UNIVERSITÉ DE STRASBOURG

ÉCOLE DOCTORALE 269
Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII)

UMR 7501
Institut de Recherche Mathématique Avancée (IRMA)

THÈSE présentée par :

Bruno WEBER

soutenue le : **26 novembre 2018**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**
Discipline / Spécialité : **Mathématiques appliquées**

Optimisation de code Galerkin Discontinu sur ordinateur hybride. Application à la simulation numérique en électromagnétisme.

THÈSE dirigée par :
M **HELLUY Philippe** Professeur, Université de Strasbourg

RAPPORTEURS :
Mme **BARUCQ Hélène** Directeur de recherche, INRIA Bordeaux
M **FERRIERES Xavier** Maître de recherche, ONERA Toulouse

EXAMINATEUR :
M **PRUD'HOMME Christophe** Professeur, Université de Strasbourg

ENCADRANT INDUSTRIEL (CIFRE) :
M **GIRAUDON Cyril** AxesSim, Illkirch-Graffenstaden

Je dédie cette thèse à Georgette, ma mère, et Gérard, mon père, qui m'ont permis d'arriver jusque là.

Merci.

Remerciements

Je tiens tout d'abord à remercier Philippe Helluy, directeur de cette thèse, qui a toujours été présent pour répondre à mes questions, qui m'a soutenu au cours de ces derniers longs mois de rédaction et sans qui je n'aurais pas eu l'opportunité de réaliser ces intéressants travaux de recherche et d'optimisations.

Dans un second temps, je remercie Christophe Girard, PDG de la société AxesSim, qui a permis à cette thèse de voir le jour au format CIFRE. Je reconnais par la même ses qualités telles que la souplesse dont il fait preuve dans la gestion de l'entreprise ainsi que son engagement dans le développement de la méthode Galerkin Discontinue (GD).

Mes remerciements s'adressent aussi à Hélène Barucq et Xavier Ferrières qui ont accepté de rapporter cette thèse. Je remercie également Christophe Prud'homme d'avoir accepté de faire partie du jury en tant qu'examinateur.

Le bon déroulement de cette thèse est aussi dû aux bons conseils, à l'expérience et l'aide apportés par toute l'équipe AxesSim. Je remercie notamment Cyril Giraudon pour le partage de ses vastes connaissances, Thomas Strub pour m'avoir épaulé à mes débuts sur le code de calcul GD, Nathanaël Muot pour sa vigilance quant au respect des délais, Didier Roissé pour sa passion de la compilation, Philippe Spinosa pour sa patience envers les *noobs*, Guillaume Prin et Xavier Romeuf pour leur réactivité et Sophie Marin pour son efficacité et sa bonne humeur.

Du côté universitaire, je remercie particulièrement Malcolm Roberts avec qui j'ai assisté à mes premières conférences ainsi que tout le personnel administratif, autant de l'IRMA que de l'école doctorale, pour m'avoir facilité les démarches administratives.

Merci aussi à tous mes proches, famille, amis, amie, de m'avoir accompagné tout au long de cette thèse (pour certains bien avant déjà), de s'être intéressé à mes travaux et de m'avoir écouté (me plaindre, parfois).

Enfin, je vous remercie par avance, vous, lecteurs, d'avoir ouvert cette thèse et vous souhaite une bonne lecture.

Table des matières

Introduction

Cette thèse a été réalisée dans le cadre du projet HOROCH [1] au sein de l'entreprise AxesSim, en collaboration avec Thales Communications & Security, les centres de recherche IRMA Strasbourg et ONERA Toulouse, les PME Cityzen Sciences et BodyCap, et financé par la DGE [2].

L'objectif de ce projet était de développer, en parallèle, des solutions de simulation et de mesure du rayonnement électromagnétique des objets connectés placés à proximité (ou à l'intérieur) du corps humain, afin de fournir des outils avancés de conception d'objets connectés (IoT, pour « *Internet of Things* ») aux PME de ce secteur d'activité. Les domaines d'application sont nombreux : téléphones mobiles, montres connectées, maillots connectés, sondes de mesure médicales (gélules ou patchs), *etc.*

Ainsi, TCS avait en charge la conception d'un banc de mesure sous la forme d'une arche composée de 32 récepteurs Vivaldi. AxesSim avait en charge l'évolution de son solveur Galerkin Discontinu (GD) `teta`. L'IRMA, en collaboration avec AxesSim au travers de thèses CIFRE, est en charge du solveur GD générique `clac` (pour « *Conservation Laws on mAny Cores* »), brique de base du solveur `teta`, spécialisé pour des applications en l'électromagnétisme.

