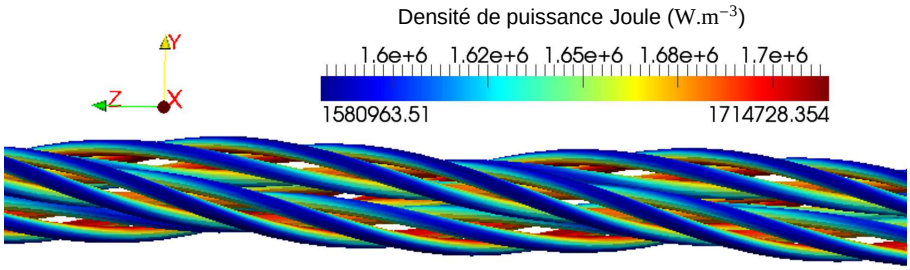
Nous faisons varier les sens de torsadage des niveaux. Dans la partie (2.1.2), nous avons décrit comment nous tenons compte du sens de torsadage : nous attribuons les valeurs +1 et −1 respectivement aux sens trigonométrique et antitrigonométrique. Nous désignons par le jeu de valeurs $(+1,-1)$ un câble à deux niveaux dont le niveau 1 est torsadé dans le sens +1 et le niveau 0 dans le sens −1.

Nous testons également trois pas de torsadage du niveau 0 : 2 mm, 3 mm et 4 mm. La figure 3.7 montre la répartition de la puissance Joule pour un pas de 3 mm et pour les deux configurations de torsadage. Le tableau 3.2 rapporte les valeurs de la résistance, de l'inductance, de la puissance Joule dissipée et du courant efficace dans l'inducteur en fonction des pas et des sens de torsadage du niveau 0.

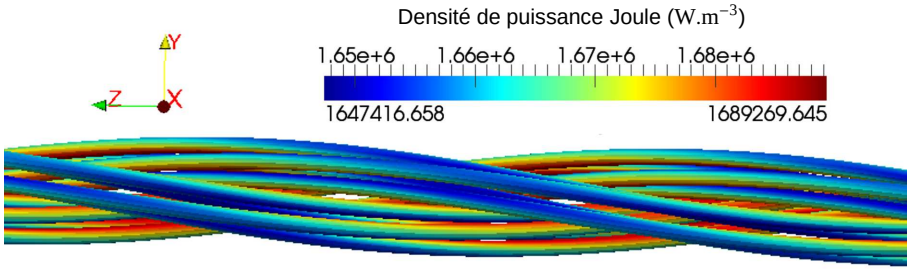
Nous constatons que, quel que soit le pas de torsadage du niveau 0, la configuration $(+1,-1)$ minimise les pertes Joule par rapport à la configuration $(+1,+1)$. La raison est que l'alternance des sens de torsadage conduit à des chemins de brins moins tourmentés et plus courts que dans le cas $(+1,+1)$, comme le montre la figure 3.7. Ainsi, premièrement, la résistance de la configuration $(+1,-1)$ est plus basse car, pour une même longueur d'inducteur, les brins sont plus courts, comme l'indique le tableau 3.3. Ceci explique la plus faible résistance de la configuration $(+1,-1)$, à l'instar de ce que nous avons constaté en partie 3.1.2. Deuxièmement, pour la même raison, l'inductance de la configuration $(+1,-1)$ est plus faible.

Pas (mm)	Sens de torsadage	Résistance (mΩ)	Inductance (nH)	Puissance Joule (mW)	Courant efficace (A)
2	$(+1,+1)$	4.5560	4.5655	2.1587e-1	2.1835e-1
	$(+1,-1)$	4.4966	4.5463	2.1574e-1	2.1917e-1
3	$(+1,+1)$	4.5611	4.4197	2.2685e-1	2.2335e-1
	$(+1,-1)$	4.4968	4.4006	2.2668e-1	2.2482e-1
4	$(+1,+1)$	4.5228	4.5622	2.1522e-1	2.1835e-1
	$(+1,-1)$	4.4870	4.5500	2.1513e-1	2.1917e-1

TABLE 3.2 – Comparatif de la résistance, de l'inductance, de la puissance dissipée et du courant efficace pour plusieurs paramètres de torsadage du niveau 0 du câble.



(a) Arrangement (+1, +1).



(b) Arrangement (+1, −1).

FIGURE 3.7 – Comparaison de la répartition de la puissance Joule entre les divers arrangements du câble 3×4 pour un pas de torsadage du niveau 0 égal à 3 mm.

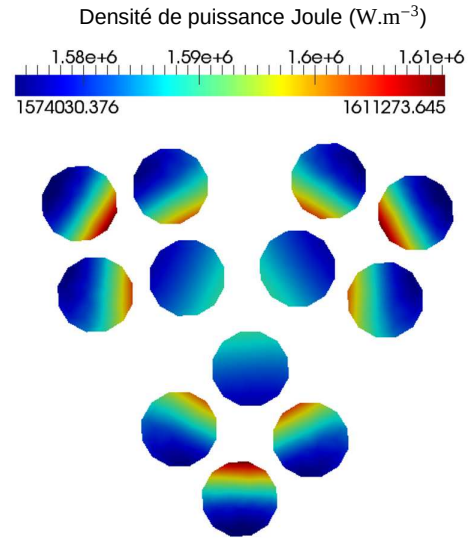
Configuration	Brin le plus court (mm)	Brin le plus long (mm)
(+1, +1)	6.17	6.168
(+1, −1)	6.066	6.068

TABLE 3.3 – Longueur des brins pour chaque configuration de torsadage de l'inducteur 3×4 (pas du niveau 0 = 3 mm).

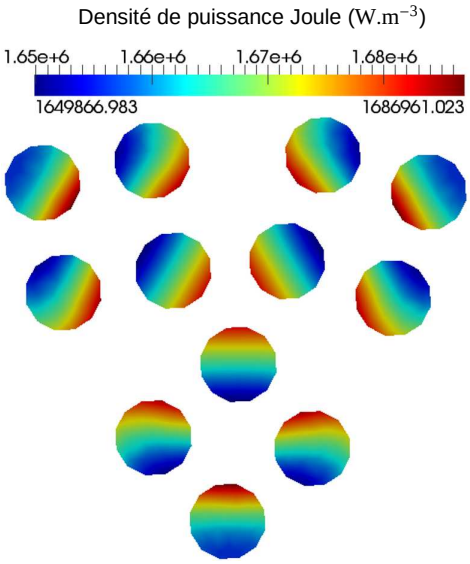
L'impact du torsadage est aussi visible localement sur la répartition de la densité de courant et, de fait, sur la répartition de la densité de puissance perdue dans l'inducteur. Les figures 3.8 (page 106) et 3.9 (page 107) montrent la répartition de la densité de puissance Joule sur une coupe orthogonale à l'inducteur et située à équidistance de ses extrémités, respectivement pour les arrangements (+1, −1) et (+1, +1).

A propos des représentations en coupe, nous pouvons faire deux remarques. Tout d'abord, les brins suivent des chemins qui passent tantôt aux environs du centre du câble, tantôt sur sa périphérie, aussi les répartitions présentées en coupe ne sont pas invariantes par translation le long des inducteurs. D'autre part, l'algorithme que nous avons décrit dans la partie 2.1.2 cherche à respecter les pas de torsadage indiqués par les paramètres de construction que nous choisissons. Pour ce faire, il procède à certaines optimisations qui peuvent avoir pour effet de modifier le rayon du câble. Ceci explique les différences de rayons visibles sur les figures 3.8 et 3.9.

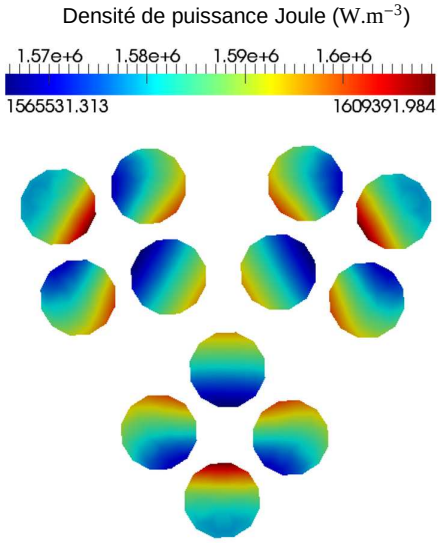
Le tableau 3.4 permet de comparer les valeurs de ΔdP en fonction du type d'arrangement et du pas de torsadage du niveau 0.



(a) Pas=2 mm. ΔdP=2.33 %. $r_{câble}$=156 µm.



(b) Pas=3 mm. ΔdP=2.22 %. $r_{câble}$=175 µm.

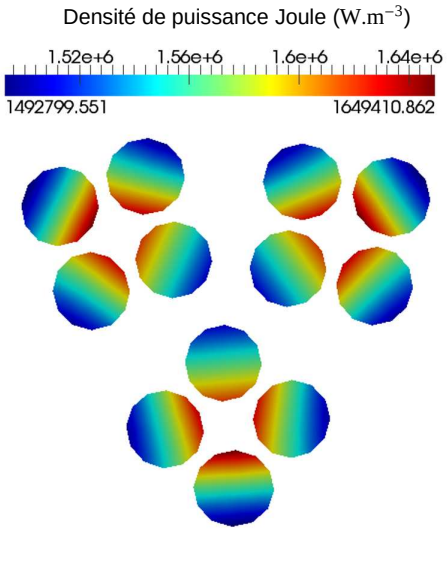


(c) Pas=4 mm. ΔdP=2.76 %. $r_{câble}$=156 µm.

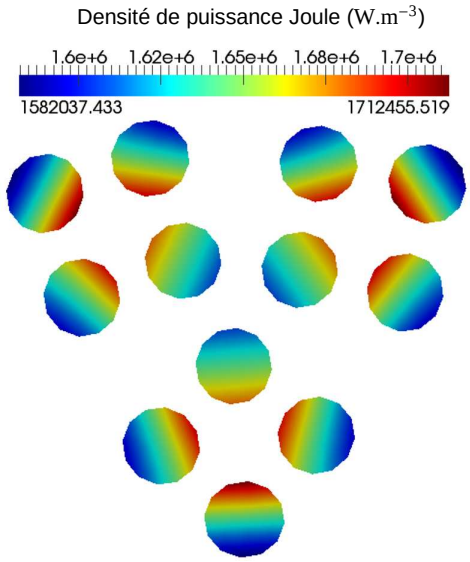
FIGURE 3.8 – Répartition de la puissance Joule dans une coupe de l'inducteur 3×4 pour l'arrangement (+1, −1) et selon différents pas de torsadage du niveau 0.

Pas de torsadage (mm)	Arrangement (+1, −1), ΔdP (%)	Arrangement (+1, +1), ΔdP (%)
2	2.33	9.97
3	2.22	7.92
4	2.76	5.91

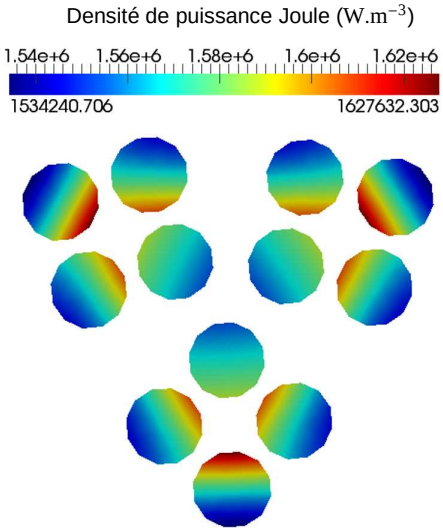
TABLE 3.4 – Comparaison des écarts ΔdP pour les différents agencements du câble.



(a) Pas=2 mm. ΔdP=9.97 %. $r_{câble}$=156 µm.



(b) Pas=3 mm. ΔdP=7.92 %. $r_{câble}$=175 µm.



(c) Pas=4 mm. ΔdP=5.91 %. $r_{câble}$=156 µm.

FIGURE 3.9 – Répartition de la puissance Joule dans une coupe de l'inducteur 3×4 pour l'arrangement (+1, +1) et selon différents pas de torsadage du niveau 0.

Nous constatons clairement que la densité de puissance est répartie de façon plus uniforme dans le cas de l'arrangement (+1, −1) avec un ΔdP variant entre 2.22 % et 2.76 %, contre un ΔdP compris entre 5.91 % et 9.97 % pour l'arrangement (+1, +1). De plus, pour ce dernier, nous notons que la densité de puissance tend à se concentrer au centre des paquets de quatre brins (voir figure 3.9). Par contre, dans chaque brin de l'arrangement (+1, −1), la densité de puissance maximale est située dans la zone du brin la plus proche du centre du câble (voir figure 3.8).

Cette étude semble indiquer que l'alternance des sens de torsadage tend à répartir plus uniformément la densité de courant dans l'inducteur.

3.2.2 Remarques sur le potentiel et la densité de courant

Nous nous intéressons ici aux profils des grandeurs locales. Nous choisissons d'étudier le câble 3×4 (4mm, 3mm) (+1, +1) alimenté à une fréquence de 300 kHz (δ/r_{brin}=4.75) et sous une tension de 0.5 V.m^{-1}.

La figure 3.10 montre l'orientation des vecteurs $\vec{J}$ au sein des brins. Comme nous l'avons constaté auparavant dans la partie 2.4.1, la direction de la densité de courant est tangente aux chemins des brins.

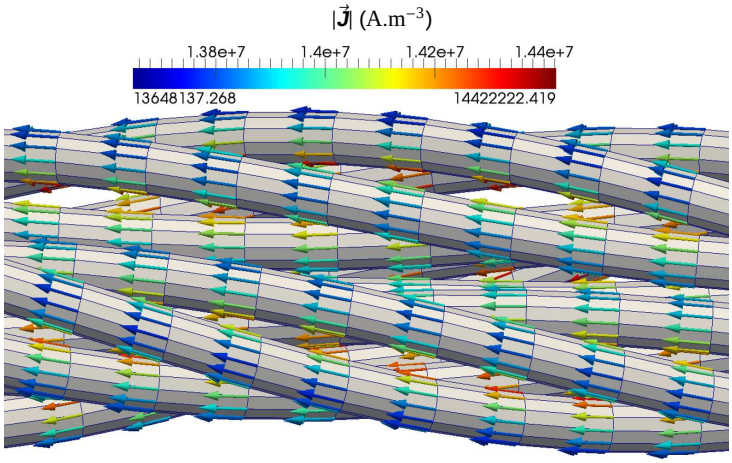


FIGURE 3.10 – Orientation de la densité de courant électrique dans l'inducteur 3×4.

La figure 3.11 présente la répartition dans l'inducteur du module du potentiel électrique. La figure 3.12 donne le profil du module du potentiel électrique évalué long d'un brin, au centre de ce brin. Nous constatons clairement que le potentiel varie linéairement le long du brin selon un gradient de 0.4987 V.m^{-1}, proche du gradient imposé de 0.5 V.m^{-1}.

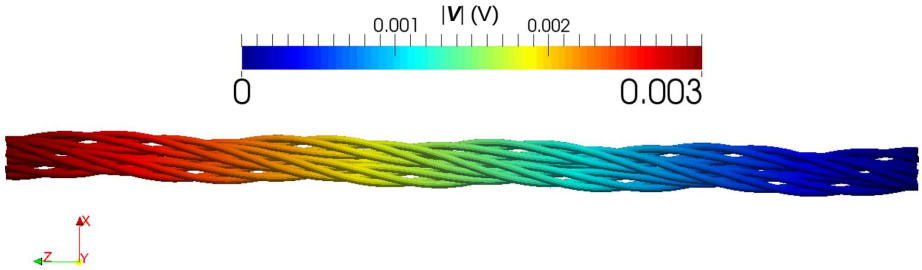


FIGURE 3.11 – Répartition du module du potentiel électrique (en module) dans l'inducteur 3×4.

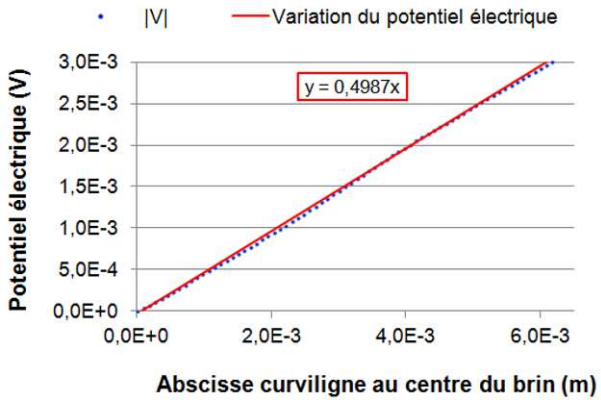


FIGURE 3.12 – Profil du potentiel électrique le long d'un brin.

3.2.3 Influence de la fréquence

Nous nous intéressons à présent au comportement du câble 3×4 (4mm, 2mm) $(+1,-1)$ en fonction de la fréquence d'alimentation. La figure 3.13 montre l'évolution de la résistance et de l'inductance de l'inducteur en fonction de la fréquence entre 50 kHz (δ/r_{brin}=11.6) et 3 Mhz (δ/r_{brin}=1.5). La figure 3.14 présente la variation du courant efficace total traversant l'inducteur en fonction de la fréquence.

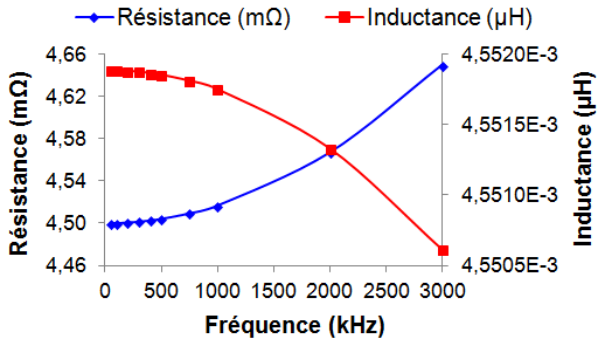


FIGURE 3.13 – Évolution de la résistance et de l'inductance l'inducteur en fonction de la fréquence d'alimentation.

Nous pouvons constater qu'entre 100 kHz et 3 MHz, la résistance totale du câble augmente d'environ 3.3 % et l'inductance diminue d'environ 0.03 %. De plus, le courant efficace traversant le câble diminue d'environ 94.5 %. Ces variations traduisent l'augmentation des effets de proximité avec la fréquence. L'impact de ces effets sur la distribution du courant dans les brins, et donc sur la distribution des pertes, est visible sur la figure 3.15 qui montre la répartition de la puissance Joule dissipée dans le câble pour les fréquences extrêmes considérées.

Pour une fréquence de 100 kHz, nous remarquons que le courant a tendance à circuler à l'intérieur du câble. De plus, la faible dispersion de la puissance Joule (ΔdP=2.66 %) montre que le courant est assez uniformément réparti entre les brins.

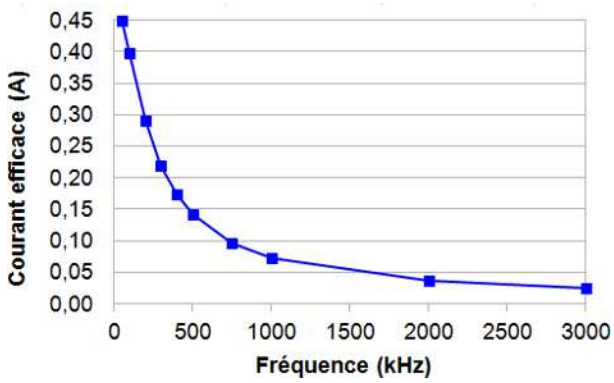
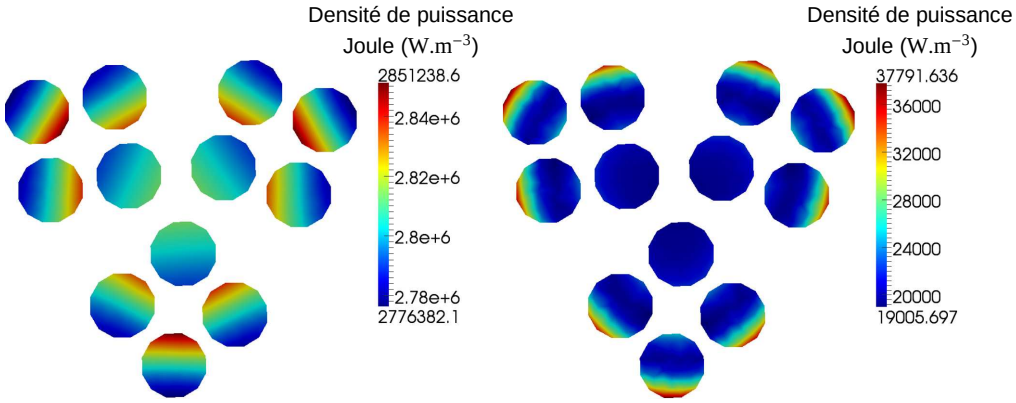


FIGURE 3.14 – Évolution du courant efficace dans l'inducteur en fonction de la fréquence d'alimentation.



(a) F=100 kHz. ΔdP=2.66 %. (b) F=3 Mhz. ΔdP=66.15 %.

FIGURE 3.15 – Répartition de la puissance Joule dans une coupe de l'inducteur 3×4 en fonction de la fréquence.

Le courant moyen par brin est de 3.32 mA. En revanche, à 3 MHz, le courant est repoussé en périphérie de l'inducteur par effet de proximité et est moins bien réparti (ΔdP=66.15 %). Dans ce cas, le courant moyen par brin est de 2.06 mA. Ainsi, le courant total traversant le câble diminue avec l'augmentation de la fréquence.

Dans cette partie, nous avons à nouveau montré le rôle majeur de la géométrie sur la répartition du courant et des pertes Joule dans les inducteurs multibrins. Ainsi, dans le cas d'un câble comptant deux niveaux de paquets, il semble que l'alternance des sens de torsadage permettent de réduire les pertes en répartissant plus uniformément le courant à travers l'inducteur. D'autre part, nous avons étudié l'impact de la fréquence et avons montré que l'augmentation consécutive des effets de proximité se traduit principalement par l'accroissement de la résistance du câble.

3.3 Études de diverses structures d'inducteurs

Nous analysons maintenant l'influence de la structure de l'inducteur. Pour ce faire, nous modélisons des portions de trois types de câbles rectilignes comptant chacun 12 brins : un câble 1×12, un câble 3×4 et un câble 4×3. Dans tous les cas, le pas de torsadage des brins est de 2 mm. Pour les câbles 3×4 et 4×3, le pas de torsadage des paquets de niveau 1 est de 4 mm. La longueur des portions de câbles modélisées est de 6 mm et la fréquence choisie est de 300 kHz (δ/r_{brin}=4.75).

Les grandeurs électriques globales pour chaque type de câble sont comparées dans le tableau 3.5.

Rappelons que la tension aux bornes est la même pour tous les câbles et correspond à 2.12 mV. Nous remarquons donc que, pour cette tension, le câble 4×3 est celui qui permet le passage d'un maximum de courant.

Le logiciel permet de connaitre la puissance Joule dissipée dans chaque brin. Le tableau 3.6 présente les valeurs minimales, maximales et moyennes des puissances Joules par brin pour les câbles 1×12, 3×4 (+1, −1) et 4×3 (+1, −1).

	Résistance (mΩ)	Inductance (nH)	Puissance Joule (mW)	Courant efficace (A)
Câble 1×12	4.6035	4.6939	2.0890e-1	2.1296e-1
Câble 3×4 (+1, +1)	4.5560	4.5655	2.1587e-1	2.1835e-1
Câble 3×4 (+1, −1)	4.4966	4.4006	2.1574e-1	2.1917e-1
Câble 4×3 (+1, +1)	4.5437	4.5095	2.1985e-1	2.2009e-1
Câble 4×3 (+1, −1)	4.5005	4.4980	2.1962e-1	2.2100e-1

TABLE 3.5 – Comparaison de la résistance, de l'inductance, de la puissance dissipée et du courant efficace pour différentes configurations de câbles à 12 brins.

	Câble 1×12	Câble 3×4	Câble 4×3
Puissance moyenne (W)	1.74079e-5	1.79782e-5	1.83017e-5
Puissance minimale (W)	1.72938e-5	1.79741e-5	1.82997e-5
Écart à la moyenne (%)	0.6576	0.0225	0.0106
Puissance maximale (W)	1.77250e-5	1.79823e-5	1.83050e-5
Écart à la moyenne (%)	2.4626	0.0454	0.0288

TABLE 3.6 – Comparaison des puissances Joule par brin pour les câbles 1×12, 3×4 (+1, −1) et 4×3 (+1, −1).

Dans chaque cas, nous constatons que les puissances minimales et maximales sont proches. Tous les brins dissipent environ la même puissance et sont donc approximativement parcourus par le même courant. Nous constatons que les écarts les plus importants sont obtenus avec le câble 1×12. Il apparaît clairement que trois brins dissipent plus de puissance que les autres, ainsi que le montre la figure 3.16.

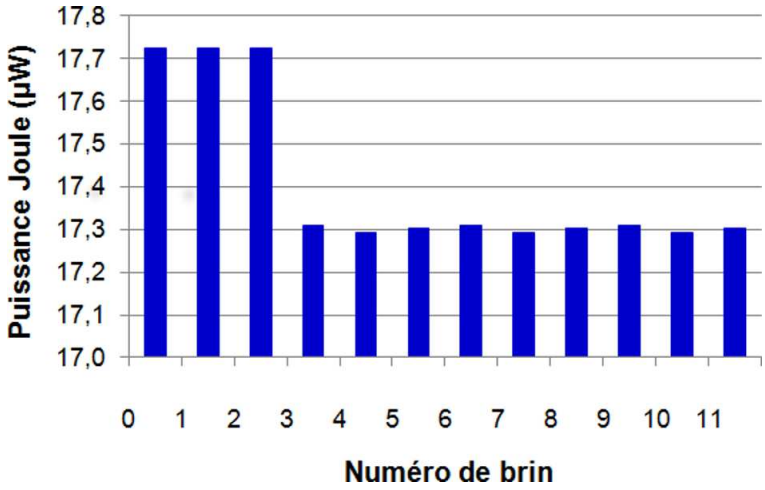
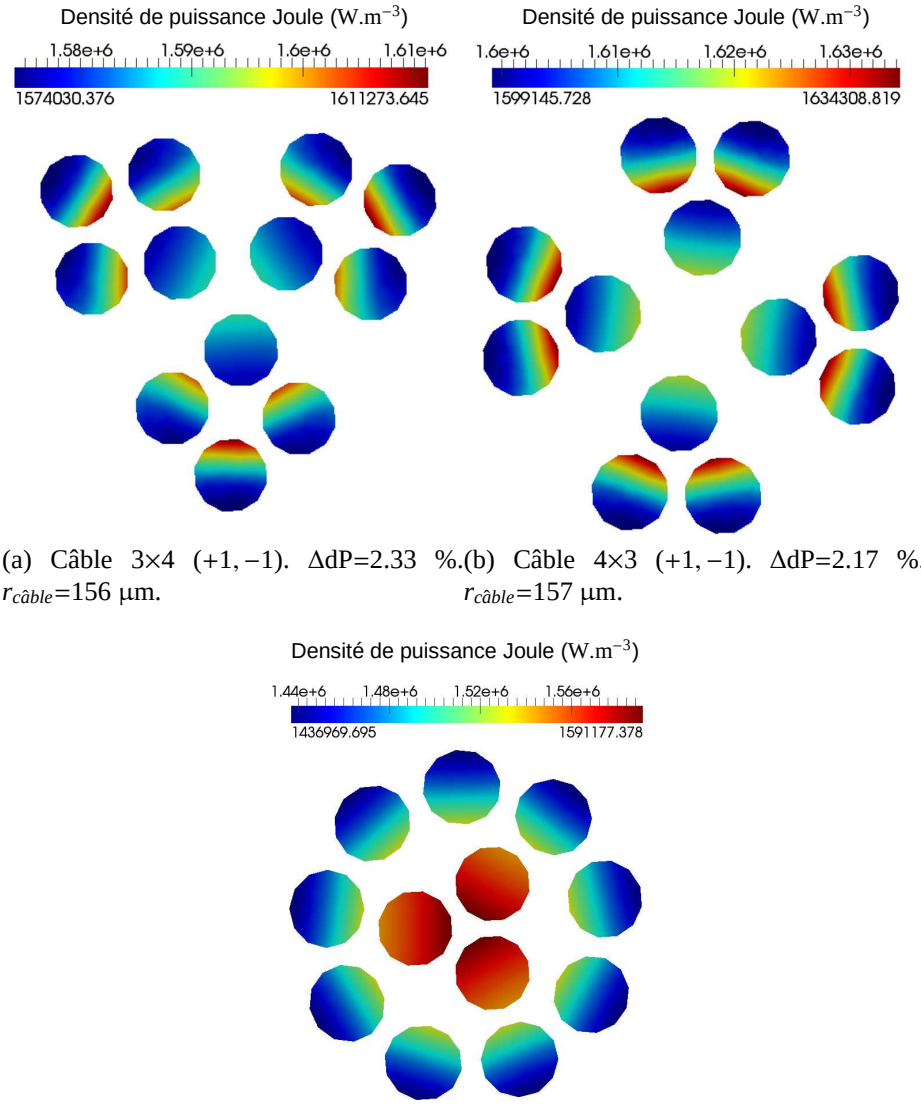


FIGURE 3.16 – Distribution de la puissance Joule entre les brins du câble 1×12.

La figure 3.17 montre la distribution de la densité de puissance Joule dans une coupe de chacun des câbles. Dans le cas du câble 1×12, se sont les trois brins centraux qui dissipent le plus de puissance. Nous remarquons que la répartition de la puissance Joule est plus homogène dans les câbles Litz (ΔdP compris entre 2.17 % et 2.33 %) que dans le câble en hélice (ΔdP=10.18 %).

A ce stade, nous n'avons pas assez d'éléments nous permettant de déterminer quel arrangement fournirait le meilleur rendement énergétique. Pour pouvoir avoir une meilleure idée de l'efficacité des ces arrangements, nous suggérons d'effectuer une étude fréquentielle. Il devrait ainsi être possible d'observer lequel d'entre eux est le moins sensible à la fréquence en regardant la variation de leurs résistances équivalentes ainsi que des écarts ΔdP pour chacun d'eux. Il serait également nécessaire d'analyser leur comportement en présence d'une charge.

Dans les configurations étudiées ici, nous avons remarqué que le courant total par brin varie peu d'un brin à l'autre. Il semble donc que soit vérifiée l'hypothèse du fil de Litz idéal – courants identiques dans tous les brins – évoquée dans la partie 1.2.1. Néanmoins, il convient d'insister sur le fait que, dans la présente étude, le nombre de brins modélisés est faible comparé aux centaines, voire milliers de brins, que compte un véritable inducteur multibrins. De plus, les fréquences étudiées génèrent des épaisseurs de peau très grandes devant le rayon des brins (δ/r_{brin}=4.75 à 300 kHz). Ces deux conditions sont favorables à la vérification de l'hypothèse du fil de Litz idéal. Pour autant, la question de sa validité pour la modélisation de câbles comptant un grand nombre de brins reste en suspens.



(a) Câble 3×4 $(+1,-1)$. ΔdP=2.33 %. $r_{câble}$=156 µm.

(b) Câble 4×3 $(+1,-1)$. ΔdP=2.17 %. $r_{câble}$=157 µm.

(c) Câble 1×12. ΔdP=10.18 %. $r_{câble}$=125 µm.

FIGURE 3.17 – Comparaison de la répartition de la puissance Joule au sein des différentes structures d'inducteurs (coupes orthogonales).

3.4 Remarque sur les limites du modèle volumique

Le tableau 3.7 indique les performances des calculs effectués pour modéliser différents arrangements. Ces calculs ont été menés sur le cluster du laboratoire. Les temps de calculs rapportés sont les temps *elapsed* (cf. partie 2.4.3).

Type de câble	Nombre de degrés de liberté	Temps de calcul (h)	Nombre de processus	Taille mémoire totale (Go)
1×12	111837	65	75	395
3×4	93552	120	36	217
4×3	110112	68	120	466
5×11	182105	254	132	975

TABLE 3.7 – Performances de divers calculs menés sur le cluster du laboratoire.

Les caractéristiques des câbles 1×12, 3×4 et 4×3 ont été données dans la partie précédente (3.3).

En ce qui concerne le câble 5×11, il s'agit d'un câble droit d'une longueur de 35 mm. Le tableau 3.8 présente sa structure. Les brins ont un rayon de 50 µm et sont entourés d'un isolant d'épaisseur 5 µm.

Indice de niveau	1	0
Nombre de paquets du niveau	5	11
Sens de torsadage	-1	1
Pas de torsadage	0.035	0.025

TABLE 3.8 – Configuration du câble 5x11.

La modélisation du câble 5×11 constitue la simulation la plus conséquente ayant été traitée par ModeLitz avec la méthode pérsentée au chapitre 2 [2].

Le cluster dispose de 1 To de mémoire RAM et de 160 cœurs virtuels (cf. partie 2.4.3). Avec ce système, nous constatons que le logiciel permet d'effectuer aisément des calculs comptant environ 110 000 degrés de liberté. En effet, ces calculs durent en moyenne entre deux et trois jours. La modélisation du câble 5×11, quant à elle, nécessite environ 10 jours de calculs et atteint presque les limites en mémoire de la machine.

Nous concluons donc que, en l'état actuel du développement et avec un système comparable au cluster du laboratoire, le logiciel permet de mener des simulations comptant jusqu'à environ 200 000 degrés de liberté. Ainsi, nous estimons qu'il est possible de modéliser un câble similaire au câble 5×11 avec, au maximum, une soixantaine de brins. Au-delà, la taille mémoire nécessaire atteint les limites de la machine et les temps de calculs deviennent beaucoup trop importants pour permettre d'effectuer des études systématiques ou des recherches de paramètres optimaux pour l'agencement des câbles.

2. La description complète de cette simulation est donnée dans l'annexe C.

3.5 Conclusion et perspectives

Dans ce chapitre, nous avons mis en évidence l'influence du pas de torsadage sur les effets de proximité dans un paquet de quatre brins. Nous avons montré que, pour une même fréquence d'alimentation, l'augmentation du pas favorise les effets de proximité. A l'inverse, la diminution du pas tend à les supprimer.

Par la suite, nous nous sommes penchés sur l'étude d'inducteurs à deux niveaux de paquets. Dans la partie 3.1.2, nous avons montré que l'alternance des sens de torsadage semble réduire les effets de proximité entre les brins, favorisant ainsi une répartition homogène du courant au sein du câble.

Nous avons remarqué que la direction du courant dans un brin est toujours tangente au chemin du brin et que le potentiel électrique varie linéairement le long de celui-ci. Nous avons également analysé la réponse en fréquence de l'inducteur 3×4 (4mm, 2mm) $(+1, 1)$ et avons montré que l'influence des effets de proximité croît avec celle-ci, ce qui se traduit par une augmentation de la résistance équivalente du câble.

Ensuite, nous avons comparé les comportements électromagnétiques de plusieurs inducteurs à 12 brins. Nous avons étudié l'influence de la structure interne de ces inducteurs sur la répartition de la puissance dissipée et avons montré qu'il est possible d'agir sur cette dernière en ajustant les paramètres de torsadage.

De toutes les études présentées dans ce chapitre, nous relevons deux points importants.

Premièrement, nous avons montré le rôle majeur de la géométrie interne des inducteurs multibrins sur l'importance et la répartition des pertes Joule en leur sein. Il semble ainsi possible de réduire les effets de proximité et donc de réduire les pertes en jouant sur le pas et le sens de torsadage des brins. Des études plus poussées sur des inducteurs à plus grands nombres de brins et comptant davantage de niveaux de paquets devraient permettre de vérifier si ces conclusions sont extensibles à l'échelle des paquets de niveaux supérieurs.

Deuxièmement, dans le cas d'un inducteur à 12 brins et à deux niveaux de paquets, nous avons montré que l'augmentation de la fréquence accroît les effets de proximité, ce qui se traduit essentiellement par une augmentation de la résistance équivalente du câble. Aussi, nous sommes enclins à penser que l'inducteur multibrin optimal devrait présenter l'augmentation la plus faible possible de sa résistance en fonction de la fréquence, mais cela demande à être confirmé par des études plus approfondies.

Nous nous sommes également aperçu que, pour des inducteurs comptant un faible nombre de brins, l'hypothèse qui suppose un courant égal dans tous les brins semble valable. Toutefois, la validité de cette hypothèse pour la modélisation d'inducteurs comptant des nombres de brins plus importants et dans des configurations plus complexes demande à être vérifiée.

Grâce au modèle volumique que nous avons décrit en partie 2.2, nous avons pu mener certaines études locales du comportement électromagnétique d'inducteurs multibrins. Ces études ont concerné des inducteurs à faibles nombres de brins, de faibles longueurs et comptant au maximum deux niveaux de paquets. Il est donc nécessaire de prolonger ces études en s'intéressant à des inducteurs plus complexes. Néanmoins, les limites actuelles imposées par le modèle volumique restreignent fortement ces objectifs. Ces contraintes devraient pouvoir être repoussées grâce aux améliorations du modèle évoquées en partie 2.5.

Au cours des travaux présentés ici, nous avons systématiquement étudié le comportement d'un inducteur seul, qui plus est, suivant un chemin rectiligne. Pour aller plus loin, il est nécessaire de considérer le comportement d'un inducteur suivant un chemin plus complexe et en présence d'une charge.

Chapitre 4

Modèle filiforme

La méthode intégrale que nous avons présentée dans le chapitre 2 est particulièrement adaptée aux calculs 3D des effets inductifs apparaissant localement dans les inducteurs multibrins. Toutefois, cette modélisation *volumique* présente deux inconvénients majeurs : elle est coûteuse en temps de calcul et en espace mémoire.

Nous présentons dans cette partie, une seconde approche pour la modélisation d'inducteurs multibrins, que nous dénommons modélisation *filiforme*, également implémentée dans ModeLitz.

4.1 Présentation du modèle

4.1.1 Motivations

La méthode intégrale présentée au chapitre 2 permet de calculer précisément les phénomènes inductifs en jeu à l'échelle locale, internes aux brins. Néanmoins, comme nous l'avons montré, cette modélisation volumique ne permet pas d'effectuer des calculs sur des inducteurs de grandes longueurs et/ou comptant beaucoup de brins (câble limité à une soixantaine de brins, cf. partie 3.4). Le modèle filiforme que nous décrivons ici se base sur deux constations tirées des études présentées dans la partie 3.2.2.

Tout d'abord, nous avons observé que, si sur une section d'inducteur la densité de courant n'est pas toujours identique dans tous les brins, en revanche, elle varie peu dans la section orthogonale d'un brin. De plus, dans chaque brin, la direction du vecteur densité de courant est toujours colinéaire à la tangente au chemin de ce brin. Nous faisons donc l'hypothèse que la densité de courant est uniforme dans la section d'un brin et s'exprime selon la formule (4.1).

$$\boldsymbol{J} = \frac{\boldsymbol{I}}{\mathcal{S}} \tag{4.1}$$

avec :

— $\boldsymbol{I}$, le courant total parcourant le brin ;
— $\mathcal{S}$, la section du brin.

De plus, nous avons montré que la variation du potentiel électrique est linéaire le long d'un brin. Aussi, sur une section située à l'abscisse curviligne s le long du chemin d'un brin, le potentiel électrique V peut s'exprimer selon l'équation (4.2).

$$V = V_0 + \int_0^s \frac{\Delta V}{L} \mathrm{d}l = V_0 + \frac{\Delta V}{L} \int_0^s \mathrm{d}l \tag{4.2}$$

avec :
— V_0, le potentiel électrique imposé à l'extrémité d'abscisse $s = 0$ du brin ;
— ΔV, la différence de potentiel électrique à laquelle est soumis le brin ;
— L, la longueur totale du brin ;
— $\mathrm{d}l$, l'élément unitaire curviligne le long du chemin du brin.

De ce fait, le potentiel électrique n'est plus une inconnue. Il est donc possible d'établir un modèle dans lequel l'inconnue est le courant total $\boldsymbol{I}$ dans chaque brin. Le nombre de degrés de liberté est alors drastiquement réduit puisque qu'il est seulement égal aux nombre de brins que compte l'inducteur. La modélisation d'inducteurs comptant plusieurs milliers de brins et de grandes longueurs est alors possible car la contrainte imposée par le stockage du système linéaire est complètement levée.

4.1.2 Equation de base

Le modèle filiforme que nous décrivons dans ce chapitre est dérivé de la méthode intégrale décrite au chapitre 2. Un modèle très similaire est exposé dans l'article [22], où il est appliqué à une géométrie relativement simple dans laquelle le chemin des brins est calculable analytiquement.

Dans le présent modèle, les brins ne sont plus des objets volumiques mais filiformes. Ils sont donc représentés par des fils en 3D et dont les chemins sont calculés par la méthode présentée dans la partie 2.1.4.

En tout point d'un brin i, la densité de courant électrique $\vec{\boldsymbol{J}}_i$ vérifie la loi d'Ohm locale en régime harmonique (4.3).

$$\vec{\boldsymbol{J}}_i = -\sigma \overrightarrow{\nabla} \boldsymbol{V}_i - \boldsymbol{i}\omega\sigma\vec{\boldsymbol{A}}_i \tag{4.3}$$

avec :
— $\boldsymbol{V}_i$, le potentiel scalaire électrique ;
— $\vec{\boldsymbol{A}}_i$, le potentiel vecteur magnétique créé par tous les fils au point considéré.

$\vec{\boldsymbol{A}}_i$ est calculé par la loi de Biot et Savart (4.4).

$$\vec{\boldsymbol{A}}_i = \frac{\mu_0}{4\pi} \sum_{j=1}^{N_{brins}} \int_{C_j} \frac{\vec{\boldsymbol{J}}_j \mathcal{S}_j}{r} \mathrm{d}l_j \tag{4.4}$$

avec :

— N_{brins}, le nombre total de brins que compte l'inducteur ;
— C_j, le chemin suivi par le brin j dans l'inducteur ;
— $\boldsymbol{J}_j$, la densité de courant du brin j ;
— $\mathcal{S}_j = \pi r_j^2$, la section du brin j, de rayon r_j ;
— r, la distance entre le point où est calculé $\vec{\boldsymbol{A}}_i$ et un point du chemin C_j ;
— $\mathrm{d}l_j$ l'élément de longueur curviligne le long du chemin du brin j.

En introduisant (4.4) dans (4.3) et en intégrant cette dernière le long du chemin C_i du brin i, nous obtenons pour l'inconnue $\vec{\boldsymbol{J}}_i$ l'équation (4.5).

$$\int_{C_i} \vec{\boldsymbol{J}}_i \mathrm{d}\vec{l}_i = -\sigma \int_{C_i} \overrightarrow{\nabla} \boldsymbol{V}_i \mathrm{d}\vec{l}_i - \boldsymbol{i}\omega\sigma \frac{\mu_0}{4\pi} \int_{C_i} \sum_{j=1}^{N_{brins}} \int_{C_j} \frac{\vec{\boldsymbol{J}}_j \mathcal{S}_j}{r} \mathrm{d}l_j \mathrm{d}\vec{l}_i \tag{4.5}$$

Avec l'hypothèse de l'uniformité de la densité de courant dans une section orthogonale du brin, $\vec{\boldsymbol{J}}_i$ s'exprime selon la relation (4.6).

$$\vec{\boldsymbol{J}}_i = \frac{\boldsymbol{I}_i}{\mathcal{S}_i} \vec{t}_i \tag{4.6}$$

avec :

— $\boldsymbol{I}_i$, le courant total traversant le brin i ;
— $\mathcal{S}_i$, la section du brin ;
— $\vec{t}_i$, le vecteur normé normal à la section $\mathcal{S}_i$, et donc tangent au chemin C_i.

Les vecteurs $\vec{t}_i$ et $\mathrm{d}\vec{l}_i$ étant colinéaires en tout point du brin i et le module de $\vec{\boldsymbol{J}}_i$ étant constant, nous pouvons écrire :

$$\int_{C_i} \vec{\boldsymbol{J}}_i \mathrm{d}\vec{l}_i = \frac{\boldsymbol{I}_i}{\mathcal{S}_i} L_i \tag{4.7}$$

où L_i est la longueur totale du brin i.