Pour AxesSim, l'objectif final de ce projet, et par extension de cette thèse, était d'effectuer une simulation d'une antenne BLE (pour « *Bluetooth Low Energy* ») placée à proximité d'un modèle de corps humain complet (avec squelette et organes). Ce modèle a été obtenu par scan à rayons X, effectué par l'IRCAD de Strasbourg d'un mannequin anthropomorphique approvisionné par TCS.

Le maillage non structuré du mannequin complet comporte environ 24 millions de mailles hexaédriques pour des applications entre 1 et 3 GHz, bande de fréquence des IoT. Les antennes utilisées ont généralement une

1. HOROCH : utilisation des HPC pour l'Optimisation des Radiocommunications des Objets Connectés proches de l'Homme
2. Direction Générale des Entreprises

taille de l'ordre de grandeur de la longueur d'onde du signal émis (soit 1-10 cm dans notre cas), avec des adaptations ou des composants d'un voire deux ordres de grandeur inférieurs en taille.

Dans le cas de cette étude, le facteur d'échelle entre la plus petite maille (antenne) et la plus grande maille (vide englobant) est de l'ordre de 1000. Ces variations dans la taille des mailles en fait un cas d'application bien adapté à la méthode GD, contrairement à la méthode des différences finies, plus fréquemment utilisée, qui nécessiterait de mailler toute la scène de manière structurée tout en respectant au mieux la géométrie des plus petits détails de l'antenne.

La première simulation sur corps humain complet a été effectuée en 11 jours sur une machine de calcul composée de 8 GPU NVidia de dernière génération pour seulement 10 ns de temps physique. Cette simulation a été exécutée avec un faible ordre d'interpolation et a nécessité plus de 40 Go de mémoire vive. Afin de permettre au solveur `teta-clac` de traiter des simulations de cette taille, un grand nombre d'optimisations et d'améliorations algorithmiques ont été nécessaires. Nous allons présenter les travaux qui ont été menés afin d'atteindre ces performances.

Dans le chapitre 1, nous commençons par rappeler la formulation des équations de Maxwell. Ce système de 4 équations, auxquelles nous ajoutons l'équation de conservation de la charge, régit la propagation des ondes électromagnétiques et nous permettra de les modéliser par la simulation numérique.

Nous écrivons ensuite ces équations sous la forme générique d'un système hyperbolique symétrique linéaire du premier ordre, aussi appelé système de Friedrichs. De nombreuses propriétés sont connues pour ces systèmes d'équations aux dérivées partielles. Nous examinons ensuite les possibles discontinuités des solutions de ce système et donnons les conditions d'existence et d'unicité d'une telle solution.

Enfin, nous introduisons le schéma GD qui permet d'approcher le système hyperbolique. Le schéma GD est caractérisé par une formulation faible avec des fonctions test discontinues et un flux numérique sur les discontinuités. Nous listons alors quelques flux numériques permettant d'assurer la stabilité du schéma. Nous donnons aussi une formulation semi-discrète du problème qui a servi à l'implémentation informatique du solveur générique `clac`. Des propriétés de convergence mathématique du problème semi-discret sont rappelées en annexe B.

Dans le chapitre 2, nous présentons les principaux modèles mathématiques nécessaires à la résolution des problèmes qui nous intéressent. Ces modèles sont implémentés dans la surcouche applicative `teta` développée par

AxesSim, spécialisation à l'électromagnétisme du solveur générique `clac`.

Nous commençons par décrire des conditions aux limites respectant les conditions d'unicité de la solution. Nous donnons aussi le détail de la construction du modèle de couches absorbantes PML (pour « *Perfectly Matched Layer* ») permettant de résoudre un problème en domaine infini simulé.

Nous présentons ensuite des techniques numériques pour représenter de fines plaques de matériaux. La suppression géométrique de ces plaques minces permet de simplifier la modélisation de la scène étudiée et diminue les temps de calcul du solveur. L'un de ces modèles de plaque mince sera également étendu en un modèle permettant l'injection de tensions ou de courants électriques utilisés dans la simulation de dispositifs rayonnants.

Enfin, dans le but de traiter les matériaux diélectriques dont la permittivité varie en fonction de la fréquence, nous décrivons aussi un modèle volumique prenant en compte ces variations. Ce type de modèle est intéressant dans le cas de simulations faisant intervenir le corps humain qui présente ce type de matériaux.

Dans le chapitre 3, nous décrivons l'implémentation générique du solveur `clac` programmé pour traiter les mailles hexaédriques à arêtes droites. La méthode GD peut théoriquement être appliquée à tout type de maillage, notamment sur mailles tétraédriques. Les hexaèdres permettent diverses optimisations de la formulation GD et sont bien adaptés à la parallélisation sur processeur graphique.

Nous commençons par introduire l'espace d'approximation dans lequel nous chercherons la solution numérique de notre problème. Cet espace est engendré par des fonctions de base à support inclus dans une unique maille. Dans le cas du solveur `teta-clac` nous utilisons une base de polynômes constituée des produits tensoriels des polynômes d'interpolation de Lagrange, eux-mêmes générés à partir des points d'intégration de Gauss-Legendre.

Nous présentons aussi les schémas d'intégration en temps actuellement les plus souvent utilisés par le solveur : les schémas de type Runge-Kutta (RK). Ces schémas explicites imposent une condition de type CFL afin d'assurer la stabilité du schéma couplé espace-temps (le plus souvent GD-RK2). Nous donnons la formulation des 2 conditions CFL utilisées pour les cas d'application du dernier chapitre ainsi que 2 diagnostics de stabilité.

Dans le chapitre 4, nous décrivons l'implémentation parallèle sur (un ou) plusieurs accélérateurs de la méthode GD discrète décrite au chapitre précédent. Pour cela, nous utilisons la bibliothèque OpenCL (pour « *Open Computing Language* ») qui permet d'exploiter la puissance de calcul de ces accélérateurs. Les cartes graphiques (GPU) récentes permettent de réaliser des

calculs de façon massivement parallèle, et sont de ce fait très utilisées dans la simulation numérique. L'implémentation GPU a initialement été réalisée en C++ en utilisant la bibliothèque OpenCL au cours de la thèse de T. Strub [69], à partir du code non parallèle FemGD développé à l'ONERA de Toulouse.

Afin de pouvoir traiter des volumes de données importants, la méthode GD est également parallélisée sur plusieurs accélérateurs à l'aide d'une implémentation du standard MPI (« *Message Passing Interface* »). Dans notre cas, MPI offre la possibilité de lancer plusieurs processus qui effectuent les calculs sur différentes parties du maillage et communiquent par un système de messages.

Cette implémentation GPU est aussi exploitable sur un processeur multi-cœurs (CPU) plus classique. Ces derniers ont beaucoup évolué au cours des années passées pour être aujourd'hui composés de plusieurs dizaines de cœurs logiques cadencés à des fréquences supérieures à celles des GPU. Cependant, de par leur architecture radicalement différente de celle des GPU, une implémentation spécifique aux CPU doit être développée afin de les exploiter au mieux.

Nous présentons aussi une méthode d'adaptation de l'ordre d'interpolation spatiale appliquée aux mailles du maillage dans le but de réduire le temps de simulation en optimisant l'échantillonnage du domaine de calcul.

Dans le chapitre 5, nous présentons une adaptation algorithmique permettant d'améliorer encore la vitesse de calcul du solveur : un schéma à pas de temps local par maille. Ce schéma permet de réduire la quantité de calculs et ne nécessite donc pas d'accélérateur supplémentaire pour obtenir de meilleures performances. Cette méthode d'intégration est théoriquement très efficace mais n'est *a priori* adaptée qu'aux maillages présentant un facteur d'échelle important entre la plus petite et les plus grandes mailles.