Pour la même raison, nous écrivons :

$$\int_{C_j} \frac{\vec{\boldsymbol{J}}_j \mathcal{S}_j}{r} \mathrm{d}l_j = \boldsymbol{I}_j \int_{C_j} \frac{1}{r} \mathrm{d}\vec{l}_j \tag{4.8}$$

Comme le potentiel électrique varie linéairement le long du brin, le gradient $\overrightarrow{\nabla} \boldsymbol{V}_i$ est également colinéaire à $\mathrm{d}\vec{l}_i$ et nous obtenons :

$$\int_{C_i} \overrightarrow{\nabla} \boldsymbol{V}_i \mathrm{d}\vec{l}_i = \Delta V \tag{4.9}$$

Par conséquent, compte tenu des relations (4.7), (4.8) et (4.9) ainsi que de la commutativité de l'intégrale sur C_i et de la somme sur les brins, nous pouvons transformer (4.5) en (4.10).

$$\frac{\boldsymbol{I}_i}{\mathcal{S}_i} L_i + \boldsymbol{i}\omega\sigma\frac{\mu_0}{4\pi}\sum_{j=1}^{N_{brins}}\left[\boldsymbol{I}_j \int_{C_i}\int_{C_j}\frac{1}{r}\mathrm{d}\vec{l}_j\mathrm{d}\vec{l}_i\right] = -\sigma\Delta V \tag{4.10}$$

L'équation (4.10) permet de construire le système linéaire de la méthode, de la forme $\boldsymbol{M}\cdot\boldsymbol{X} = \boldsymbol{K}$, où $\boldsymbol{M}$ est la matrice du système, $\boldsymbol{X}$ le vecteur des inconnues complexes $\boldsymbol{I}$ et $\boldsymbol{K}$ le second membre. Ce système compte donc N_{brins} degrés de liberté : les N_{brins} courants électriques de chaque brin. Le terme diagonal $\boldsymbol{M}_{ii}$ de la matrice contient la somme de la résistance ohmique du brin i et de son auto-inductance. Les termes $\boldsymbol{M}_{ij}$ constituent les inductances mutuelles entre les brins i et j. Le système linéaire obtenu est donc symétrique.

4.1.3 Discrétisation

Dans cette méthode, les brins sont maillés avec des éléments linéïques de Lagrange d'ordre 2, soient des éléments à trois nœuds. Les coordonnées de chaque point d'un élément peuvent être exprimées par la relation (4.11) grâce aux polynômes de Lagrange (4.12). Dans ces équations, u représente la coordonnée curviligne sur l'élément de linéïque de référence, prise sur l'intervalle $[-1;1]$.

$$x(u) = \sum_{n=1}^{3}\alpha_n(u)\,x_n \qquad y(u) = \sum_{n=1}^{3}\alpha_n(u)\,y_n \qquad z(u) = \sum_{n=1}^{3}\alpha_n(u)\,z_n \tag{4.11}$$

$$\alpha_1(u) = \frac{1}{2}u(1-u) \qquad \alpha_2(u) = 1-u^2 \qquad \alpha_3(u) = \frac{1}{2}u(1+u) \tag{4.12}$$

L'essentiel des calculs nécessaires à l'assemblage du système linéaire consiste dans l'évaluation des termes d'inductance mutuelle et d'auto-inductance. Ces termes ont une nature purement géométrique et sont exprimés, dans l'équation (4.10), par la double intégrale (4.13).

$$\int_{C_i}\int_{C_j}\frac{1}{r}\mathrm{d}\vec{l}_j\mathrm{d}\vec{l}_i \tag{4.13}$$

Le calcul de cette expression est réalisé grâce à la méthode de Gauss par intégration sur les éléments linéïques. Ainsi, (4.13) se discrétise suivant (4.14) :

$$\begin{aligned}\int_{C_i}\int_{C_j}\frac{1}{r}\mathrm{d}\vec{l}_j\mathrm{d}\vec{l}_i &= \sum_{e_i=1}^{NBE_i}\left[\int_{e_i}\sum_{e_j=1}^{NBE_j}\left[\int_{e_i j}\frac{1}{r}\mathrm{d}\vec{l}_j\right]\mathrm{d}\vec{l}_i\right] \\ &= \sum_{e_i=1}^{NBE_i}\left[\sum_{e_j=1}^{NBE_j}\left[\int_{e_i}\int_{e_j}\frac{1}{r}\mathrm{d}\vec{l}_j\mathrm{d}\vec{l}_i\right]\right]\end{aligned} \tag{4.14}$$

avec :
- NBE_i, le nombre d'éléments du brin i ;
- NBE_j, le nombre d'éléments du brin j ;
- e_i, l'élément e_i du brin i ;
- e_j, l'élément e_j du brin j.

Nous posons NPI_{e_i}, respectivement NPI_{e_j}, le nombre de points d'intégration de Gauss sur l'élément e_i, respectivement sur l'élément e_j.

Le long d'un élément e_j, l'élément de longueur $\mathrm{d}\vec{l}_j$ s'exprime selon (4.15).

$$\mathrm{d}\vec{l}_j = \vec{T}_j du \tag{4.15}$$

avec

$$\vec{T}_j(u) = \begin{bmatrix} \sum_{n=1}^{3} \frac{\partial \alpha_n}{\partial u}(u)\, x_n \\ \sum_{n=1}^{3} \frac{\partial \alpha_n}{\partial u}(u)\, y_n \\ \sum_{n=1}^{3} \frac{\partial \alpha_n}{\partial u}(u)\, z_n \end{bmatrix} \tag{4.16}$$

L'intégrale sur l'élément e_j peut être discrétisée selon l'expression (4.17).

$$\int_{e_j} \frac{1}{r} \mathrm{d}\vec{l}_j = \sum_{k_{e_j}=1}^{NPI_{e_j}} \frac{1}{r(u_{k_{e_j}})} \vec{T}_j(u_{k_{e_j}}) w_{k_{e_j}} \tag{4.17}$$

avec :
- k_{e_j} le point de Gauss k_{e_j} de élément e_j ;
- $w_{k_{e_j}}$ le poids d'intégration associé à ce point ;
- $r(u_{k_{e_j}})$ la distance entre un point du chemin C_i, de coordonnées (x, y, z), et le point k_{e_j}, de coordonnées $(x(u_{k_{e_j}}), y(u_{k_{e_j}}), z(u_{k_{e_j}}))$, et calculée selon (4.18).

$$r(u_{k_{e_j}}) = \sqrt{(x - x(u_{k_{e_j}}))^2 + (y - y(u_{k_{e_j}}))^2 + (z - z(u_{k_{e_j}}))^2} \tag{4.18}$$

Calcul des termes d'inductance mutuelle

Nous détaillons ici la discrétisation de la double intégrale (4.14) entre deux brins distincts i et j. Ceci constitue donc le calcul du terme d'inductance mutuelle M_{ij}.

La discrétisation de l'intégrale sur un élément e_i sur le brin i s'effectue de manière analogue à celle de l'intégrale sur l'élément e_j. Ainsi, en adaptant les relations (4.15) et (4.16) à l'intégration sur l'élément e_i, nous pouvons écrire :

$$\int_{e_i} \int_{e_j} \frac{1}{r} \mathrm{d}\vec{l}_j \mathrm{d}\vec{l}_i = \sum_{k_{e_i}=1}^{NPI_{e_i}} \int_{e_j} \frac{1}{r(u_{k_{e_i}})} \mathrm{d}\vec{l}_j \vec{T}_i(u_{k_{e_i}}) w_{k_{e_i}} \tag{4.19}$$

avec :

— k_{e_i}, le point de Gauss k de élément e_i ;
— $w_{k_{e_i}}$, le poids d'intégration associé à ce point ;
— $r(u_{k_{e_i}})$, la distance entre le point k_{e_j}, de coordonnées $(x(u_{k_{e_i}}), y(u_{k_{e_i}}), z(u_{k_{e_i}}))$, et un point du chemin C_j, de coordonnées (x, y, z), calculée par la relation (4.20).

$$r(u_{k_{e_i}}) = \sqrt{(x(u_{k_{e_i}}) - x)^2 + (y(u_{k_{e_i}}) - y)^2 + (z(u_{k_{e_i}}) - z)^2} \tag{4.20}$$

Enfin, en regroupant les relations (4.17) et (4.19), l'expression (4.14) s'écrit :

$$\int_{C_i}\int_{C_j} \frac{1}{r} \mathrm{d}\vec{l}_j \mathrm{d}\vec{l}_i = \sum_{e_i=1}^{NBE_i} \left[\sum_{e_j=1}^{NBE_j} \left[\sum_{k_{e_i}=1}^{NPI_{e_i}} \sum_{k_{e_j}=1}^{NPI_{e_j}} \frac{1}{r(u_{k_{e_j}}, u_{k_{e_i}})} \vec{T}_i(u_{k_{e_i}}) \cdot \vec{T}_j(u_{k_{e_j}}) w_{k_{e_i}} w_{k_{e_j}} \right] \right] \tag{4.21}$$

avec $r(u_{k_{e_i}}, u_{k_{e_j}})$ la distance entre les points d'intégration k_{e_i} et k_{e_j}, calculée selon (4.22).

$$r(u_{k_{e_i}}, u_{k_{e_j}}) = \sqrt{(x(u_{k_{e_i}}) - x(u_{k_{e_j}}))^2 + (y(u_{k_{e_i}}) - y(u_{k_{e_j}}))^2 + (z(u_{k_{e_i}}) - z(u_{k_{e_j}}))^2} \tag{4.22}$$

Calcul des termes d'auto-inductance

Appliquée au calcul des termes d'auto-inductance M_{ii}, la discrétisation dans la section précédente (4.1.3) pose problème. En effet, la double intégrale (4.21) est effectuée sur deux brins filiformes distincts i et j. Par contre, le calcul du terme M_{ii} requiert le calcul de la double intégrale sur le même brin i. Or, l'intégration réalisée selon la méthode ci-dessus diverge.

Pour résoudre ce problème, nous construisons temporairement un second brin i, non plus selon l'approximation filiforme, mais selon le modèle volumique. Ce brin est donc construit et maillé de la même façon que nous l'avons décrit dans la partie 2.2.3. Le maillage $\vec{J}$ obtenu est utilisé pour réaliser une partie de l'intégration. Ainsi, l'intégrale est calculée sur un volume fini et le problème de divergence ne se pose plus.

Nous distinguons ici le brin i_f, le brin i d'origine, traité selon l'approche filiforme, et le brin i_v, le brin i volumique utilisé pour l'intégration. Ce dernier joue le rôle du brin j dans l'équation (4.21) qui se réécrit alors selon (4.23).

$$\int_{C_{i_f}}\int_{C_{i_v}} \frac{1}{r} \mathrm{d}\vec{l}_{i_v} \mathrm{d}\vec{l}_{i_f} = \sum_{e_{i_f}=1}^{NBE_{i_f}} \sum_{e_{i_v}=1}^{NBE_{i_v}} \left[\sum_{k_{e_{i_f}}=1}^{NPI_{e_{i_f}}} \sum_{k_{e_{i_v}}=1}^{NPI_{e_{i_v}}} \left[\frac{1}{r(u_{k_{e_{i_v}}}, u_{k_{e_{i_f}}})} \vec{T}_{i_f}(u_{k_{e_{i_f}}}) \cdot \vec{T}_{i_v}(u_{k_{e_{i_v}}}) w_{k_{e_{i_f}}} w_{k_{e_{i_v}}} \right] \right] \tag{4.23}$$

où les indices f et v désignent respectivement les termes intervenant dans l'intégrale sur le brin i_f et i_v. Les termes $\vec{T}_{i_v}(u_{k_{e_{iv}}})$ sont calculés par interpolation sur les éléments volumiques du maillage $\vec{J}$.

Pour résumer, le système linéaire est donc construit grâce à l'équation (4.10). Le calcul de la double intégrale (4.13) est réalisé de deux manières : grâce à l'équation (4.21) pour les brins $i \neq j$ et à l'équation (4.23) pour $i = j$.

4.1.4 Implémentation dans ModeLitz

Comme nous l'avons précisé, le principal avantage d'un tel modèle réside dans la réduction significative du nombre de degrés de liberté du système, ce qui rend possible la modélisation d'inducteurs comptant plusieurs milliers de brins et de grandes longueurs. En revanche, la méthode de calcul des termes du système est similaire aux calculs réalisés dans le cadre de l'approche volumique et nécessite un temps de calcul conséquent. Néanmoins, le modèle filiforme implémenté dans ModeLitz profite largement du travail de parallélisation effectué pour l'implémentation du modèle volumique. Ainsi, tant la construction que la résolution du système sont parallélisées.

La répartition du système linéaire est effectuée simplement en attribuant à chaque processus un certain nombre d'équations à calculer ce qui revient à répartir les brins entre les processus. Il n'est donc pas possible d'utiliser un nombre de processus supérieur au nombre de brins. Ceci constitue une limite de l'implémentation actuelle car, lorsque le nombre d'éléments par brins est important, le nombre de calculs devient conséquent et les performances du logiciel sont bridées par cette contrainte.

4.2 Validation du modèle

4.2.1 Calculs à basse fréquence

Afin de tester la validité du modèle filiforme, nous réalisons deux modélisations d'un même inducteur multibrins, l'une par le modèle volumique et l'autre grâce au modèle filiforme. Nous comparons ensuite les résultats obtenus.

Nous choisissons de simuler une portion de câble 5×11 droit et d'une longueur de 35 mm. Les brins en cuivre ont un rayon de 50 µm. La structure de l'inducteur est donnée par le tableau 4.1. La différence de potentiel aux bornes de l'inducteur est de 0.35 V et la fréquence choisie est de 10 kHz. Dans ces conditions, l'épaisseur de peau électromagnétique est de 650 µm, largement supérieure au rayon d'un brin (δ/r_{brin}=13). Le modèle filiforme étant basé sur l'hypothèse d'une densité de courant uniforme dans la section d'un brin, ce choix de fréquence est approprié. Ceci est confirmé par les résultats retournés par le modèle volumique. La figure 4.1 montre en effet que le module de la densité de courant dans un brin varie très peu.

Indice de niveau	1	0
Nombre de paquets du niveau	5	11
Sens de torsadage	-1	1
Pas de torsadage	0.035	0.025

TABLE 4.1 – Configuration du câble 5×11.

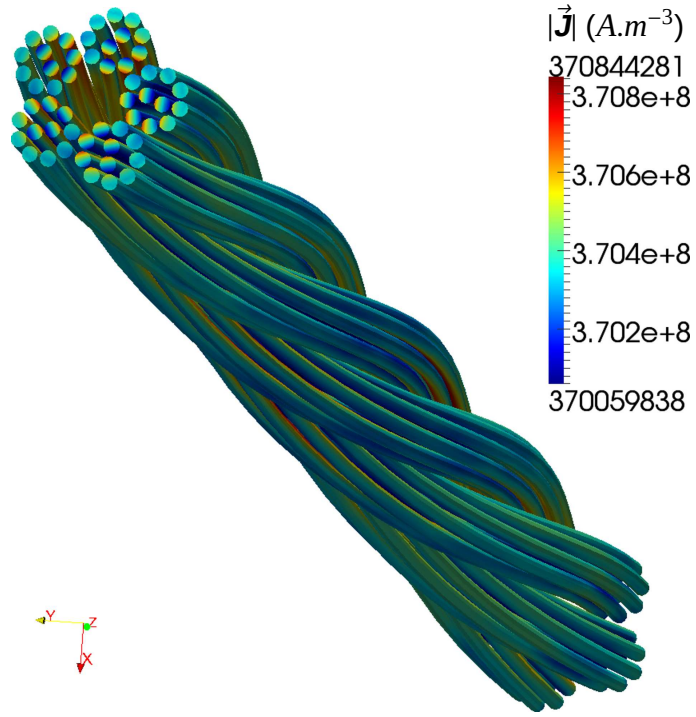


FIGURE 4.1 – Répartition de la densité de courant (en module) au sein de l'inducteur 5×11.

Le tableau 4.2 montre les valeurs globales pour l'inducteur obtenues par chacun des deux modèles, ainsi que leurs écarts relatifs. Nous constatons une bonne correspondance entre les modèles, notamment pour le calcul de l'inductance.

L'étude menée avec le modèle volumique correspond à la modélisation la plus conséquente que nous ayons menée. Elle comptait 36 960 nœuds $\vec{\boldsymbol{J}}_3$, 18 480 nœuds $\vec{\boldsymbol{J}}_2$, 3 960 nœuds $\vec{\boldsymbol{J}}_1$ et 30 305 nœuds $\boldsymbol{V}$ soit un total de 182 105 degrés de liberté. Ce calcul a été parallélisé sur 132 processus sur le cluster du laboratoire et a requis 975 Go de mémoire, dont 445 Go pour le stockage du maillage et 515 Go pour le stockage du système linéaire. Une durée d'exécution de 10 jours et 14 heures a été nécessaire pour réaliser cette simulation [1], soit un temps de calcul cumulé d'environ 3 ans, 10 mois et 16 jours.

1. Ceci est le temps *elapsed time* du processus maître (voir partie 2.4.3). Le temps *CPU*, lui, se monte à 8 jours et 10 heures ($7.2741804e+5$ s). L'importante différence constatée est due à l'emploi du *multithreading*. Néanmoins, il est impossible de prédire si la parallélisation sur un nombre de processus plus petit permettant d'éviter cet usage réduit ou non le temps de calcul (voir partie 2.4.3).

	Modèle volumique	Modèle filiforme	Ecarts relatifs (%)
Courant, partie réelle (A)	9.4586e+1	9.3493e+1	1.16
Courant, partie imaginaire (A)	-1.2002e+2	-1.2417e+2	3.40
Courant efficace (A)	1.08e+2	1.10e+2	1.83
Résistance (Ω)	1.4176e-3	1.3543e-3	4.57
Inductance (H)	2.8630e-8	2.8629e-8	0.003
Puissance Joule (W)	1.6542e+1	1.6361e+1	1.10

TABLE 4.2 – Comparaison des grandeurs globales obtenues pour la modélisation d'un inducteur 5×11 par les modèles volumique et filiforme.

Avec le modèle filiforme, chaque brin est maillé avec 15 éléments linéiques d'ordre 2, soit un total de 825 éléments. Dans ce modèle, la taille du système linéaire est considérablement réduite car l'inconnue est le courant par brin, donc le nombre de degrés de liberté est égal au nombre de brins, soit 55. L'étape de construction du système linéaire est donc celle qui nécessite le plus de temps par rapport au temps total de calcul. Dans cette étude, le temps de calcul se monte à 11.27 s (*elapsed time*) avec une parallélisation sur 55 processus. Le temps *CPU* est très proche : 10.97 s. La construction du système a duré 10.83 s, en temps *CPU*, et la résolution 0.01 s. La taille mémoire maximale nécessaire au calcul était de 3.45 Go. En résumé, le modèle filiforme a permis de réaliser la modélisation 66 277 fois plus rapidement que le modèle volumique en utilisant un espace mémoire 282 fois moins important.

4.2.2 Estimation du domaine de validité fréquenciel

Nous comparons à présent les modèles volumique et filiforme dans le domaine fréquentiel. Nous étudions le comportement d'un inducteur 3×6 dont la description est donnée par le tableau 4.3. Sa longueur est de 5 cm et son rayon de 355 µm. Les brins sont en cuivre et d'un rayon de 50 µm.

Indice de niveau	1	0
Nombre de paquets du niveau	3	6
Sens de torsadage	-1	1
Pas de torsadage	0.035	0.025

TABLE 4.3 – Configuration du câble 3×6.

Nous modélisons le comportement de ce câble à l'aide des deux modèles sur une plage de fréquence comprise entre 1 kHz (δ/r_{brin}=41.1) et 3 MHz (δ/r_{brin}=1.5). Le modèle volumique étant un modèle local nous le prenons comme référence. Pour chaque fréquence, nous calculons l'écart entre les résultats obtenus par le modèle filiforme et ceux délivrés par le modèle volumique. La figure 4.2 montre l'évolution