Nous commençons par introduire une variante locale du schéma RK d'ordre 2 (RK2) que nous appelons LRK2 (pour « *Local* RK2 »). Nous utilisons ensuite les propriétés locales du schéma LRK2 afin de découpler l'avancée en temps de l'approximation de la solution sur les différentes mailles. Ainsi, ce second schéma, que nous appelons LTS2 (pour « *Local Time Step* 2 »), permet de réduire considérablement le temps de simulation en divisant la quantité de calculs lorsque la géométrie est composée de mailles de tailles très hétérogènes.

Dans le chapitre 6, nous montrons dans quelle mesure un solveur GD s'adapte à l'exécution sur ordinateur hybride. Les (super)calculateurs actuels sont généralement un regroupement de nœuds de calcul identiques, compo-

sés d'un (ou plusieurs) CPU et éventuellement d'un GPU. L'équilibrage de la charge de calcul est donc aisé entre périphériques de même nature. Cependant, les CPU sont très différents des GPU, tant au niveau de l'architecture que de la puissance de calcul. Ainsi, afin d'être capable d'utiliser toute la puissance de calcul d'un nœud hybride, il est très intéressant de savoir équilibrer la charge ce calcul entre un CPU et un GPU.

À ces fins, nous avons utilisé la bibliothèque StarPU développée à l'INRIA de Bordeaux. Cette bibliothèque permet de se défaire de l'équilibrage manuel des tâches via un graphe des tâches. Ce dernier est automatiquement généré par StarPU qui utilise la dépendance des données passées en entrée et en sortie des tâches. Les tâches sont alors soumises aux périphériques automatiquement et les données sont transférées en fonction du besoin.

Les expérimentations de l'utilisation de cette bibliothèque ont été effectuées sur le solveur universitaire `schnaps` et sont présentées sous la forme d'un article qui sera prochainement soumis.

Enfin, dans le chapitre 7, nous donnons un ensemble de résultats validant notre implémentation de solveur GD, autant par la précision des calculs que par leur vitesse d'exécution.

Nous commençons par présenter des résultats de convergence sur un cas académique de propagation d'une onde plane dans un cube. Nous en profitons pour effectuer des comparaisons entre maillages structurés (hexaèdres droits) et non structurés (issus du découpage de tétraèdres).

Nous comparons ensuite notre solveur GD à un solveur implémentant la méthode des différences finies sur un cas d'application mettant en scène une antenne dipolaire à proximité de l'oreille d'une tête humaine simplifiée. Ce cas d'application est un modèle simplifié de rayonnement d'un téléphone portable. Nous donnons aussi les résultats, sur ce même cas d'application, des différents schémas d'intégration en temps que nous avons évoqués plus tôt.

Nous présentons ensuite des calculs réalisés sur un modèle de corps humain complet, à côté duquel rayonne une antenne BLE. Il s'agit de la plus grande simulation réalisée par le solveur à ce jour. Ce calcul a été réalisé sur un calculateur monté par AxesSim et disposant de 8 cartes graphiques.

Enfin, dans le cadre d'un appel à projets PRACE, nous avons pu avoir accès au supercalculateur PizDaint. Cela nous a permis, sur le cas d'application du corps humain complet, de réaliser un test de scalabilité forte du code GD sur plusieurs centaines de GPU.

Chapitre 1

Généralités

Dans ce premier chapitre, nous donnons la formulation des équations de Maxwell. Ces équations régissent la propagation des ondes électromagnétiques et nous permettront de la modéliser par la simulation numérique.

Nous écrivons ensuite ces équations sous la forme d'un système hyperbolique symétrique linéaire du premier ordre, aussi appelé système de Friedrichs, pour lesquels de nombreuses propriétés sont connues. Nous examinons ensuite les possibles discontinuités des solutions de ce système et donnons les conditions d'existence et d'unicité d'une telle solution.

Enfin, nous introduisons la formulation Galerkin Discontinue (GD) du système hyperbolique ainsi obtenu, en passant par la formulation faible du problème d'évolution étudié. Nous listons alors quelques flux numériques permettant d'améliorer la stabilité du schéma et terminons par donner la formulation semi-discrète du problème.

1.1 Équations de Maxwell

Les équations de Maxwell décrivent la propagation des champs électromagnétiques dans un milieu quelconque. À la fin du XIX$^{\text{ème}}$ siècle, le physicien James Clerk Maxwell s'est basé sur les travaux existants concernant l'électricité et le magnétisme afin de les unifier en un système de huit équations. Plus tard, Olivier Heaviside les reformula sous forme de quatre équations aux dérivées partielles que nous présentons ci-après.

Nous notons $x = (x_1, x_2, x_3)$ un point de l'espace $\mathbb{R}^3$ et ∂_i la dérivée partielle suivant x_i. De plus, nous notons $\nabla = (\partial_1, \partial_2, \partial_3)^{\mathrm{T}}$ l'opérateur gradient. Les opérateurs rotationnel et divergence sont respectivement notés $\nabla\times$ et $\nabla\cdot$,

où $\times$ représente le produit vectoriel et $\cdot$ le produit scalaire usuel. La dérivée partielle en temps est notée ∂_t.

Soient $\boldsymbol{E}(x,t) \in \mathbb{R}^3$ le champ électrique, $\boldsymbol{H}(x,t) \in \mathbb{R}^3$ le champ magnétique, $\boldsymbol{J}(x,t) \in \mathbb{R}^3$ le courant électrique et $\rho(x,t) \in \mathbb{R}$ la densité de charge électrique. Soient ε, μ, σ et $\sigma^\star$ les paramètres constitutifs des matériaux, représentant respectivement la permittivité, la perméabilité et les conductivités électrique et magnétique du milieu. Dans le cas des matériaux linéaires, homogènes, isotropes, et avec réponse instantanée aux changements du champ électrique, ces paramètres sont des constantes propres au matériau.

Avec ces notations et en tenant compte des équations constitutives des milieux linéaires, les équations de Maxwell s'écrivent :

$$\partial_t \varepsilon \boldsymbol{E} + \sigma \boldsymbol{E} - \nabla \times \boldsymbol{H} = -\boldsymbol{J}, \tag{1.1a}$$
$$\partial_t \mu \boldsymbol{H} + \sigma^\star \boldsymbol{H} + \nabla \times \boldsymbol{E} = 0, \tag{1.1b}$$
$$\nabla \cdot \varepsilon \boldsymbol{E} = \rho, \tag{1.1c}$$
$$\nabla \cdot \mu \boldsymbol{H} = 0, \tag{1.1d}$$
$$\partial_t \rho + \nabla \cdot (\boldsymbol{J} + \sigma \boldsymbol{E}) = 0. \tag{1.1e}$$

L'équation (1.1a) est appelée équation de Maxwell-Ampère. Cette équation énonce qu'une variation du champ électrique ou la présence d'un courant électrique génèrent un champ magnétique. L'équation (1.1b) est appelée équation de Maxwell-Faraday. Cette seconde équation indique que la variation d'un champ magnétique génère un champ électrique.

À ces deux équations s'ajoutent des conditions sur les divergences des champs électromagnétiques exprimées par les équations de Maxwell-Gauss (1.1c) et Maxwell-Thomson (1.1d). Ces équations précisent que le flux électrique sortant d'un volume est lié à la charge électrique contenue dans ce volume et que le flux magnétique à travers une surface fermée est nul.

Enfin, la dernière équation (1.1e) donne la propriété de conservation de la charge électrique.

Proposition 1. *Si les équations de Maxwell-Gauss (1.1c) et Maxwell-Thomson (1.1d) sont vérifiées à l'instant initial pour une solution des équations de Maxwell, alors elles sont vérifiées à chaque instant.*

Démonstration. En appliquant l'opérateur divergence aux équations de Maxwell-Ampère (1.1a) et Maxwell-Faraday (1.1b), par le fait que la di-

vergence d'un rotationnel est nulle, nous obtenons :

$$\partial_t \nabla \cdot \varepsilon \boldsymbol{E} = -\nabla \cdot (\boldsymbol{J} + \sigma \boldsymbol{E}), \tag{1.2a}$$
$$\partial_t \nabla \cdot \mu \boldsymbol{H} = -\nabla \cdot \sigma^\star \boldsymbol{H}. \tag{1.2b}$$