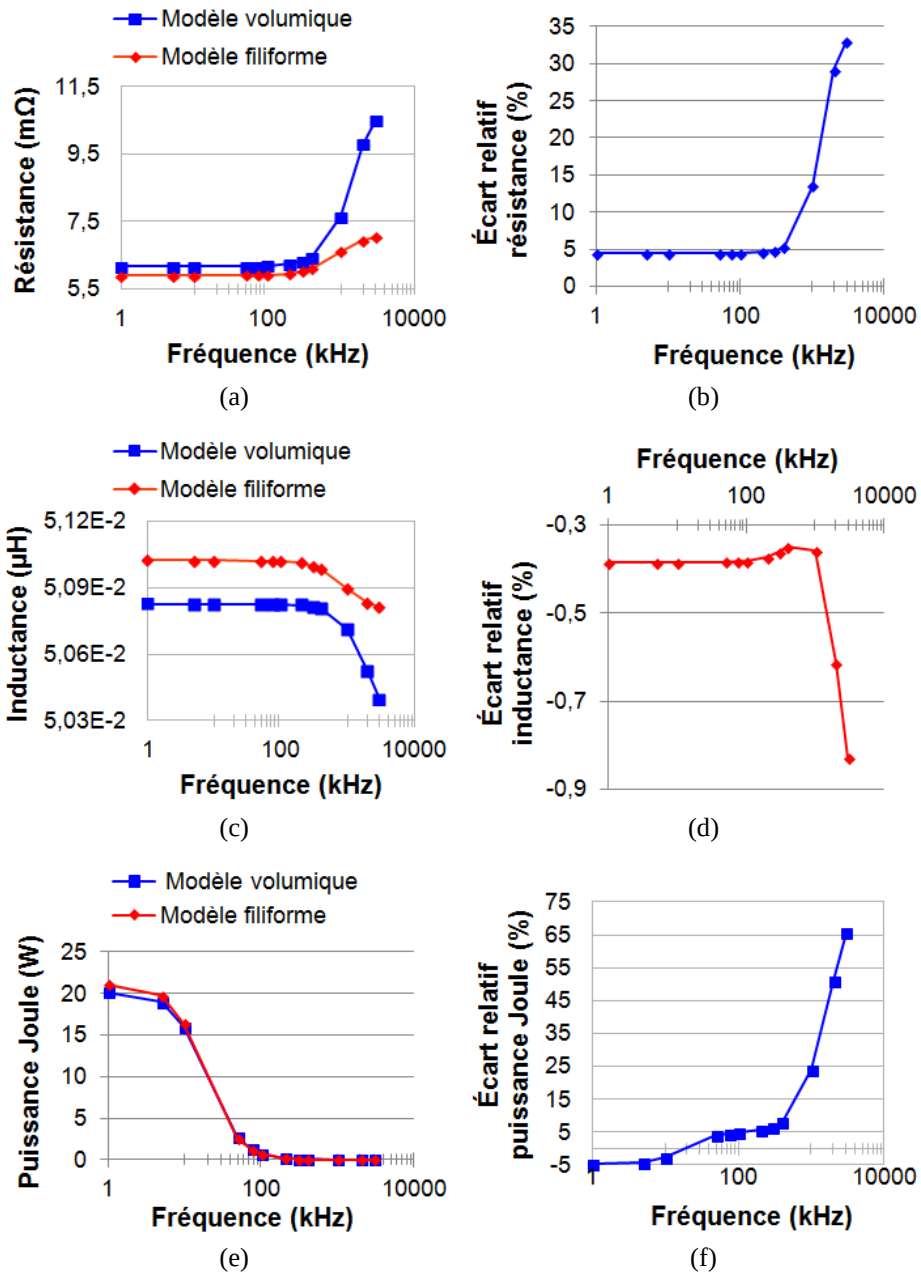


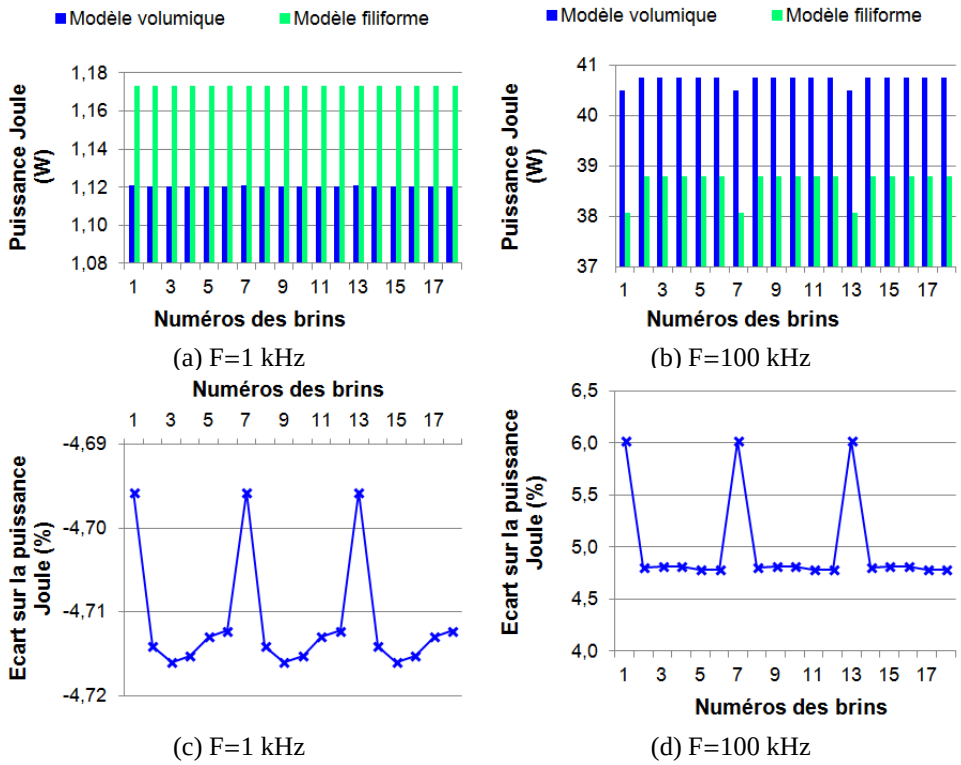
FIGURE 4.2 – Évolution des valeurs et des écarts calculés par le modèle filiforme par rapport au modèle volumique et en fonction de la fréquence, pour la résistance, l'inductance et la puissance Joule de l'inducteur 3×6.

des valeurs et des écarts sur la résistance, l'inductance de l'inducteur et la puissance Joule dissipée.

Nous constatons que le modèle filiforme a toujours tendance à sous-évaluer la résistance (4.2b) et à sur-évaluer l'inductance du câble (4.2d). Nous remarquons également que le modèle filiforme sur-évalue la puissance Joule dissipée dans l'in-

ducteur, quand la fréquence est inférieure à 10 kHz. Au-delà, les deux modèles divergent (4.2f). L'évolution des écarts sur la résistance et l'inductance sont également clairement visibles sur les figures 4.2a et 4.2c. Comme nous l'avons vu dans le chapitre précédent, l'augmentation de la fréquence favorise les effets de proximité. Cela se traduit principalement par l'accroissement de la résistance de l'inducteur (cf. partie 3.2.3) selon un profil similaire aux variations mesurées dans l'article [82] sur des câbles comptant plusieurs dizaines de brins. Il en résulte une diminution du courant et de la puissance dissipée dans le câble.

Pour ces trois grandeurs, nous constatons que les écarts entre les modèles sont relativement faibles et restent constants pour une fréquence inférieure à 500 kHz : ils sont inférieurs à 5 % pour la résistance et la puissance Joule et inférieurs à 0.5 % pour l'inductance. Au-dessus de 500 kHz, plus la fréquence augmente, plus les écarts deviennent importants. Ceci se vérifie particulièrement pour la résistance équivalente de l'inducteur et, par conséquent, pour la puissance Joule dissipée en son sein. Nous pouvons donc conclure que le modèle filiforme est adapté pour modéliser cet inducteur pour des fréquences d'alimentation inférieures à 500 kHz.



(a) F=1 kHz (b) F=100 kHz

(c) F=1 kHz (d) F=100 kHz

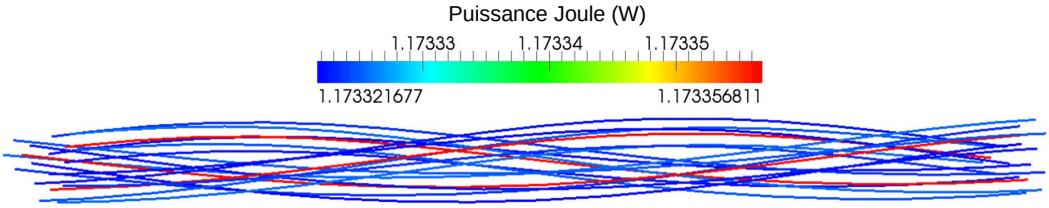
FIGURE 4.3 – Valeurs de la puissance Joule dissipée dans chaque brins et écarts entre les modèles à 1 kHz (a), (c) et à 100 kHz (b), (d).

La figure 4.3 montre la puissance Joule dissipée dans chaque brin et calculée par les deux modèles, ainsi que les écarts correspondants pour les fréquences 1 kHz et 100 kHz. Nous observons à nouveau que le modèle filiforme sur-évalue la puis-

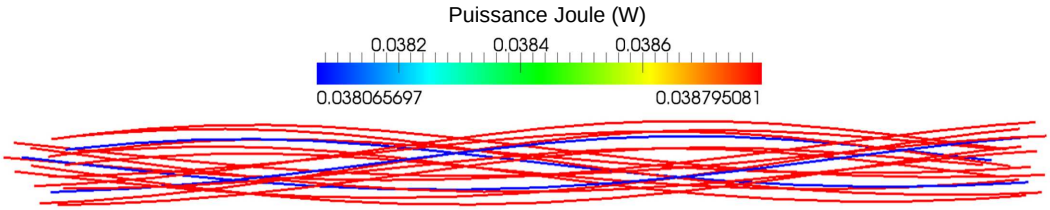
sance dissipée quand la fréquence est inférieure à 10 kHz, et la sous-évalue au-delà. L'écart entre les modèles est d'environ 4.7 % à 1 kHz et de 5 % à 100 kHz.

Chaque paquet de six brins est constitué d'un brin central entouré de cinq brins. Les brins 1, 7 et 13 sont situés chacun au centre d'un paquet. A 1 kHz, nous constatons que la puissance dissipée est quasiment la même dans tous les brins, alors qu'à 100 kHz les brins centraux dissipent moins de puissance que les autres. Ceci est visible sur la figure 4.4 qui présente la géométrie du câble ainsi que la puissance Joule dissipée dans chaque brin à 1 kHz et 100 kHz calculée par le modèle filiforme. Précisons que, sur les figures 3D résultant du post-traitement du modèle filiforme, les grandeurs affichées sont des grandeurs globales données pour chaque brin. Par conséquent, ces figures ne sont pas interprétables d'un point de vue local.

A 1 kHz (figure 4.4a), les brins dissipant le maximum de puissance sont au nombre de trois et sont situés chacun au centre d'un paquet. Ils dissipent une puissance de 1.173357 W contre une puissance à peine plus faible (1.173322 W) pour les autres brins. L'écart ΔP vaut 0.003 %. Le câble a donc un comportement résistif et le courant est réparti équitablement entre les brins. En revanche, à 100 kHz (figure 4.4b), les brins aux centres des paquets dissipent une puissance de 0.03807 W contre 0.0388 W pour les autres brins. L'écart ΔP vaut 2 %. Le courant circule donc davantage dans les brins périphériques des paquets en raison de l'augmentation des effets de proximité consécutive à la montée en fréquence.



(a) F=1 kHz. ΔP=0.003 %.

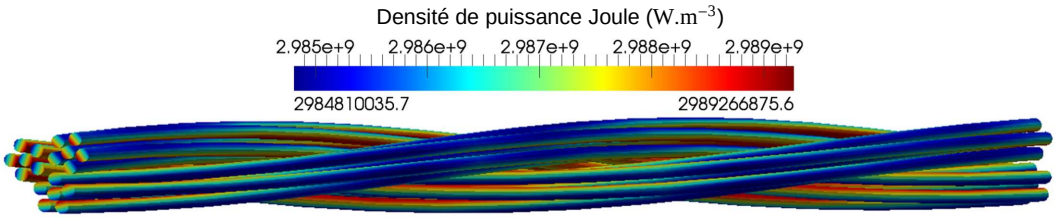


(b) F=100 kHz. ΔP=2 %.

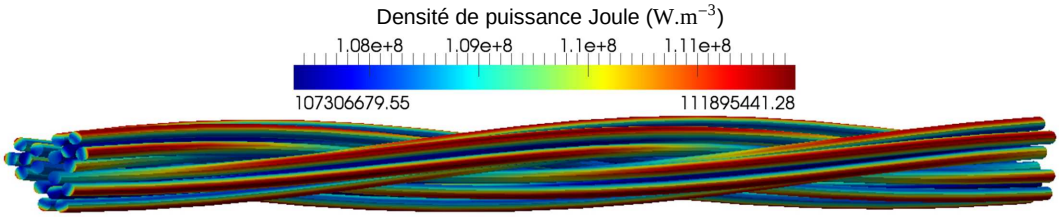
FIGURE 4.4 – Répartition de la puissance Joule entre les brins de l'inducteur 3×6 à 1 kHz et à 100 kHz (modèle filiforme).

Ces observations sont confirmées par la figure 4.5 qui présente la répartition locale (calculée par le modèle volumique) de la densité de puissance Joule dissipée dans l'inducteur. A 1 kHz (figure 4.5a), nous vérifions effectivement que, dans chaque brin, la puissance est concentrée dans une zone proche du centre du câble, ce qui est le signe d'un comportement résistif. De plus, le courant est réparti unifor-

mément entre les brins (ΔdP=0.15 %). A 100 kHz (figure 4.5b), nous remarquons bien que la puissance est davantage dissipée en périphérique du câble que dans les brins centraux (ΔdP=4.19 %).



(a) F=1 kHz. ΔdP=0.15 %.



(b) F=100 kHz. ΔdP=4.19 %.

FIGURE 4.5 – Répartition locale de la densité de puissance Joule dans l'inducteur 3×6 à 1 kHz et 100 kHz (modèle volumique).

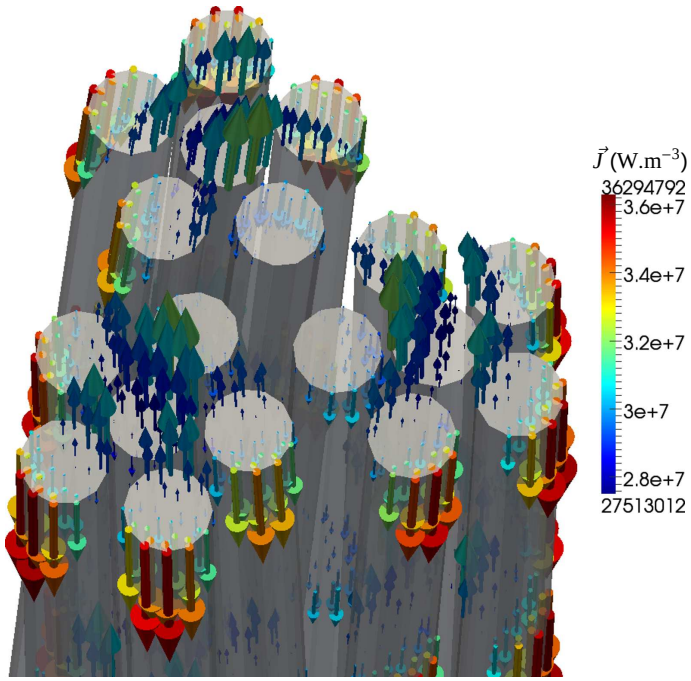


FIGURE 4.6 – Distribution de la partie réelle de la densité de courant dans l'inducteur 3×6 à 400 kHz (modèle volumique).

La figure 4.6 montre la distribution de la densité de courant dans l'inducteur 3×6 à 400 kHz calculée par le modèle volumique. Nous constatons que le courant ne

circule pas dans le même sens dans tous les brins. En raison des effets de proximité, certains brins, en l'occurrence les brins situés aux centres des paquets, sont traversés par un courant circulant dans le sens inverse du courant global.

Nous présentons maintenant les performances des deux approches pour la modélisation de l'inducteur 3×6 pour une seule fréquence et parallélisées sur le même nombre de processus (10 processus). Avec le modèle volumique, le calcul a nécessité 26 heures et 40 minutes (96 018 s) pour la construction du système linéaire et 56 heures et 16 minutes (202 585 s) pour la résolution. Avec le modèle filiforme, la modélisation complète a duré 69.91 s. Le modèle filiforme a donc été 4 271 fois plus rapide que le modèle volumique.

Nous avons vérifié la validité du modèle filiforme en le comparant au modèle volumique. Nous avons constaté que, dans le cas d'un inducteur 3×6, les deux modèles donnent des résultats proches tant que la fréquence est inférieure à 500 kHz. Ces études ont également mis en évidence l'intérêt de l'approche filiforme. Elle génère en effet des gains considérables en temps de calcul et en espace mémoire, ce qui permet d'envisager des modélisations d'inducteurs plus conséquentes.

4.3 Modélisation d'un inducteur à grand nombre de brins

Dans cette partie, nous présentons de manière plus concrète les potentialités du modèle filiforme en comparant les comportements électromagnétiques d'un inducteur en fil de Litz et d'un inducteur plein de même section. Nous supposons que le modèle filiforme est toujours valable pour la modélisation d'un inducteur comptant un grand nombre de brins et pour des fréquences inférieures à 500 kHz, d'après les conclusions de l'étude menée dans la partie 4.2.2.

L'inducteur multibrins compte 1512 brins selon un agencement 6×14×18. La structure de l'inducteur est donnée par le tableau 4.4. Les brins sont en cuivre et de rayon 50 µm. Le rayon total du câble $r_{câble}$ est de 3.64 mm. L'inducteur plein en cuivre est de même rayon.

Indice de niveau	2	1	0
Nombre de paquets du niveau	6	14	18
Sens de torsadage	1	-1	1
Pas de torsadage	0.071	0.035	0.025

TABLE 4.4 – Configuration de l'inducteur multibrins 6×14×18.

Les deux inducteurs suivent un chemin en hélice de pas 1 cm, de rayon interne 5 cm et de hauteur 5 cm. Les câbles mesurent environ 1.57 m linéaires. La différence de potentiel aux bornes des inducteurs est fixée à 11.1 V.

Nous comparons les deux inducteurs sur un domaine de fréquence compris entre 20 kHz (δ/r_{brin}=9.2, $\delta/r_{câble}$=0.13) et 400 kHz (δ/r_{brin}=2, $\delta/r_{câble}$=0.03). L'épaisseur de peau est donc grande devant le rayon des brins. L'inducteur multibrins est modélisé grâce au modèle filiforme. En revanche, l'épaisseur de peau est faible par rapport au rayon de l'inducteur plein. Celui-ci est simulé par le moyen de MIGEN. Ce logiciel permet en effet la modélisation d'objets en condition de faible épaisseur de peau : ceux-ci ne sont maillés qu'en surface et l'épaisseur de peau est prise en compte par une loi de décroissance exponentielle de la densité de courant. La figure 4.7 montre l'évolution, en fonction de la fréquence, de la résistance et de l'inductance pour chacun des deux inducteurs.

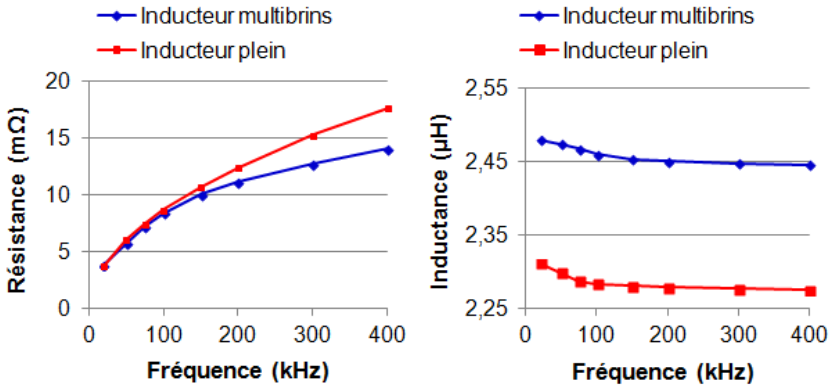


FIGURE 4.7 – Évolution, selon la fréquence, de la résistance et de l'inductance de l'inducteur 6×14×18 et de l'inducteur plein.

Nous constatons que l'inductance de l'inducteur à 1512 brins est légèrement plus élevée que celle de l'inducteur plein (écart relatif de 7 % environ) mais, dans les deux cas, l'inductance varie peu en fréquence. Nous remarquons également que les deux inducteurs ont une résistance identique pour une fréquence inférieure à 100 kHz mais, au-delà, celle de l'inducteur plein est plus élevée. L'inducteur multibrins est donc plus performant.

Ceci est confirmé par la figure 4.8 qui compare la puissance dissipée dans chacun des deux inducteurs en fonction de la fréquence entre 100 kHz et 400 kHz. Nous constatons que, sur cette plage de fréquence, l'inducteur multibrins dissipe moins de puissance que l'inducteur plein.

La figure 4.9 montre la répartition de la densité de puissance dissipée par l'inducteur plein à 50 kHz. Nous remarquons que la densité de puissance, et donc le courant, se concentre vers l'intérieur de l'hélice formée par l'inducteur.

La figure 4.10 (page 133) montre la répartition de la puissance dissipée dans chaque brin de l'inducteur multibrins à 1 kHz et 50 kHz. En raison de leur grand nombre, il n'est pas possible de voir chaque brin. Par contre, nous pouvons distinguer les paquets. La figure 4.11 (page 134) présente un grossissement d'une extrémité de l'inducteur multibrins à 1 kHz et 50 kHz.

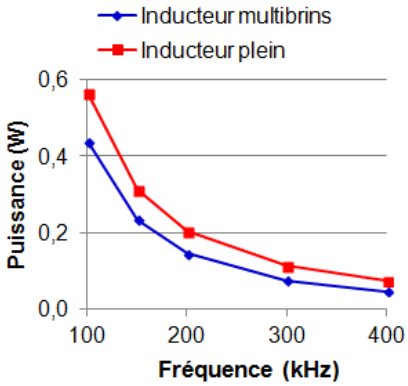


FIGURE 4.8 – Évolution de la puissance Joule dissipée dans l'inducteur 6×14×18 et dans l'inducteur plein en fonction de la fréquence.

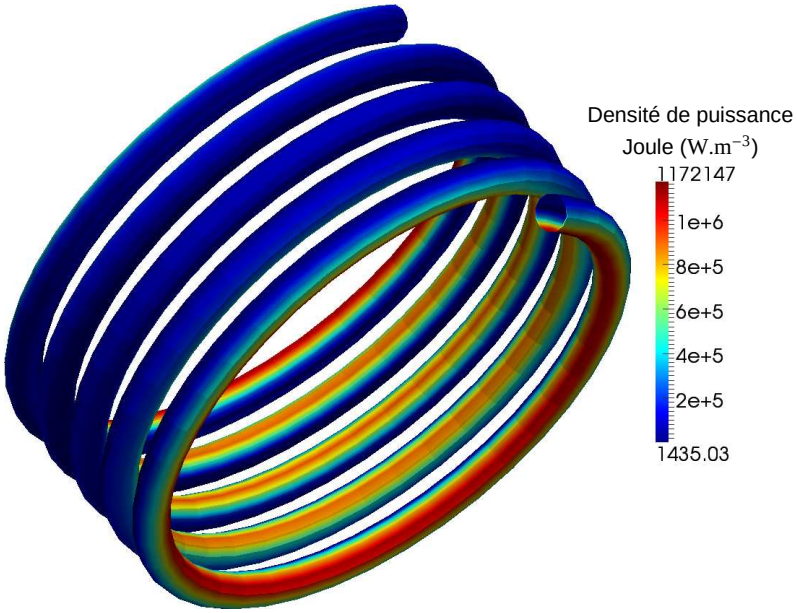
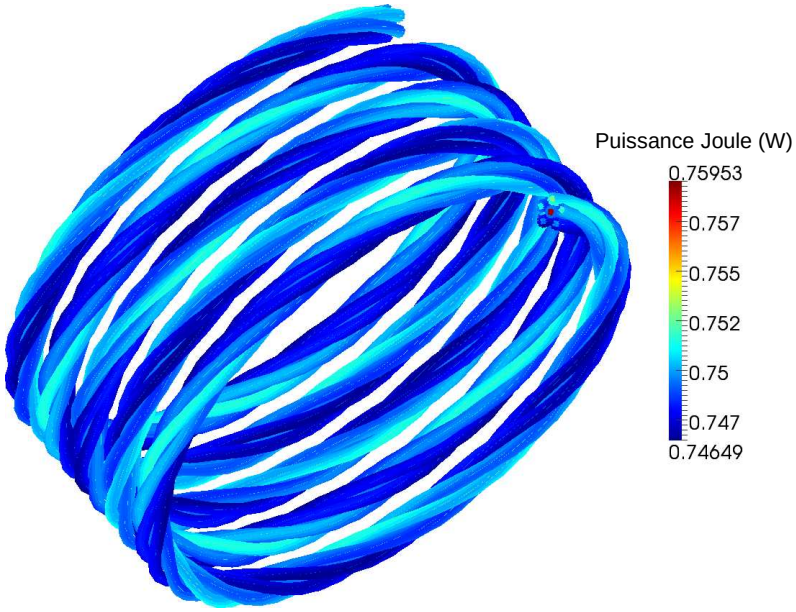


FIGURE 4.9 – Répartition de la densité de puissance dissipée par l'inducteur plein à 50 kHz.

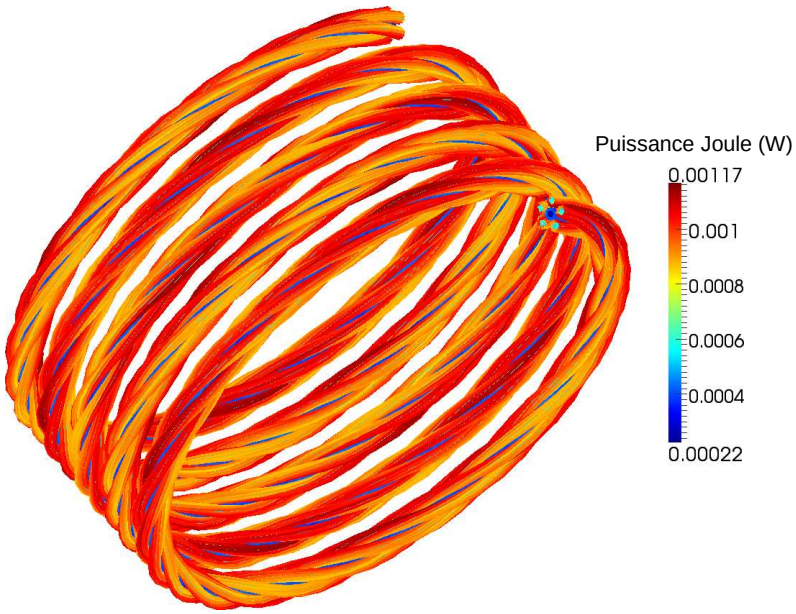
A 50 kHz, nous constatons que la puissance est dissipée principalement par les brins des paquets périphériques. Ainsi, le courant ne se concentre pas vers le centre de l'hélice formée par l'inducteur, mais est réparti assez uniformément entre les brins des paquets périphériques. Ces derniers dissipent environ 5.2 fois plus de puissance que les brins centraux.

A 1 kHz, nous observons que la puissance est répartie de manière quasi uniforme entre tous les brins, le rapport entre la puissance maximale et la puissance minimale étant seulement de 1.02. Ce sont les brins situés au centre du câble qui dissipent le plus de puissance.

La figure 4.12 (page 134) montre la répartition du sens des courants dans chaque



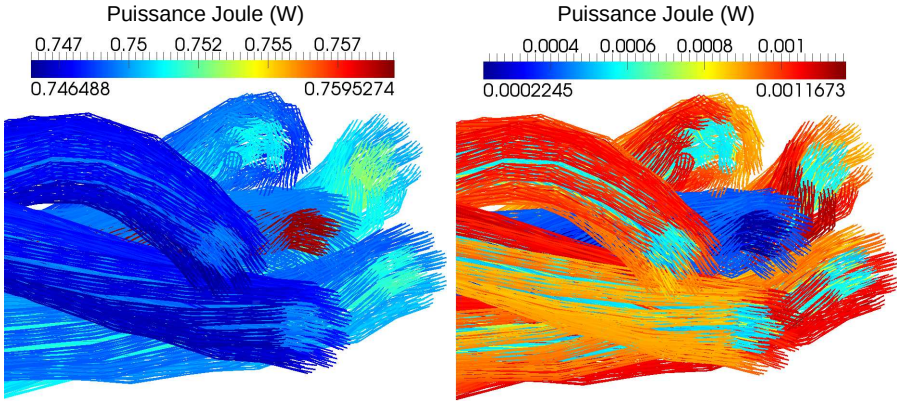
(a) F=1 kHz



(b) F=50 kHz

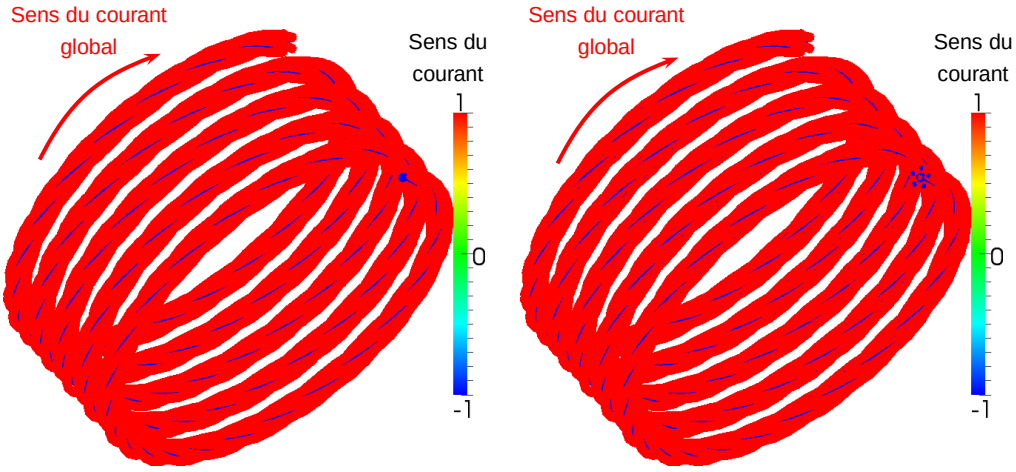
FIGURE 4.10 – Distribution de la puissance dissipée par brin dans l'inducteur 6×14×18 aux fréquences 1 kHz (a) et 50 kHz (b).

brin de l'inducteur 6×14×18 à 1 kHz et à 50 kHz. La figure 4.13 (page 135) présente un zoom de la figure 4.12 sur une extrémité du câble. La valeur +1 est attribuée aux brins dont le courant circule dans le même sens que celui du courant total de l'inducteur. La valeur −1 est attribuée aux brins dont le courant circule dans le sens opposé. Pour les deux fréquences, nous constatons qu'il existe des brins dans lesquels le courant circule dans le sens opposé au sens dans lequel circule le courant



(a) F=1 kHz (b) F=50 kHz

FIGURE 4.11 – Distribution de la puissance dissipée par brin dans l'inducteur 6×14×18 aux fréquences 1 kHz (a) et 50 kHz (b). Zoom sur une extrémité du câble.

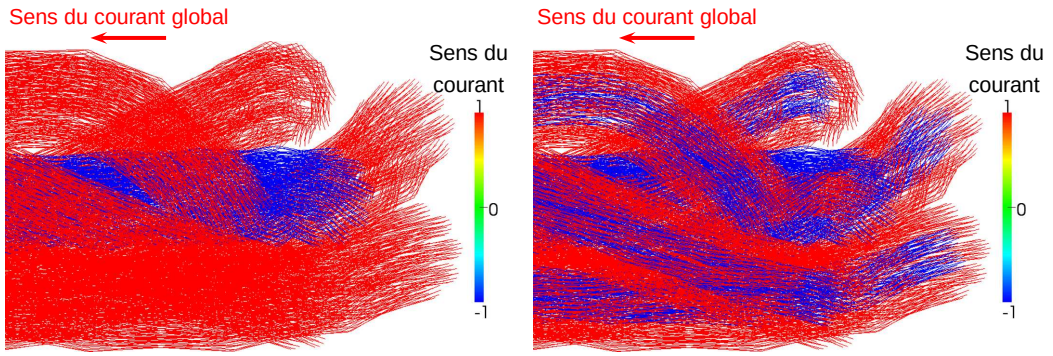


(a) F=1 kHz (b) F=50 kHz

FIGURE 4.12 – Sens du courant dans les brins de l'inducteur 6×14×18 à 1 kHz et à 50 kHz (+1 = sens du courant global, −1 = sens opposé).

total de l'inducteur. Sous réserve que le modèle filiforme soit toujours valable pour modéliser un nombre de brins aussi important et pour les fréquences étudiées, il semble donc que l'hypothèse du fil de Litz idéal, qui suppose un courant égal dans tous les brins, ne soit pas valable dans ce cas.

La simulation de l'inducteur 6×14×18 à la fréquence de base et parallélisée sur 1512 processeurs sur le cluster Ada de l'IDRIS a utilisé 1.2 To de mémoire, essentiellement en raison de la duplication du maillage, l'espace nécessaire au stockage du système linéaire étant de seulement 3.6 Mo. Le calcul a duré environ 1 heure et 23 minutes (temps *CPU* mesuré sur par le processus maître).



(a) F=1 kHz (b) F=50 kHz

FIGURE 4.13 – Sens du courant dans les brins de l'inducteur 6×14×18 à 1 kHz et à 50 kHz (+1 = sens du courant global, −1 = sens opposé). Zoom sur une extrémité du câble.

4.4 Conclusion et perspectives

Dans ce chapitre, nous avons présenté un modèle filiforme dérivé du modèle volumique présenté au chapitre 2 et dévolu à la modélisation d'inducteurs multibrins. Ce modèle est basé sur deux constations. Premièrement, la densité de courant varie peu dans la section d'un brin et sa direction est tangente au chemin du brin. Dans ce modèle, nous considérons que la densité de courant est constante dans un brin. Deuxièmement, le potentiel électrique varie linéairement le long d'un brin. Le modèle ne compte donc qu'un seul type d'inconnue : le courant par brin.

Nous avons montré la validité de ce modèle en le comparant au modèle volumique sur des configurations comptant un faible nombre de brins (55 brins au maximum). Dans le cas d'un inducteur 3×6, nous avons déterminé que le modèle filiforme est valable pour des fréquences inférieures à 500 kHz.

Nous avons également présenté un cas concret d'utilisation du modèle filiforme en comparant les comportements électromagnétiques d'un inducteur en cuivre plein et d'un inducteur multibrins (1512 brins) de sections équivalentes. Nous avons observé que, pour des fréquences supérieures à 100 kHz, l'inducteur multibrins est le plus performant. De plus, cette étude indique que l'hypothèse couramment employée pour la modélisation électromagnétique de fil de Litz et qui suppose un courant égal dans tous les brins n'est pas toujours valable dans le cas d'inducteurs comptant un grand nombre de brins.

Ces études nous ont permis de montrer l'intérêt du modèle filiforme ainsi que ses potentialités. Grâce à ce modèle, la modélisation d'inducteurs à grands nombres de brins est envisageable.

Cependant, des études complémentaires sont nécessaires pour poursuivre la validation du modèle. Il nous semble crucial de poursuivre les comparaisons avec le modèle volumique sur des inducteurs à plus grands nombres de brins. Ceci permet-

trait de tester la validité du modèle sur des inducteurs plus complexes et d'affiner notre connaissance du domaine de validité fréquentiel. En effet, pour l'étude sur l'inducteur à 1512 brins, nous avons supposé que le domaine de validité fréquentiel est limité à 500 kHz. Cependant, nous ne pouvons affirmer avec certitude que cela est vrai quel que soit le type d'inducteur modélisé.

D'autre part, des améliorations incontournables en termes d'implémentation sont à réaliser. En l'état actuel du logiciel, il n'est en effet pas possible de paralléliser le calcul sur un nombre de processus supérieur au nombre de brins. Ceci constitue clairement une limite car lorsque les brins sont longs, le nombre de calculs est important et les performances sont bridées par cette contrainte. Comme dans le cas du modèle volumique, un partage du maillage entre les processus permettrait de profiter davantage des performances apportées par le parallélisme. De manière générale, toutes les suggestions faites pour améliorer le modèle volumique (voir partie 2.5) trouvent également leur intérêt dans l'amélioration du modèle filiforme.

Conclusion générale et perspectives

La simulation numérique est un outil essentiel à la conception d'inducteurs multibrins à faibles pertes. Dans cette thèse, nous avons cherché à étudier l'influence de l'arrangement 3D interne des brins sur les pertes Joule dues aux effets de proximité dans ces inducteurs. La représentation des chemins suivis par les brins constitue en soit un premier problème de modélisation. Notre travail bibliographique ne nous a pas permis de trouver de modèle permettant de décrire ces chemins dans un inducteur quelconque, que ce soit de manière analytique ou numérique.

Nous avons réalisé un travail de compilation des modèles existants de calcul des pertes dans ce type de câble. Nous avons montré que la plupart de ceux-ci ont été développés dans le cadre d'études menées sur les bobinages de transformateurs ou sur les inducteurs de plaques de cuisson domestiques. Quelques études traitent des bobinages de transformateurs planaires. Dans tous ces cas, les modèles procèdent à des simplifications, compte tenu du fait que ces géométries présentent une certaine symétrie axiale. Deux grandes familles de modèles peuvent être distinguées.

Tout d'abord, les modèles analytiques. Nous avons constaté que la plupart de ceux-ci sont basés sur l'hypothèse du fil de Litz idéal – courant équitablement réparti entre les brins – dont la validité a été établie pour un système simple (solénoïde). Nous pensons que les bonnes cohérences entre modèles et mesures apparaissant dans la littérature sont dues aux configurations particulières des câbles étudiés. Nous avons trouvé un seul modèle dans lequel la géométrie des chemins des brins est prise en compte ([21], [22]), mais ceux-ci sont décrits analytiquement.

Quant aux modèles numériques de type Éléments Finis, compte tenu des contraintes de maillage, tous procèdent à des approximations permettant d'effectuer des calcul en 2D. Des modèles d'homogénéisation permettent ainsi de remplacer une portion de la géométrie réelle présentant une périodicité par une géométrie plus simple, affectée de propriétés électriques équivalentes et déterminées à partir de calculs effectués sur le motif de base de cette portion de géométrie. Certains de ces modèles peuvent aussi être employés en 3D. Pour pouvoir les utiliser, il faut donc être capable d'identifier ce motif. Nous avons montré que, dans le cas d'un inducteur multibrins, cette approche n'est pas forcément envisageable. Nous avons néanmoins rapporté des études récentes dans lesquelles la modélisation 3D de l'arrangement des brins dans des câbles en fil de Litz commence à être abordée ([36]).

Pour parvenir à répondre à notre problématique, nous avons décidé de développer le logiciel ModeLitz permettant la modélisation électromagnétique d'induc-

teurs multibrins. Compte tenu des capacités de calcul des machines, le logiciel a été implémenté de manière à fonctionner en parallèle. C'est pourquoi nous avons succinctement décrit l'évolution des architectures et des capacités de calcul des ordinateurs ainsi que l'avènement du calcul intensif et introduit les notions qui y sont rattachées. Nous avons également présenté la bibliothèque MPI utilisée pour gérer le parallélisme des calculs dans ModeLitz.

Nous avons décrit un algorithme permettant de calculer les chemins suivis par les brins dans un inducteur. Ainsi, nous sommes capables de modéliser la géométrie d'un câble multibrins comptant plusieurs niveaux de paquets et suivant lui-même un chemin arbitraire. Nous avons également décrit une première méthode numérique utilisée dans le logiciel. Il s'agit d'une méthode intégrale s'appuyant sur la loi d'Ohm, la loi de Biot et Savart et sur l'équation de conservation de la densité de courant. Elle permet d'effectuer une modélisation volumique en 3D et de calculer localement le potentiel électrique et la densité de courant électrique. Nous avons détaillé les avantages et inconvénients de cette méthode qui s'est imposée dans notre travail essentiellement en raison du fait qu'elle ne requiert pas de maillage de l'air et autres matériaux isolants, contrairement à la Méthode des Éléments Finis. La création du maillage ainsi que le contrôle de sa qualité s'en trouvent donc simplifiés et le nombre de degrés de liberté du système peut ainsi être diminué.

Nous avons décrit l'implémentation parallèle de la méthode intégrale dans le logiciel. Grâce à la parallélisation de la construction du système linéaire ainsi que de sa résolution, les temps de calcul peuvent être réduits et les capacités de modélisation peuvent être étendues en profitant des ressources en termes de puissance de calcul et d'espace de stockage de plusieurs machines. ModeLitz a été validé par comparaison avec le logiciel MIGEN. D'autre part, les tests menés sur différents systèmes de calcul, de la station de travail au cluster, ont démontré les excellentes performances de cet outil.

Grâce à ce logiciel, nous avons pu mettre en évidence certaines influences de la géométrie des brins sur l'importance et la répartition des pertes Joule au sein d'un inducteur multibrins. Nous avons ainsi pu montrer que l'alternance des sens de torsadage entre deux niveaux de paquets tend à diminuer la résistance de l'inducteur. Cette résistance est une résistance globale, générée à la fois par la résistance ohmique de chaque brin et par les effets de proximité entre brins. Par conséquent, le profil de l'augmentation de la résistance du câble en fonction de la fréquence doit permettre d'estimer l'importance des effets de proximité concomitants.

Nous avons également présenté une seconde approche, basée sur la méthode intégrale décrite auparavant. Dans celle-ci, les brins ne sont plus modélisés de manière volumique mais filiforme. Ce modèle présente l'avantage de réduire considérablement le nombre de degrés de liberté du système linéaire. En effet, il s'appuie sur la seule loi d'Ohm, associée à la loi de Biot et Savart, et les inconnues du système linéaire sont les courants circulant dans chaque brin. L'implémentation de ce modèle profite des développements effectués pour l'implémentation parallèle de l'approche volumique et permet des modélisations qui ne seraient pas envisageables avec cette

dernière. Par comparaison avec le modèle volumique, nous avons déterminé que le modèle filiforme est valable pour des fréquences inférieures à 500 kHz. Grâce aux études de validation que nous avons réalisées sur un câble comptant un faible nombre de brins (55 brins), nous avons montré que le courant électrique n'est pas toujours réparti de manière équitable entre les brins. Dans certains cas, l'importance des effets de proximité est telle que le sens du courant dans certains brins est opposé au sens du courant total circulant dans l'inducteur, qui est imposé par la différence de potentiel aux bornes de celui-ci.

Grâce au modèle filiforme, nous avons également comparé les comportements électromagnétiques d'un inducteur comptant 1512 brins et d'un inducteur plein de section équivalente. Nous avons montré que l'inducteur multibrins est plus performant que l'inducteur plein à hautes fréquences. Nous avons à nouveau constaté que le courant n'est pas réparti équitablement entre les brins. Dans certains brins, à cause des effets de proximité, le courant circule dans le sens opposé au sens du courant total traversant l'inducteur. Nous sommes donc enclin à penser que l'hypothèse du fil de Litz idéal doit être considérée avec prudence. Si elle semble permettre des calculs précis dans le cas de bobinages de transformateurs ou d'inducteurs de plaques de cuisson, elle ne semble pas appropriée à la modélisation d'inducteurs multibrins de type industriel.

Nous avons évoqué l'essentiel des perspectives ouvertes par ce travail. En premier lieu, des travaux doivent être menés afin d'approfondir la compréhension de l'impact de l'arrangement des brins sur les pertes dans les inducteurs. Nos études se sont essentiellement concentrées sur des inducteurs à deux niveaux de paquets et à faibles nombres de brins. L'étude d'inducteurs plus complexes, comptant davantage de brins et de niveaux, permettrait de s'approcher de la configuration des inducteurs industriels. D'autre part, il est capital de considérer la présence d'une charge. Pour toutes ces études, des modèles d'optimisations peuvent être mis en place.

Nous avons détaillé les limites de nos modèles. Pour pouvoir les repousser, des améliorations en termes de description des chemins d'inducteurs, de méthode numérique et d'implémentation sont nécessaires. Nous avons notamment déterminé que nous pouvons améliorer les performances du solveur en tenant compte de la topologie de la matrice du système linéaire. Nous suggérons également de redévelopper le logiciel en utilisant une implémentation hybride MPI-OpenMP afin de réduire les besoins en mémoire. Une telle approche permettrait aussi d'accélérer certaines étapes du calcul en réduisant le nombre et le volume des communications et en rendant l'équilibrage de charge en calculs plus dynamique. Nous avons d'ores et déjà identifié les portions d'algorithmes concernées et déterminé les solutions à implémenter.

Enfin, une fusion des compétences des logiciels MIGEN et ModeLitz pourrait permettre, à plus long terme, de disposer d'un logiciel parallèle de calcul électromagnétique dévolu à la modélisation de systèmes inductifs plus variés (creusets froids, lévitation électromagnétique, chauffage, brassage électromagnétique,...).

Appendices

Annexe A

Fonctionnement d'un code parallèle

Dans cette annexe, nous décrivons en détail le fonctionnement d'un programme parallélisé avec MPI.

A.1 Typographie

Nous présentons ici les conventions typographiques que nous allons suivre dans cette annexe.

Les instructions sensées être utilisées en ligne de commande ou dans des fichiers de configuration ont la forme suivante : `commande -[options de commande]`. De même, les noms de fichiers, sont écrits sous la forme `nom_de_fichier`, quelle que soit leur fonction (fichier texte ou exécutable de programme).

Pour l'insertion d'éléments de code C++ dans le texte, nous suivons le code couleur ci-dessous :

— `function`, pour les noms de fonctions ;
— `string`, pour les chaînes de caractères ;
— `comment`, pour les commentaires ;
— `variable1`, pour les noms des variables prédéfinies par le préprocesseur ;
— **`variable2`**, pour les noms de variables classiques.

Ce code couleur est le même, qu'il s'agisse d'inclusions de code dans le document sous forme de texte ou sous forme de figures. Pour alléger la lecture de ces dernières, les noms de variables ne sont pas en gras. Les variables prédéfinies ont donc l'écriture suivante : `variable1`. Quant aux autres variables, elles sont alors écrites `variable2`, tout comme les autres éléments de code pour lesquels aucune coloration syntaxique n'est définie.

A.2 Le cœur du programme

Le principe de base de la programmation parallèle d'un algorithme est simple : faire exécuter par plusieurs entités différentes tâches indépendantes. La conception d'un code parallèle nécessite donc d'identifier les parties de l'algorithme qui peuvent être partagées ainsi que les dépendances ou liens éventuels entre elles. Nous l'avons vu, les modèles de programmation parallèle intègrent des directives qui permettent de distribuer les calculs entre les entités et de gérer leurs dépendances. Ces entités de calculs peuvent être *hardware* (processeurs, cœurs de processeurs), et/ou *software* (programmes, threads).

Le calcul de la somme de tous les nombres contenus dans un tableau donné constitue un exemple basique d'algorithme parallèlisable. Le somme totale peut être divisée en plusieurs sous-sommes indépendamment calculables. Une fois ces sous-sommes calculées, la somme totale est évaluée. Cette dernière étape crée une dépendance entre les sous-calculs. L'algorithme 2 décrit les étapes de ce calcul.

Algorithme 2 : Algorithme parallèle de calcul de la somme des éléments d'un tableau.

Données : tableau de nombres, nombres d'élements du tableau, nombre de sous-calculs
Résultat : somme des éléments du tableau

```
1 Initialisation : partage du tableau entre les sous-entités de calcul;
2 pour chaque sous-calcul faire en parallèle
3     Somme partielle des éléments de chaque sous-tableau;
4 Récupération des sommes partielles;
5 Calcul de la somme totale;
```

Détaillons à présent le cœur du code du programme `Sum.exe` qui constitue un programme à peine plus complexe que celui décrit dans l'algorithme 2. Nous allons profiter de cette partie pour introduire certaines fonctionnalités MPI que nous utilisons dans notre logiciel de calcul électromagnétique. La *topologie*, l'organisation des processus, adoptée pour ce programme est dite *maître-esclaves* : un processus maître, en l'occurrence le processus de rang 0, est chargé de répartir le travail entre lui-même et tous les autres processus, les esclaves. Le programme évaluera la somme de tous les éléments d'un tableau nommé **`data`**. Nous allons examiner comment implémenter un programme qui effectuera les étapes suivantes :

- l'initialisation du tableau par le processus maître ;
- la répartition des données entre tous les processus ;
- le calcul des sommes partielles ;
- l'envoi au processus maître des sommes partielles ;
- le calcul de la somme totale ;
- l'envoi à tous les processus du résultat qu'ils afficheront ensuite.

La structure générale du code est donnée par la figure A.1. Nous allons décrire chaque partie séparément. Le code complet est disponible dans l'annexe B.

```
1  #include <mpi.h>       /* Gives MPI functions */
2  #include <stdlib.h>    /* Gives : exit() */
3  #include <iostream>    /* Gives : cout */
4
5  #define  ARRAYSIZE  16000000
6  #define  MASTER   0
7
8  double data[ARRAYSIZE];
9
10 using namespace std;
11
12 /***** Compute function *****/
13 double compute(int myoffset, int chunksize) {
14   /* Perform addition to each of myarray elements and keep mysum */
15   Voir figure A.3
16 }
17
18 int main(int argc, char *argv[]) {
19   /***** Initializations *****/
20   Voir figure A.4
21   /***** Master rank only ******/
22   Voir figure A.5
23   /***** Slaves ranks only *****/
24   Voir figure A.7
25   /***** Common programm *****/
26   Voir figure A.8
27 } /* end of main */
```

FIGURE A.1 – Structure générale de `Sum.cpp`

La taille du tableau **`data`** est ici de 16 millions d'éléments. Dans cet exemple, chacun des processus allouera son propre tableau. Chacun recevra, de la part du maître, une portion de données de taille **`chunksize`** à traiter et la stockera dans son propre tableau **`data`**, à la même place que dans le tableau **`data`** du maître, à partir de la position **`myoffset`**. Cela est illustré par la figure A.2. Il apparaît clairement que les processus esclaves réservent largement plus de mémoire que nécessaire. Nous pourrions imaginer un programme plus économe en mémoire dans lequel, pour chaque processus esclave, le tableau **`data`** aurait une taille **`chunksize`**, mais, pour conserver la simplicité du programme, nous allons négliger ce détail.

Chaque processus utilisera la fonction **`compute`** afin d'évaluer sa propre somme partielle. Le code de la fonction **`compute`** est présenté dans la figure A.3.

La figure A.4 (page 147) présente la partie d'initialisation du programme. On y trouve notamment l'initialisation du contexte de communication MPI *via* l'appel de la fonction **`MPI_Init`**, indispensable pour l'utilisation des communications et autres routines MPI. Cette fonction procède à la création du communicateur par défaut **`MPI_COMM_WORLD`**. Pour connaître le nombre de processus présents dans ce communicateur, en fait le nombre total de processus sur lequel tourne le programme, il suffit d'employer **`MPI_Comm_size`**. Ce nombre est retourné dans la variable **`nbr_procs`**.

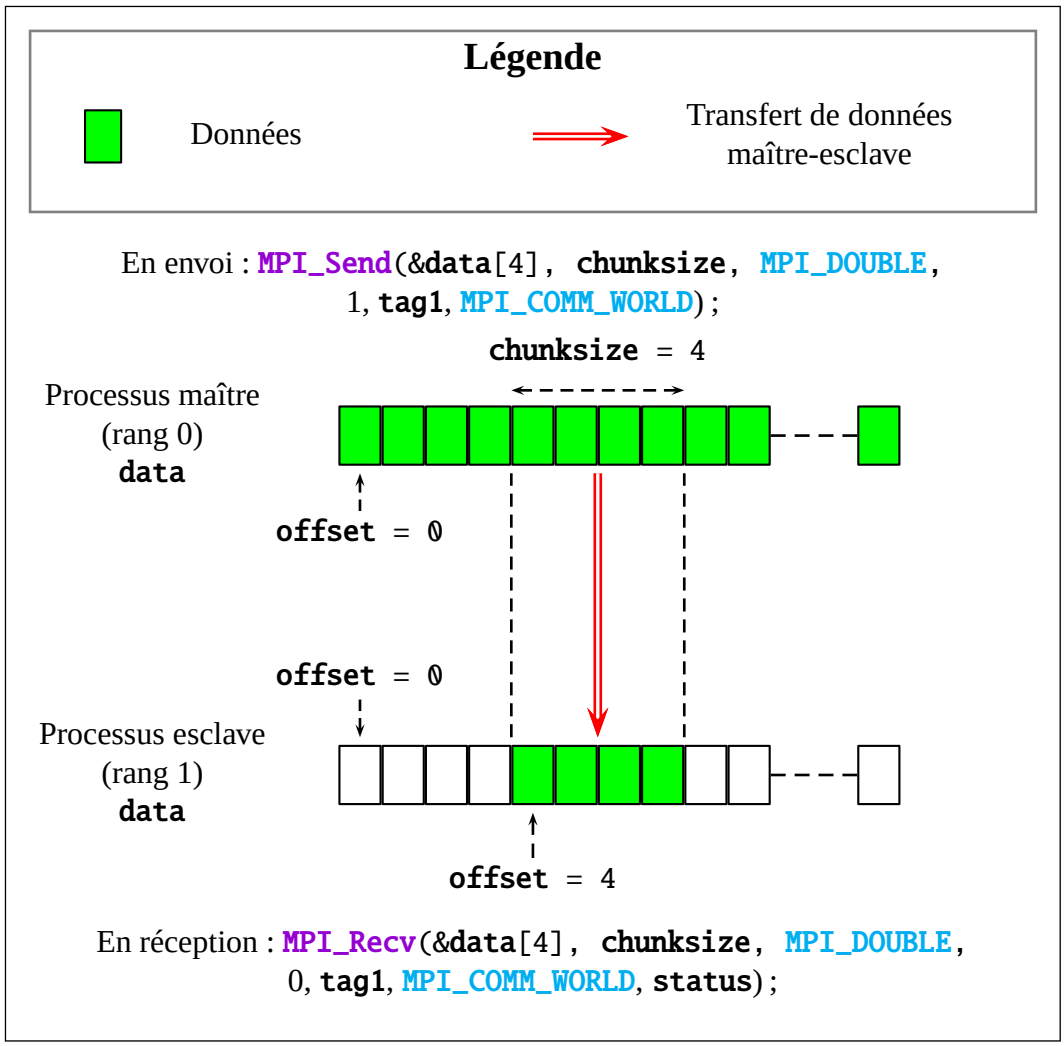


FIGURE A.2 – Principe de répartition des données entre le processus maître et le processus esclave de rang 1. La taille (**chunksize**) des blocs de données est de quatre éléments. Le processus 1 stocke le bloc reçu dans son propre tableau **data** à partir de la position (ou **offset**) 4.

```
1  /***** Compute function *****/
2  /* Cette partie s'insère ici dans la figure A.1 */
3  double compute(int myoffset, int chunksize) {
4    int i;
5    double mysum;
6    /* Perform addition to each of myarray elements and return mysum */
7    mysum = 0;
8    for (i = myoffset; i < myoffset + chunksize; i++) mysum += data[i];
9    return mysum;
10 }
```

FIGURE A.3 – Code de la fonction **compute**.

La fonction **MPI_Comm_rank** permet de connaître le rang d'un processus. La fonction **MPI_Abort** permet d'interrompre le programme en lançant une erreur tout en

fermant proprement le contexte de communication.

```
int main(int argc, char *argv[]) {
  /***** Initializations *****/
  /* Cette partie s'insère ici dans la figure A.1 */

  int nbr_procs, rank, rc, dest, offset, i, tag1, tag2, source, chunksize;

  double mysum, sum;
  MPI_Status status;

  /* Initialize MPI communications context
   * Create the default MPI communicator MPI_COMM_WORLD
   */
  MPI_Init(&argc, &argv);

  /* Get number of MPI processes in MPI_COMM_WORLD communicator */
  MPI_Comm_size(MPI_COMM_WORLD, &nbr_procs);

  if (nbr_procs % 4 != 0) {
    cout << "Quitting. Number of MPI ranks must be divisible by 4." << endl;
    MPI_Abort(MPI_COMM_WORLD, rc);
    exit(0);
  }

  /* Get process rank in MPI_COMM_WORLD communicator */
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  /* Compute size of array partitions */
  chunksize = ARRAYSIZE / nbr_procs;
  tag1 = 1;
  tag2 = 2;

  /***** Master rank only ******/

  /***** Slaves ranks only *****/

  /***** Common programm *****/

} /* end of main */
```

FIGURE A.4 – Initialisation des variables et du contexte de communication MPI.

Les variables de type entier **tag1** et **tag2** sont des étiquettes utilisées pour distinguer les communications. Dans ce programme, elles n'ont pas d'utilité particulière. Quant à la variable **status** de type MPI_Status, elle contient des informations sur les communications (source, destinataire, ...).

Intéressons-nous à présent au travail effectué par le processus maître et présenté dans la figure A.5. On distingue simplement le maître des esclaves par le test (**rank** == MASTER). Le tableau **sample_sum** alloué par le maître servira plus loin à réceptionner les sommes partielles de tous les processus.

Le maître est d'abord chargé d'initialiser le tableau **data**. Il communique ensuite à chaque processus esclave la valeur **offset** qui indiquera à cet esclave à

```
1  /***** Master rank only ******/
2  /* Cette partie s'insère ici dans la figure A.1 */
3  if (rank == MASTER) {
4    /* Allocate the sample_sums array and initialize it */
5    double sample_sums[nbr_procs];
6    sum = 0;
7    for (i = 0; i < ARRAYSIZE; i++) {
8      data[i] = i * 1.0;
9      sum = sum + data[i];
10   }
11   cout << "Sequential computation of array sum = " << sum << endl;
12
13   /* Send each rank its portion of the array - master keeps 1st part */
14   offset = chunksize;
15   cout<<"Master (rank "<<MASTER<<") keeps "<< chunksize <<" elements,
     offset= "<<offset<<endl;
16   for (dest = 1; dest < nbr_procs; dest++) {
17     MPI_Send(&offset, 1, MPI_INT, dest, tag1, MPI_COMM_WORLD);
18     MPI_Send(&data[offset], chunksize, MPI_DOUBLE, dest, tag2,
     MPI_COMM_WORLD);
19     cout << "Master sends " << chunksize << " elements to rank " << dest <<
      " offset= " << offset << endl;
20     offset = offset + chunksize;
21   }
22
23   /* Master does its part of the work */
24   offset = 0;
25   mysum = compute(offset, chunksize);
26
27   /* Wait to receive all sample sums in one time */
28   dest = MASTER;
29   MPI_Gather(&mysum, 1, MPI_DOUBLE, &sample_sums[0], 1, MPI_DOUBLE, dest,
     MPI_COMM_WORLD);
30   /* Print sample results */
31   cout  << "Sample sums: " << endl;
32   for (i = 0; i < nbr_procs; i++)cout << "  Rank "<< i << " : sample sum =
     " << sample_sums[i]<<endl;
33
34   /* Computation of the total sum */
35   MPI_Reduce(&mysum, &sum, 1, MPI_DOUBLE, MPI_SUM, MASTER, MPI_COMM_WORLD);
36   cout << "*** Final sum= " << sum << " ***" << endl;
37 } /* end of master section */
```

FIGURE A.5 – Code exécuté par le processus maître.

partir de quel emplacement, dans son propre tableau **data**, il devra stocker les données communiquées par le maître (voir la figure A.2). Puis l'envoi des données est effectué. Ces communications sont réalisées grâce à la fonction **MPI_Send**.

Le prototype de la fonction **MPI_Send** est le suivant :

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

avec :

— **buf**, l'adresse du *buffer* d'envoi, autrement dit l'adresse mémoire (pointeur) du premier élément à envoyer ;

— **count**, la taille du *buffer* d'envoi, soit le nombre d'éléments à envoyer et lus à partir de l'adresse **buf** ;

— **datatype**, le type des données envoyées ;
— **dest**, le rang du processus destinataire au sein communicateur **comm** ;
— **tag**, l'étiquette de la communication ;
— **comm**, le communicateur dans lequel s'inscrit cette communication.

La communication réalisée grâce à `MPI_Send` est dite *point-à-point* car un seul processus émetteur s'adresse à un seul processus destinataire.

Tous les processus ayant reçu leur part de données à traiter, chacun évalue alors sa propre somme partielle *via* la fonction `compute`. Une fois toutes les sommes partielles calculées, par le maître comme par les esclaves, elles sont réceptionnées dans le tableau **sample_sum** par le processus maître grâce à la fonction `MPI_Gather` qui initie une communication *collective*. Les communications collectives diffèrent des communications point-à-point en ce que plus d'un processus émetteur et/ou plus d'un processus destinataire sont impliqués. `MPI_Gather` permet ici à plusieurs processus de communiquer en même temps un ensemble de données vers un même processus, ici le processus maître. Les données sont concaténées dans un *buffer* de réception. Le prototype cette fonction est le suivant :

```
int MPI_Gather(void *sendbuf, int sendcnt,
MPI_Datatype sendtype, void *recvbuf, int recvcnt,
MPI_Datatype recvtype, int root,  MPI_Comm comm);
```

avec :

— **sendbuf**, l'adresse du *buffer* d'envoi ;
— **sendcnt**, la taille du *buffer* **sendbuf** ;
— **sendtype**, le type des données envoyées ;
— **recvbuf**, l'adresse du *buffer* de réception
— **recvcnt**, la taille du *buffer* **recvbuf** ;
— **recvtype**, le type des données reçues ;
— **root**, le rang, au sein communicateur **comm**, du processus qui recevra toutes les données ;
— **comm**, le communicateur dans lequel s'inscrit cette opération.

Les arguments concernant la réception des données n'ont d'importance que pour le processus récepteur **root**. Les tailles **sendcnt** et **recvcnt** peuvent être différentes. Dans l'utilisation que nous faisons ici de cette fonction, elles sont égales. La figure A.6 expose le principe de fonctionnement de `MPI_Gather` sous cette condition.

A ce stade, le maître pourrait aisément calculer la somme totale. Néanmoins, nous avons choisi d'effectuer ce calcul autrement afin de présenter une autre fonctionnalité : les opérations collectives, aussi appelées opération *de réduction* [1]. Dans notre cas, c'est la fonction `MPI_Reduce` qui permet de le faire. Le prototype de cette fonction est le suivant :

1. Ces opérations ne sont pas propres à MPI. On les retrouve également dans d'autres modèles de programmation parallèle comme OpenMP.

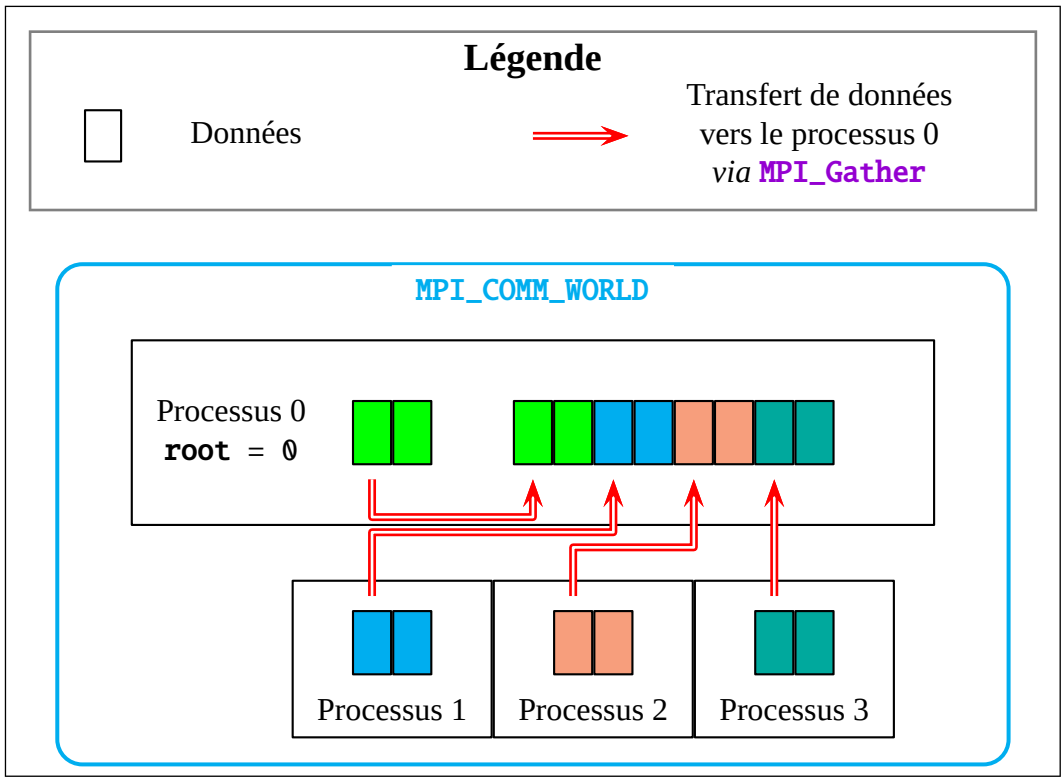


FIGURE A.6 – Principe de transfert de données grâce à la fonction `MPI_Gather` au sein du communicateur `MPI_COMM_WORLD` comptant quatre processus. Chacun des processus envoie un bloc de données de taille deux vers le processus 0.

```
int MPI_Reduce(void *sendbuf, void *recvbuf, int
count, MPI_Datatype datatype, MPI_Op op, int root,
MPI_Comm comm);
```

avec :

- **sendbuf**, l'adresse du *buffer* d'envoi ;
- **recvbuf**, l'adresse du *buffer* de réception
- **count**, la taille des *buffers* **sendbuf** et **recvbuf** ;
- **datatype**, le type des données envoyées ;
- **op**, le type d'opération de réduction souhaitée ;
- **root**, le rang, au sein communicateur **comm**, du processus sur lequel aboutira le résultat de l'opération ;
- **comm**, le communicateur dans lequel s'inscrit cette opération.

Dans notre exemple, le *buffer* d'envoi de chaque processus est de type `double` (on utilise alors le type MPI `MPI_DOUBLE`) et de taille 1 et a pour adresse l'adresse de la variable **mysum** allouée par le processus. Le *buffer* de réception est de même type et de même taille et est adressé à l'emplacement **&sum** par le processus maître, processus sur lequel sera reçu le résultat de l'opération. La valeur `MPI_SUM` indique que l'opération de réduction est une somme.

La procédure de distinction des processus esclaves est analogue à celle em-

ployée pour le maître : on effectue le test (**rank** > **MASTER**). La figure A.7 montre le code exécuté par les esclaves.

```
1  /***** Slaves ranks only *****/
2  */ Cette partie s'insère ici dans la figure A.1 */
3  if (rank > MASTER) {
4    /* Receive my portion of array from the master rank */
5    source = MASTER;
6    MPI_Recv(&offset, 1, MPI_INT, source, tag1, MPI_COMM_WORLD, &status);
7    MPI_Recv(&data[offset], chunksize, MPI_DOUBLE, source, tag2,
     MPI_COMM_WORLD, &status);
8
9    mysum = compute(offset, chunksize);
10
11   /* Send my result back to the master rank */
12   dest = MASTER;
13   MPI_Gather(&mysum, 1, MPI_DOUBLE, &mysum, 1, MPI_DOUBLE, dest,
     MPI_COMM_WORLD);
14
15   /* Computation of the total sum */
16   MPI_Reduce(&mysum, &sum, 1, MPI_DOUBLE, MPI_SUM, MASTER, MPI_COMM_WORLD);
17 } /* end of slaves */
```

FIGURE A.7 – Code exécuté par les processus esclaves.

Ces processus reçoivent de la part du maître la position à partir de laquelle, dans leur tableau **data**, ils devront stocker les données à traiter, qu'ils reçoivent ensuite. Ils effectuent donc deux réceptions grâce à la fonction **MPI_Recv**. Cette fonction est le pendant logique de la fonction **MPI_Send**, aussi son prototype est-il simplement [2] :

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

avec :

— **buf**, l'adresse du *buffer* de réception ;
— **count**, la taille du *buffer* de réception, ou encore le nombre d'éléments à réceptionner et à stocker à partir de l'adresse **buf** ;
— **datatype**, le type des données reçues ;
— **source**, le rang du processus expéditeur au sein communicateur **comm** ;
— **tag**, l'étiquette de la communication ;
— **comm**, le communicateur dans lequel s'inscrit cette communication ;
— **status**, le statut de la communication.

En ce qui concerne la communication collective **MPI_Gather** le code est, à peu de choses près, le même que celui du processus maître. Ceci est également le cas pour l'opération **MPI_Reduce**. Notons toutefois que, dans ces fonctions, les

2. Voir la figure A.2 pour le lien entre ces deux fonctions.

adresses des *buffers* de réception sont des arguments qui n'ont une importance que pour le maître.

Nous avons précédemment suggéré que notre programme procède à une dernière étape : la communication à tous les processus de la somme totale puis son affichage par chacun des processus. Cette étape est commune à tous les processus, maître comme esclaves. Le code commun qui la décrit est présenté dans la figure A.8.

```
1  /***** Common programm *****/
2  /* Cette partie s'insère ici dans la figure A.1 */
3  /* Master sends final result to all processes */
4  MPI_Bcast(&sum, 1, MPI_DOUBLE, MASTER, MPI_COMM_WORLD);
5
6  /* Each process display the final result */
7  for (i = 0; i < nbr_procs; i++) {
8    if (i == rank) cout << "  Rank " << rank << " : Final sum = " << sum <<
     endl;
9
10   /* Wait for other processes : here use to ordonate processes displays */
11   MPI_Barrier(MPI_COMM_WORLD);
12 }
13
14 /* Close MPI communications context */
15 MPI_Finalize();
```

FIGURE A.8 – Code commun à tous les processus.

La communication **MPI_Bcast** consiste en un envoi collectif à l'ensemble des processus d'un communicateur. Chaque processus l'implémente donc selon le prototype suivant :

```
int MPI_Bcast(void *buf, int count, MPI_Datatype datatype,
int source, MPI_Comm comm);
```

avec :

- **buf**, l'adresse du *buffer* d'envoi (pour le processus émetteur) ou de réception (pour les autres processus) ;
- **count**, la taille du *buffer* d'envoi ou de réception ;
- **datatype**, le type des données transférées ;
- **source**, le rang du processus émetteur au sein communicateur **comm** ;
- **comm**, le communicateur dans lequel est effectuée cette communication.

La figure A.9 illustre le comportement de la fonction **MPI_Bcast**.

Quant à la fonction **MPI_Barrier**, elle permet la synchronisation des processus. Dans notre exemple, elle sert à séquentialiser l'affichage des processus et ainsi à éviter que tous les processus délivrent leur message en même temps, ce qui pourrait rendre la lecture difficile car l'ordre n'est pas assuré. Pire encore, l'affichage en ligne de commande est forcément séquentialisé, aussi les messages de plusieurs processus peuvent apparaître imbriqués les uns dans les autres.

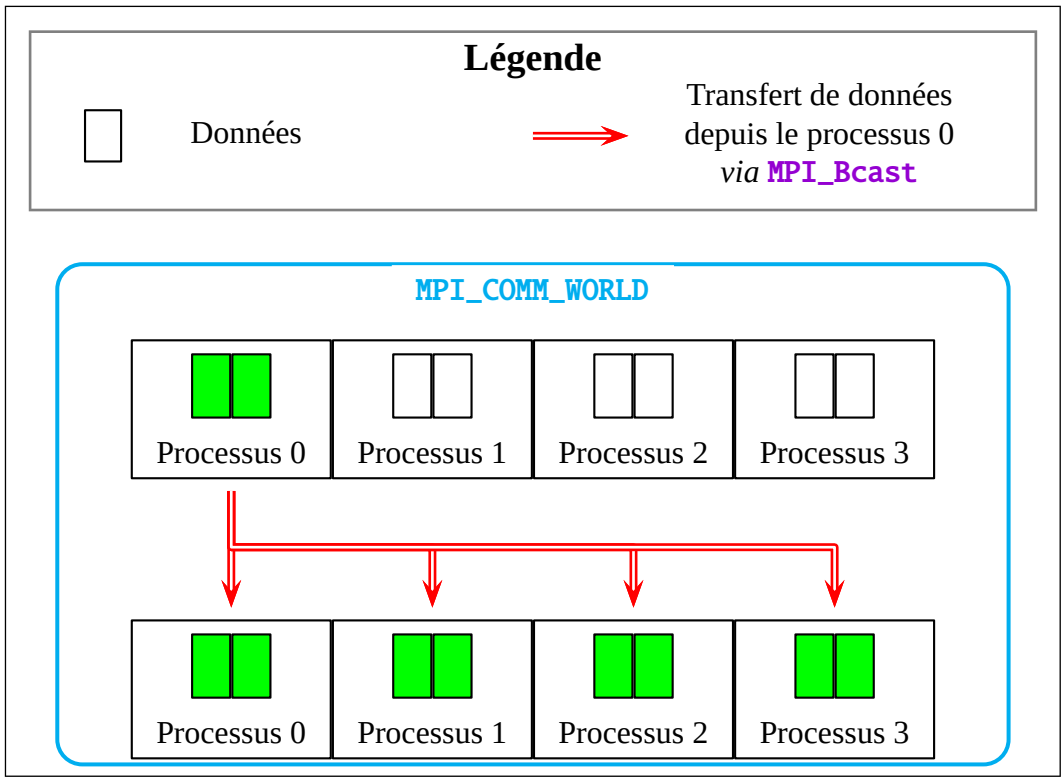


FIGURE A.9 – Principe de transfert de données grâce à la fonction `MPI_Bcast` avec quatre processus. Le processus 0 envoie un bloc de données de taille deux vers tous les processus du communicateur `MPI_COMM_WORLD`.

Enfin, l'instruction `MPI_Finalize` clos le contexte de communication MPI. Le communicateur `MPI_COMM_WORLD` est alors détruit et aucune nouvelle communication ne peut être effectuée.

Nous avons présenté un exemple simple de programme parallélisé grâce à MPI. Nous avons détaillé plusieurs communications. Il existe bien d'autres fonctionnalités, néanmoins, nous avons exposé l'essentiel des fonctions MPI que nous utilisons dans notre logiciel.

Annexe B

Exemple de code MPI

Cet appendice contient le code complet détaillé dans la section A.2. Rappel : nous employons la typographie suivante pour l'écriture d'éléments de code C++ dans les figures :

— `function`, pour les noms de fonctions ;
— `string`, pour les chaînes de caractères ;
— `comment`, pour les commentaires ;
— `variable1`, pour les noms des variables prédéfinies par le préprocesseur ;
— `variable2`, pour les noms de variables classiques, comme pour le restant des éléments de code.

```
1  #include <mpi.h>       /* Gives MPI functions */
2  #include <stdlib.h>    /* Gives : exit() */
3  #include <iostream>    /* Gives : cout */
4
5  #define  ARRAYSIZE  16000000
6  #define  MASTER     0
7
8  double data[ARRAYSIZE];
9
10 using namespace std;
11
12 /***** Compute function *****/
13 double compute(int myoffset, int chunk, int myrank) {
14   int i;
15   double mysum;
16   /* Perform addition to each of myarray elements and keep mysum */
17   mysum = 0;
18   for (i = myoffset; i < myoffset + chunk; i++) mysum = mysum + data[i];
19   return mysum;
20 }
21
22 int main(int argc, char *argv[]) {
23
24   /***** Initializations *****/
25
26   int nbr_procs, rank, rc, dest, offset, i, tag1,
27           tag2, source, chunksize;
28
29   double mysum, sum;
30   MPI_Status status;
31
32   /* Initialize MPI communications context
```

```
33    * Create the default MPI communicator MPI_COMM_WORLD
34    */
35  MPI_Init(&argc, &argv);
36
37  /* Get number of MPI processes in MPI_COMM_WORLD communicator */
38  MPI_Comm_size(MPI_COMM_WORLD, &nbr_procs);
39
40  if (nbr_procs % 4 != 0) {
41    cout << "Quitting. Number of MPI ranks must be divisible by 4." << endl;
42    MPI_Abort(MPI_COMM_WORLD, rc);
43    exit(0);
44  }
45
46  /* Get process rank in MPI_COMM_WORLD communicator */
47  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
48
49  /* Compute size of array partitions */
50  chunksize = (ARRAYSIZE / nbr_procs);
51  tag1 = 1;
52  tag2 = 2;
53
54  /***** Master rank only ******/
55  if (rank == MASTER) {
56    /* Allocate the sample sums array */
57    double sample_sums[nbr_procs];
58
59    /* Initialize the data array */
60    sum = 0;
61    for (i = 0; i < ARRAYSIZE; i++) {
62      data[i] = i * 1.0;
63      sum = sum + data[i];
64    }
65    cout << "Sequential computation of array sum = " << sum << endl;
66
67    /* Send each rank its portion of the array - master keeps 1st part */
68    offset = chunksize;
69    cout<<"Master (rank "<<MASTER<<") keeps "<< chunksize <<" elements, "
70            "offset= "<<offset<<endl;
71    for (dest = 1; dest < nbr_procs; dest++) {
72      MPI_Send(&offset, 1, MPI_INT, dest, tag1, MPI_COMM_WORLD);
73      MPI_Send(&data[offset], chunksize, MPI_DOUBLE, dest, tag2,
    MPI_COMM_WORLD);
74      cout << "Master sends " << chunksize << " elements to rank " << dest
75              << " offset= " << offset << endl;
76      offset = offset + chunksize;
77    }
78
79    /* Master does its part of the work */
80    offset = 0;
81    mysum = compute(offset, chunksize, rank);
82
83    /* Wait to receive sample sums all processes in one time */
84    dest = MASTER;
85    MPI_Gather(&mysum, 1, MPI_DOUBLE, &sample_sums[0], 1, MPI_DOUBLE,
86            dest, MPI_COMM_WORLD);
87    /* Print sample results */
88    cout  << "Sample sums: " << endl;
89    offset = 0;
90    for (i = 0; i < nbr_procs; i++)cout << "  Rank "<< i << " : sample sum =
    " << sample_sums[i]<<endl;
91
92    /* Computation of the total sum */
93    MPI_Reduce(&mysum, &sum, 1, MPI_DOUBLE, MPI_SUM, MASTER, MPI_COMM_WORLD);
94    cout << "*** Final sum= " << sum << " ***" << endl;
95  } /* end of master section */
96
97
```

```
98    /***** Slaves ranks only *****/
99    if (rank > MASTER) {
100     /* Receive my portion of array from the master rank */
101     source = MASTER;
102     MPI_Recv(&offset, 1, MPI_INT, source, tag1, MPI_COMM_WORLD, &status);
103     MPI_Recv(&data[offset], chunksize, MPI_DOUBLE, source, tag2,
104             MPI_COMM_WORLD, &status);
105
106     mysum = compute(offset, chunksize, rank);
107
108     /* Send my result back to the master rank */
109     dest = MASTER;
110     MPI_Gather(&mysum, 1, MPI_DOUBLE, &mysum, 1, MPI_DOUBLE, dest,
111             MPI_COMM_WORLD);
112
113     /* Computation of the total sum */
114     MPI_Reduce(&mysum, &sum, 1, MPI_DOUBLE, MPI_SUM, MASTER, MPI_COMM_WORLD);
115   } /* end of slaves */
116
117
118   /***** Common programm *****/
119   /* Master sends final result to all processes */
120   MPI_Bcast(&sum, 1, MPI_DOUBLE, MASTER, MPI_COMM_WORLD);
121
122   /* Each process display the final result */
123   for (i = 0; i < nbr_procs; i++) {
124     if (i == rank) cout << "  Rank " << rank << " : Final sum = " << sum <<
      endl;
125
126     /* Wait for other processes : here use to ordonate processes displays */
127     MPI_Barrier(MPI_COMM_WORLD);
128   }
129
130   /* Close MPI communications context */
131   MPI_Finalize();
132
133 } /* end of main */
```

Annexe C

Exemple complet d'une simulation sous ModeLitz

Dans cet appendice, nous présentons un exemple complet de modélisation et constituant la simulation la plus conséquente ayant été menée avec ModeLitz.

C.1 Description du calcul

Nous modélisons un câble 5×11 droit. Le tableau C.1 présente sa structure. La longueur de câble simulée est de 35 mm. Les brins en cuivre ont un rayon de 50 µm et sont entourés d'un isolant d'épaisseur 5 µm. La différence de potentiel aux bornes de l'inducteur est de 0.35 V et la fréquence choisie est de 10 kHz. Les calculs sont effectués grâce au modèle volumique.

Indice de niveau	1	0
Nombre de paquets du niveau	5	11
Sens de torsadage	-1	1
Pas de torsadage	0.035	0.025

TABLE C.1 – Configuration du câble 5×11.

ModeLitz implémente plusieurs modes de description d'une simulation. Un premier mode, utilisé essentiellement à des fins de développement, permet d'implémenter directement la modélisation dans le logiciel. Un deuxième mode permet de charger les paramètres de certaines simulations réalisées sous MIGEN pour les exécuter sous ModeLitz. Il est ainsi possible de comparer les résultats donnés par les deux codes. Enfin, le troisième mode, le mode standard, est conçu pour permettre un usage simplifié du logiciel, particulièrement utile pour réaliser des campagnes de calculs. Dans ce mode, l'ensemble des paramètres nécessaires à la mise en place de la simulation est transmis au logiciel grâce à un fichier de description. La figure C.1 présente le fichier décrivant le calcul considéré dans cette annexe.

```
## Fichier de description automatique du projet '
    vol_cable_5x11_droit_length_0.035m_10000Hz' sous ModeLitz_1.3.5

#BEGIN General
ProjectPath /home/scapolan/MI_ISIS_validations/calculs_these/2.7
    _et_ulterieurs/compare_filiform_volumique/cable_5x11/
ProjectName vol_cable_5x11_droit_length_0.035m_10000Hz
ProjectOptions -erase -f
Model VOLUME_MODEL

Frequency 10000
PathsNbr 1
ObjectsNbr 1
#END General

#BEGIN Path
PathType STRAIGTH_PATH
PathAxis 0 0 1
PathNdiv 10
Length 0.035
#END Path

#BEGIN Object
ObjectCompositeType CABLE_COMP_TYPE
CableType LITZ_CABLE
GeomType WIRE_GEOM_OBJ
WireRadius 50e-6
InsulatorThickness 5e-6
CircleNdiv 1
DV/m 10
Vref 0
HoleRadius 0.0
TargetElemsNbr 30
XYZCenter 0 0 0
LitzDescr
LevelNbr 2
## LevelIdx          |   1   |   0   |
   LevelPacketNbr    |   5   |   11  |
   WindingWise       |  -1   |   1   |
   ThetaInit         |   0   |   0   |
   LengthCoef        | 0.035 | 0.025 |
#END Object
```

FIGURE C.1 – Fichier de description de la modélisation du câble 5×11.

Quel que soit le mode choisi, en fin de calcul, le logiciel crée automatiquement le fichier de description correspondant à la modélisation qu'il a exécuté. Il est donc aisé de relancer la simulation telle quelle ou avec de nouveaux paramètres.

C.2 Lancement du calcul

Le fichier de description ne contient pas d'informations concernant la parallélisation du calcul : le nombre de processus est choisi au lancement du programme (cf. partie 1.3.8). Le calcul a été lancé sur 132 processus sur les quatre nœuds du cluster du laboratoire. La commande de lancement est donc la suivante :

```
mpiexec -n 132 -machinefile ./machines ./modelitz_1.3.5
    -general -load ./fichier_description
```

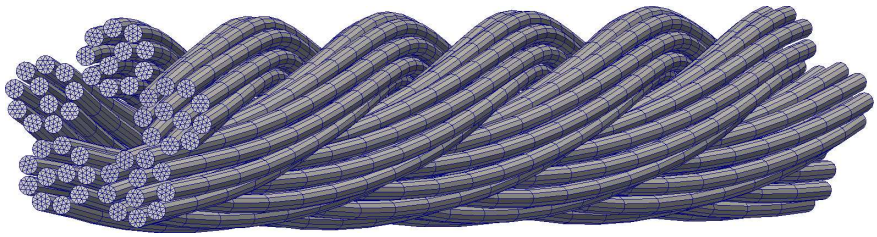
La répartition des processus entre les nœuds est décrite dans le fichier `machines` dont le contenu est le suivant :

```
node001 : 33
node002 : 33
node003 : 33
node004 : 33
```

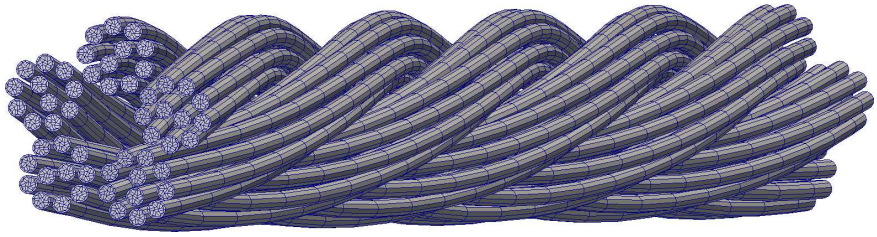
L'argument `-general` désigne le mode de description standard. L'argument `-load ./fichier_description` précise le chemin du fichier de description.

C.3 Maillage

A partir des informations contenues dans le fichier de description, le logiciel construit la géométrie puis le maillage. Celui-ci est stocké dans un fichier au format *vtk* avant le lancement de la construction du système linéaire. Il est ainsi possible de visualiser le maillage correspondant aux paramètres choisis et, si besoin, d'interrompre le calcul pour les modifier. La figure C.2 présente les maillages $\boldsymbol{V}$ et $\vec{\boldsymbol{J}}$.



(a) Maillage $\boldsymbol{V}$.



(b) Maillage $\vec{\boldsymbol{J}}$.

FIGURE C.2 – Maillages du câble 5×11.

C.4 Suivi du calcul

Le logiciel fourni en continu l'état de la progression de chaque étape. Ces informations sont envoyées soit dans le terminal de commande depuis lequel le calcul a été lancé, soit dans un fichier texte. La figure C.3 donne un aperçu des informations essentielles délivrées en cours de calcul.

```
Run project called 'vol_cable_5x11_droit_length_0.035m_10000Hz'
Project location : /home/scapolan/MI_ISIS_validations/calculs_these/2.7
    _et_ulterieurs/compare_filiform_volumique/cable_5x11/

 Disposition du cable :
5 x 11
  Litz_Cable : packets creation   : |------------------------------| 100 %
  Litz_Cable : wires building  : |------------------------------| 100 %

Le systeme compte :
   36960 nodes J3
   18480 nodes J2
   3960 nodes J1
   30305 nodes V
La taille du systeme est 182105
Il comptera 151800 lignes  pleines, soit 83.4 % des lignes de la matrice
Occupation memoire totale pour le maillage         : 444.66 Go (repartie sur
    132 processus).
Occupation memoire totale pour le systeme          : 515.01 Go (repartie sur
    132 processus).
-------------------------------------------------
Occupation memoire totale                          : 959.67 Go.
  System building : lines J3 : |----------------------| 100 %
  System building : lines J2 : |----------------------| 100 %
  System building : lines J1 : |----------------------| 100 %
  System building : lines div J : |----------------------| 100 %
 System building done
 Solving started
  Solver used : GaussPivotPar_4
  Progress : |----------------------| 100 %
 Solving sucessfull
```

FIGURE C.3 – Suivi des calculs pour la modélisation du câble 5×11.

C.5 Post-traitement

ModeLitz permet de calculer des grandeurs locales comme le potentiel électrique et de la densité de courant (cf. partie 2.3.4), ainsi que le montrent les figures C.4 et C.5. La figure C.6 montre l'orientation des vecteurs densité de courant dans une petite portion de l'inducteur. Il est également possible de visualiser chaque objet de la géométrie indépendamment des autres, ainsi que le montre la figure C.7 (page 163).

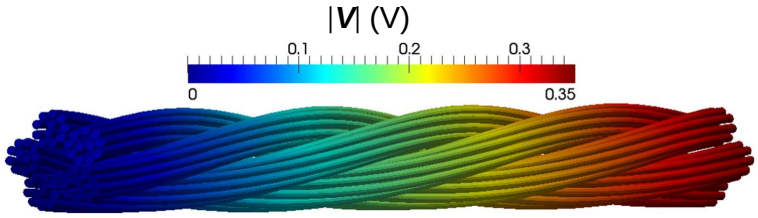


FIGURE C.4 – Répartition du potentiel électrique (en module) au sein de l'inducteur.

Le logiciel permet également de calculer des grandeurs globales. La figure C.8 (page 164) présente le fichier dans lequel sont rapportées les valeurs globales calculées pour chaque objet de la géométrie, en l'occurrence pour chaque brin, soient :

— le courant total complexe $\boldsymbol{I}_b$ parcourant le brin ;

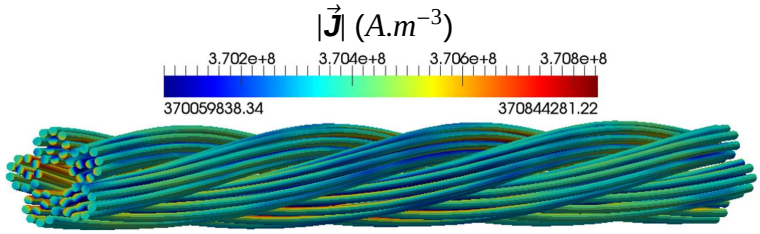


FIGURE C.5 – Répartition de la densité de courant (en module) au sein de l'inducteur.

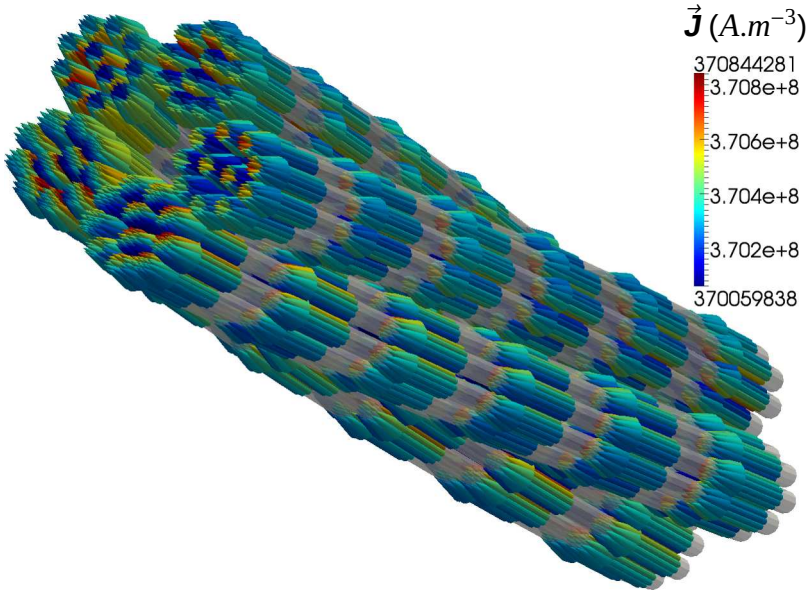


FIGURE C.6 – Répartition et orientation de la partie réelle de la densité de courant dans une portion de l'inducteur 5×11.

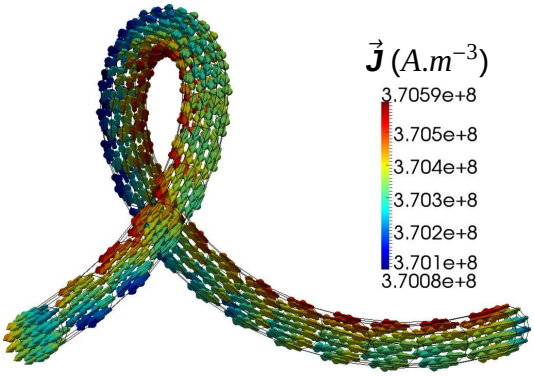


FIGURE C.7 – Répartition et orientation de la partie réelle de la densité de courant dans un brin.

— l'écart relatif (en %) des courants entrant et sortant du brin (indicateur de la précision du calcul) ;
— le courant efficace I_{eff_b} ;
— la puissance Joule perdue $\mathcal{P}_b$;

```
##        Fichier de post-traitement du projet 'vol_cable_5x11_droit_length_0
    .035m_10000Hz'.
##        Post-traitement - Valeurs globales pour tous les objets de la
    geometrie
##        Cree le : Fri May 23 01:52:28 2014

#% Objects number
55

#% | Object |                                                  |
#% |  index |             I/O Average current (A)              |
## |--------|--------------------------------------------------|
#% |    0   | (1.6906016325597992e+00,-2.2034631376324736e+00) |
         .                          .
#% |   54   | (1.7385687050404717e+00,-2.1673747191480754e+00) |
## |--------|--------------------------------------------------|

#% | Object |                            | RMS average |             |
#% |  index | I/O Current deviation (%)  | current (A) | RMS ddp (V) |
## |--------|----------------------------|-------------|-------------|
#% |    0   |  (9.7767e-03,4.8814e-02)   |  1.9638e+00 |  2.4749e-01 |
         .                 .                    .             .
#% |   54   |  (1.8838e-02,5.8026e-02)   |  1.9647e+00 |  2.4749e-01 |
## |--------|----------------------------|-------------|-------------|

#% | Object |                         | Joule power |      Resistance by     |
#% |  index |    Joule power (W)      |  ratio (%)  |    Joule power (Ohm)   |
## |--------|-------------------------|-------------|------------------------|
#% |    0   | 3.0082161897687454e-01 |    1.8185   | 7.7999909981378415e-02 |
         .                .                 .                    .
#% |   54   | 3.0077827972982440e-01 |    1.8182   | 7.7920477725582749e-02 |
## |--------|-------------------------|-------------|------------------------|
```

FIGURE C.8 – Fichier de post-traitement contenant les grandeurs globales calculées pour chaque brin du câble 5×11 (Les symboles · remplacent les brins 1 à 53).

— la fraction de puissance Joule perdue dans le brin (en %), par rapport à la puissance Joule totale perdue dans l'ensemble de l'inducteur ;
— la résistance électrique, calculée à partir de la puissance Joule.

Nous distinguons les brins, qui sont des objets géométriques purs, des inducteurs qui sont des objets composites, car formés par un ensemble de brins. Ainsi, pour un inducteur, le logiciel calcule :

— le courant total complexe $\boldsymbol{I}_{induc}$;
— l'écart relatif (en %) des courants entrant et sortant ;
— le courant efficace ;
— la puissance Joule perdue $\mathcal{P}_{induc}$;
— l'impédance complexe de l'inducteur $\boldsymbol{Z}$;
— la résistance équivalente R_{induc} et l'inductance équivalente L_{induc}.

La figure C.9 présente le fichier dans lequel sont stockées ces grandeurs ainsi que certaines informations sur la structure de l'inducteur.

```
##        Fichier de post-traitement du projet 'vol_cable_5x11_droit_length_0
    .035m_10000Hz'.
##        Post-traitement - Valeurs globales pour tous les objets composites
    de la geometrie
##        Cree le : Fri May 23 01:52:28 2014

#% Objects number
1

## Informations sur les objets :
##    Objet 0 de type INDUCTOR :
##        Frequence : 1.000000e+04 Hz
##        Epaisseur de peau : 6.4991e-04 m
##        Type d'objet : cable (rayon : 0.000594841 m)
##        Nombre de brins : 55
##        Rayon des brins : 5e-05 m
##        Materiaux : copper, (sigma = 5.997e+07 S/m)
##        r/delta = 0.0769337
##        Nombre de divisions curvilignes choisi : 10
##        Type d'arrangement de cable : cable Litz
##        Brin le plus long : 2
##           Longueur : 0.0350959 m
##        Brin le plus court : 7
##           Longueur : 0.0350666 m
##        Description du cable : 5 x 11
##        Rayon interne (cable evide) = 0
##        Parametres (niveau 0 = brin) :
##           Indice de niveau         |   1   |   0   |
##           Nbr. paquets du niveau   |   5   |   11  |
##           Sens de torsadage        |   -1  |   1   |
##           Angle initial de torsade |   0   |   0   |
##           Pas de torsadage         | 0.035 | 0.025 |


#% | Object |                                                  |
#% |  index |             I/O Average current (A)              |
## |--------|--------------------------------------------------|
#% |    0   | (9.4586712325426447e+01,-1.2002227507186208e+02) |
## |--------|--------------------------------------------------|

#% | Object |                            | RMS average |             |
#% |  index | I/O Current deviation (%) | current (A) | RMS ddp (V) |
## |--------|---------------------------|-------------|-------------|
#% |    0   |   (-3.91e-03,-1.47e-03)   |   1.08e+02  |   2.47e-01  |
## |--------|---------------------------|-------------|-------------|

#% | Object |                                                  |
#% |  index |     Resistance (Ohm)    |     Inductance (H)     |
## |--------|-------------------------|------------------------|
#% |    0   |  1.4176669971222270e-03 | 2.8630312292274098e-08 |
## |--------|-------------------------|------------------------|

#% | Object |                           | Joule power |
#% |  index |     Joule power (W)       |  ratio (%)  |
## |--------|---------------------------|-------------|
#% |    0   |  1.6542511620914844e+01   |    100.00   |
## |--------|---------------------------|-------------|

#% | Object |      Resistance by      |  Resistance   |
#% |  index |     Joule power (Ohm)   | deviation (%) |
## |--------|-------------------------|---------------|
#% |    0   | 1.4167965757341874e-03  |    6.14e-02   |
## |--------|-------------------------|---------------|
```

FIGURE C.9 – Fichier de post-traitement contenant les grandeurs globales calculées pour l'inducteur 5×11.

C.6 Suivi des performances

ModeLitz dispose de son propre outil de suivi de ses performances. La taille mémoire utilisée et le temps *CPU* écoulé sont mesurés à différents moments. De plus, le logiciel évalue le temps *CPU* de chaque étape du calculs. La figure C.10 détaille les performances du calcul.

```
##         Performances d'execution du projet 'vol_cable_5x11_droit_length_0
    .035m_10000Hz'.
##         Cree le : Fri May 23 01:52:32 2014
##         Version du logiciel : ModeLitz_1.3.5

#% Numerical parameters
#%   Nodes distribution :
#%     30305 nodes V
#%     3960 nodes J1
#%     18480 nodes J2
#%     36960 nodes J3
#%   Matrix size : 182105
#%   Full lines number : 151800 (83.4 % of the matrix)

#% Computing parameters
#%   Number of process : 132
#%   Seating time : 9.1433530e+05 s or 10d 13h 58min 55.3001s or 253.982 h
#%   Cost time : 1.2069226e+08 s or 3y 10mon 16d 21h 37min 39.6111s
#%   Total memory peak : 975019828 ko
#%   CPU time : 7.2741804e+05 s or 8d 10h 3min 38.0386s


#% Steps number
6

#% Details
#% |                    | User time | User duration | Total memory |
#% |       Steps        |    (s)    |       (s)     |   size (ko)  |
## |--------------------|-----------|---------------|--------------|
#% | Geometry building  |   0.107   |   0.106983    |   1580308    |
#% |       Meshing      |   233.5   |    233.374    |  444657680   |
#% |  System building   | 1.042e+05 |     103941    |  959668276   |
#% |       Solving      | 7.274e+05 |     623211    |  975019828   |
#% |   Posttreatment    | 7.274e+05 |    28.8216    |   7601340    |
## |--------------------|-----------|---------------|--------------|
```

FIGURE C.10 – Rapport des performances des calculs réalisés sur le câble 5×11.

Ce fichier décrit la topologie du système linéaire. Ensuite, le nombre de processus sur lesquels le calcul a été distribué est rapporté. La ligne `Seating time` indique la durée physique du calcul et est à distinguer de la durée *CPU* `CPU time` (voir la partie 2.4.3). La valeur `Cost time` évalue le temps de calcul total et est égale au produit de la durée `Seating time` par le nombre de processus [1]. `Total memory peak` correspond à la consommation de mémoire crête.

1. Nous nous sommes inspirés de la méthode d'évaluation du temps de calcul consommé sur le cluster de calcul Ada de l'IDRIS. `Cost time` correspond donc, dans ce cas, à un temps facturé.

Enfin, le détail de chaque étape du programme (`Steps`) est rapporté dans un tableau. La colonne `User time` indique le temps *CPU* écoulé depuis le lancement du programme jusqu'au moment où l'étape correspondante débute. `User duration` précise la durée *CPU* de l'étape et `Total memory size` indique la quantité totale de mémoire RAM utilisée à la fin de l'étape.

Bibliographie

[1] International Energy Agency, 9 rue de la Fédération, 75739 Paris Cedex 15, France. *Key world energy statistics*, 2013.

[2] Commision Energie 2050. *Rapport Énergies 2050 : les différents scénarios de politique énergétique pour la France*. Gouvernement français, 2012.

[3] La consommation d'énergie dans l'industrie de 1993 à 2009, April 2011.

[4] O. Lucía et al. Induction heating technology and its applications : past developments, current technology, and future challenges. *IEEE Trans. on Indus. Elec.*, 61(5), May 2014.

[5] A. Ducluzaux. *Pertes supplémentaires dans les conducteurs pour forte intensité par effet de peau et de proximité*. Schneider Electric, 2002. cahiers technique n° 83.

[6] A. Reatti and M.K. Kazimierczuk. Comparison of various methods for calculating the ac resistance of inductors. *IEEE Trans. on Magn.*, 44 :1512–1518, May 2002.

[7] P.L. Dowell. Effects of eddy currents in transformer windings. *Proc. Inst. Elect. Eng.*, 113 :1387–1394, Aug. 1966.

[8] J. Schutz, J. Roudet, and A Shellmann. Modeling litz wire windings. In *Industry Applications Conference, 1997. Thirty-Second IAS Annual Meeting, IEEE*, volume 2, pages 190–195, Oct. 1997.

[9] J.A. Ferreira. Analytical computation of ac resistance of round and rectangular litz wire windings. *IEEE Proc. B Electr. Power Appl.*, 139(1) :21–25, Jan. 1992.

[10] J.A. Ferreira. Improved analytical modeling of conductive losses in magnetic components. *IEEE Trans. Power Electron.*, 9 :127–131, Jan. 1994.

[11] R.P. Wojda and M.K. Kazimierczuk. Winding resistance of litz-wire and multi-strand inductors. *IET Power Electron.*, 5(2) :257–268, May 2012.

[12] C.R. Sullivan. Optimal choice for number of strands in a litz-wire transformer winding. *IEEE Trans. on Power Electron.*, 14(2) :283–291, Mar. 1999.

[13] X. Nan and C.R. Sullivan. An improved calculation of proximity-effect loss in high frequency windings of round conductors. In *34th Annual IEEE Power Electronics Specialists Conference*, volume 2, pages 853–860, 2003.

[14] F. Tourkhani and P. Viarouge. Accurate analytical model of winding losses in round litz wire windings. *IEEE Trans. on Magn.*, 37(1) :538–543, Jan. 2001.

[15] J. Acero et al. Domestic induction appliances : an overview of recent research. *IEEE Indus. Appl. Magazine*, 16(2) :39–47, March-April 2010.

[16] J. Acero et al. Simple resistance calculation in litz-wire planar windings for induction heating appliances. *IEEE Trans. on Magn.*, 41(4) :1280–1288, Apr. 2005.

[17] J. Acero et al. Frequency-dependent resistance in litz-wire planar windings for domestic induction heating appliances. *IEEE Trans. on Power Electron.*, 21 :856–866, Jul. 2006.

[18] D.N. Murgatroyd. Calculation of proximity losses in multistranded conductor bunches. *IEEE Prac. A . Phys. Sci. Meas. Instrum. Manage. Educ.*, 136(3) :115–120, May 1989.

[19] A.W. Lotfi and F.C. Lee. A high frequency model for litz wire for switch-mode magnetics. In *IEEE Industry Applications Soc. Annu. Meeting Conf. Rec. 1993*, volume 2, pages 1169–1175, Oct. 1993.

[20] G.W.O. Howe. The high-frequency resistance of multiply-stranded insulated wire. *Proc. Royal Soc. Lon. A. Math. Phys. Sci.*, 93 :468–492, Oct. 1917.

[21] G. Cerri, S.A. Kovyryalov, and V.M. Primiani. Rigorous electromagnetic analysis of domestic induction heating appliances. *PIERS Online*, 5(5) :491–495, 2009.

[22] G. Cerri, S.A. Kovyryalov, and V.M. Primiani. Iet sci. meas. technol. *Modelling of a Litz-wire planar winding geometry for an accurate reactance evaluation*, 4(4) :214–219, 2010.

[23] W. Water and J. Lu. Eddy current and structure optimization of high frequency coaxial transformers using the numerical computation method. In *Electromagnetic Field Problems and Applications (ICEF), 2012 Sixth Int. Conf. on*, June 2012.

[24] W. Water and J. Lu. Simulation of multi-strands conductors thanks to equivalent electric conductivity. In *Proceedings of Numelec 2012*, page 98, July 2012.

[25] C.R. Sullivan. An equivalent complex permeability model for litz-wire windings. *IEEE Trans. on Indus. Appl.*, 45(2) :854–860, March-April 2009.

[26] G. Meunier, A.T. Phung, et al. Propriétés macroscopiques équivalentes pour représenter les pertes dans les bobines conductrices. *European Jour. of Elec. Eng.*, 11(6) :675–694, 2008.

[27] M. Etemadrezaei and S.M. Lukic. Equivalent complex permeability and conductivity of litz wire in wireless power transfer systems. In *Energy Conversion Congress and Exposition (ECCE), 2012 IEEE*, pages 3833–3840, Sept. 2012.

[28] P.B. Reddy, T.M. Jahns, and T.P. Bohn. Transposition effects on bundle proximity losses in high-speed pm machines. In *Energy Conversion Congress and Exposition, 2009. ECCE 2009. IEEE*, pages 1919–1926, Sept. 2009.

[29] V. Väisänen et al. Ac resistance calculation methods and practical design considerations when using litz wire. In *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pages 368–375, Nov. 2013.

[30] M. Bartoli et al. Modeling litz-wire winding losses in high-frequency power inductors. In *Power Electronics Specialists Conference, PESC, 27th Annual IEEE*, volume 2, pages 1690–1696, June 1996.

[31] L. Ignacio et al. Ac power losses model for planar windings with rectangular cross-sectional conductors. *IEEE Trans. on Power Electron.*, 29(1) :23–28, Jan. 2014.

[32] H. Hämäläinen et al. Ac resistance factor of litz-wire windings used in low-voltage high-power generators. *IEEE Trans. on Indus. Electron.*, 61(2) :693–700, Feb. 2014.

[33] A. Stadler et al. Analytical calculation of copper losses in litz-wire windings of gapped inductors. *IEEE Trans. on Magn.*, 50(2), Feb. 2014.

[34] A. Roßkopf, E. Bär, and C. Joff. Influence of inner skin- and proximity effects on conduction in litz wires. *IEEE Trans. on Power Electron.*, 29(10) :5454–5461, Oct. 2014.

[35] C.R. Sullivan and R.Y. Zhang. Simplified design method for litz wire. In *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*, pages 2667–2674, March 2014.

[36] R.Y. Zhang et al. Realistic litz wire characterization using fast numerical simulations. In *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*, pages 738–745, March 2014.

[37] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012. Pages 9-57.

[38] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012. Page 24.

[39] D. Etiemble. *Évolution de l'architecture des ordinateurs*. Dossier Techniques de l'Ingénieur. Ed. Techniques Ingénieur, Fév. 2009.

[40] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012. Page 25.

[41] D. Koufaty and DT. Marr. Hyperthreading technology in the netburst microarchitecture. *IEEE Micro*, 23(2) :56–65, March-April 2003.

[42] S. Casey. How to determine the effectiveness of hyper-threading technology with an application. *Intel Technology Journal*, 6(1) :11, April 2011.

[43] Projet TOP500, www.top500.org.

[44] Présentation du *benchmark* LINPACK, www.top500.org.

[45] Présentation de l'*APU* Llano, www.pcworld.fr.

[46] Sortie des processeurs Haswell, www.pcworld.fr.

[47] Présentation des *APU* Kaveri, www.cnetfrance.fr.

[48] Configuration matérielle du calculateur Tianhe-2, www.spectrum.ieee.org.

[49] Le supercalculateur Tianhe-2, www.china.org.cn.

[50] Le supercalculateur Titan, www.zdnet.fr.

[51] La plate-forme BOINC, www.boinc.berkeley.edu.

[52] Statistiques de la plate-forme BOINC, http://boincstats.com.

[53] Performances de la plate-forme BOINC, www.boinc-af.org.

[54] I. Foster and C. Kesselman. *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[55] Le projet WLCG, www.wlcg.web.cern.ch.

[56] F. Malek. Le calcul scientifique des expériences lhc - une grille de production mondiale. *Reflets phys.*, 20 :11–15, Juil. Aug. 2010.

[57] M.J. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, C-21 :948–960, Sept. 1972.

[58] M.J. Flynn and K.W. Rudd. Parallel architectures. *ACM Computing Surveys*, 28(1) :67–70, March 1996.

[59] J-P. Cappello, F. et Sansonnet. *Introduction au parallélisme et aux architectures parallèles*. Dossier Techniques de l'Ingénieur. Ed. Techniques Ingénieur, Août 1999.

[60] J-P. Sansonnet. *Architecture des ordinateurs parallèles*. Dossier Techniques de l'Ingénieur. Ed. Techniques Ingénieur, Déc. 1992.

[61] Pipeline, www.cs.waikato.ac.nz.

[62] La technologie CUDA, www.nvidia.fr.

[63] AMD. *Computer Abstraction Layer (CAL)*. Advanced Micro Device Inc., Déc. 2010.

[64] Khronos Group, www.khronos.org.

[65] A. Munsh. *The OpenCL Specification*. The Khronos Group Inc., Jan. 2011.

[66] Le standard OpenMP, www.openmp.org.

[67] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012. Pages 37-39.

[68] J.J. Dongarra et al. *A proposal for a user-level, message passing interface in a distributed memory environment*. Oak Ridge National Laboratory, Tennessee 37831, Feb. 1993. Technical Report TM-12231.

[69] Message Passing Interface Forum. *MPI : A Message-Passing Interface Standard, Version 3.0*. University of Tennessee, Sept. 2012.

[70] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012. Pages 42-47, 98, 99.

[71] L'implémentation OpenMPI, www.open-mpi.org.

[72] L'implémentation MPICH2, www.mcs.anl.gov.

[73] Modelitz®, March 2014. Agence pour la Protection des Programmes (APP), www.app.asso.fr, N° : IDDN.FR.001.120006.000.S.P.2014.000.30645.

[74] A. Gagnoud. Three-dimensional integral method for modelling electromagnetic inductive processes. *IEEE Trans. on Magn.*, 40(1) :29–36, Jan. 2004.

[75] V. Labbé. *Modélisation numérique du chauffage par induction – Approche éléments finis et calcul parallèle*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, Apr. 2002.

[76] Paraview®, Oct. 2012. Edited by Kitware. Available on www.paraview.org.

[77] G. Dhatt and Touzot G. *Une présentation de la méthode des éléments finis*. Maloine S.A., Paris, Les presses de l'Université de Laval, Québec, 1981.

[78] M. Al-Towaiq. Clustered gauss-huard algorithm for the solution of ax=b. *Appl. Math. Comput.*, 184(2) :485–495, Jan. 2007.

[79] Migen®, 2011. Agence pour la Protection des Programmes (APP), www.app.asso.fr, N° : IDDN.FR.001.440008.000.S.C.2011.000.30645, Available on www.gravit-innovation.org.

[80] J. Clet-Ortega. *Exploitation efficace des architectures parallèles de type grappes de NUMA à l'aide de modèles hybrides de programmation*. PhD thesis, Ecole Doctorale de Mathématiques et Informatique, Université de Bordeaux 1, Apr. 2012.

[81] R. Scapolan, A. Gagnoud, and Y. Du Terrail. 3d multi-strands inductor modeling : influence of complex geometry arrangements. *IEEE Trans. on Magn.*, 50(2), Feb. 2014.

[82] I. Stefanini. *Méthodologie de conception et optimisation d'actionneurs intégrés sans fer*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, Apr. 2012.

Résumé : Afin de permettre l'utilisation de hautes fréquences dans le domaine du chauffage par induction industriel, l'emploi d'inducteurs multibrins est envisagé. Or, les pertes occasionnées dans ces inducteurs peuvent être importantes et dépendent fortement de leur géométrie interne qui est complexe. Pour faciliter la conception d'inducteurs multibrins à faibles pertes, il est nécessaire d'en comprendre le comportement électromagnétique. Dans cette thèse, nous présentons le développement d'un logiciel de calcul parallèle dévolu à la modélisation électromagnétique 3D d'inducteurs multibrins. Nous décrivons une méthode originale de construction de la géométrie des inducteurs. Ce logiciel est basé sur une méthode numérique de type intégrale ayant l'avantage de ne pas nécessiter le maillage des espaces entre les brins. L'emploi du calcul parallèle est une des grandes forces de ce logiciel. Les études réalisées montrent l'impact de la géométrie sur le comportement de ce type d'inducteur.

Mots-clé : modélisation numérique, fils de Litz, méthode intégrale, chauffage par induction, calcul parallèle, efficacité énergétique

Abstract : *In order to enable to use high frequencies in the domain of the industrial inductive heating, the use of multi-wires inductors is considered. But, losses occurring into that inductors can be important and strongly depend on their complex internal geometry. To facilitate the design of lossless multi-wires inductors, it is necessary to under stand their electromagnetic behavior. In this thesis, we present the development of a software of parallel computation intended to the 3D electromagnetic modeling of multi-wires inductors. We describe an original method of building of the geometry of that inductors. This software is based on an integral method in which the meshing of spaces between the wires is unnecessary. The use of parallel computing is one of the great forces of this software. The studies we realized show the impact of the geometry on the behavior of that type of inductor.*

Keywords : numeric modeling, Litz wires, integral method, heating induction, parallel computing, energy efficiency