Nous utilisons ensuite l'équation de conservation de la charge (1.1e) sur la première équation ainsi obtenue (1.2a) et l'équation de Maxwell-Thomson (1.1d) sur la seconde équation (1.2b) :

$$\partial_t \nabla \cdot \varepsilon \boldsymbol{E} = \partial_t \rho, \tag{1.3a}$$
$$\partial_t \nabla \cdot \mu \boldsymbol{H} = 0. \tag{1.3b}$$

Enfin, en appliquant les conditions initiales données par les équations de Maxwell-Gauss (1.1c) et Maxwell-Thomson (1.1d) :

$$\nabla \cdot \varepsilon \boldsymbol{E}(x, 0) = \rho(x, 0), \tag{1.4a}$$
$$\nabla \cdot \mu \boldsymbol{H}(x, 0) = 0, \tag{1.4b}$$

nous obtenons le résultat souhaité. □

Nous pouvons donc nous contenter de ne considérer que les équations de Maxwell-Ampère et Maxwell-Faraday dans la suite, en gardant à l'esprit que les équations de Maxwell-Gauss et Maxwell-Thomson doivent être vérifiées à l'instant initial du problème considéré.

1.2 Système de Friedrichs

Nous venons de justifier le fait que, en principe, il suffit de résoudre les équations de Maxwell-Ampère (1.1a) et Maxwell-Faraday (1.1b). Dans la suite, nous appellerons le système constitué de ces deux équations « les équations de Maxwell ». Nous pouvons écrire ces équations sous la forme plus générique d'un système dit de Friedrichs [17, 20, 24] afin de bénéficier des propriétés connues sur ces derniers.

1.2.1 Définitions

Définition 1. *Soit $\boldsymbol{w}(x, t)$ un vecteur de m fonctions solution du système d'équations aux dérivées partielles suivant :*

$$\partial_t \boldsymbol{A}_t \boldsymbol{w} + \boldsymbol{C}\boldsymbol{w} + \sum_{i=1}^{3} \boldsymbol{A}_i \partial_i \boldsymbol{w} = \boldsymbol{S} \tag{1.5}$$

avec $\boldsymbol{A}_t$, $\boldsymbol{C}$ *et* $(\boldsymbol{A}_i)_{i\in[\![1\,;3]\!]}$ *des matrices carrées réelles de taille* m *et* $\boldsymbol{S}$ *un vecteur de* m *fonctions sources. Ce système est dit* ***de Friedrichs*** *si pour toute direction* $\boldsymbol{n} = (n_1, n_2, n_3)^{\mathrm{T}}$ *de* $\mathbb{R}^3$ *la matrice* $\boldsymbol{A} = \sum_{i=1}^{3} \boldsymbol{A}_i n_i$ *est symétrique et si la matrice* $\boldsymbol{A}_t$ *est symétrique et définie positive.*

Notation 1. *Dans la suite nous utiliserons la convention de sommation des indices répétés, i.e.* $\sum_{i=1}^{3} \boldsymbol{A}_i \partial_i = \boldsymbol{A}_i \partial_i$.

Dans le cas des équations de Maxwell, le vecteur $\boldsymbol{w} = (\boldsymbol{E}^{\mathrm{T}}, \boldsymbol{H}^{\mathrm{T}})^{\mathrm{T}}$ est le vecteur des champs électromagnétiques avec $m = 6$. Les matrices $\boldsymbol{A}_t$, $\boldsymbol{C}$ et $(\boldsymbol{A}_i)_{i\in[\![1\,;3]\!]}$ et le vecteur $\boldsymbol{S}$ se déduisent des équations de Maxwell et valent :

$$\boldsymbol{A}_t = \begin{pmatrix} \varepsilon & 0 & 0 & 0 & 0 & 0 \\ 0 & \varepsilon & 0 & 0 & 0 & 0 \\ 0 & 0 & \varepsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{pmatrix}, \tag{1.6a}$$

$$\boldsymbol{C} = \begin{pmatrix} \sigma & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^\star & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^\star & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^\star \end{pmatrix}, \tag{1.6b}$$

$$\boldsymbol{A}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{1.6c}$$

$$\boldsymbol{A}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{1.6d}$$

$$\boldsymbol{A}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{1.6e}$$

$$\boldsymbol{S} = \left(-\boldsymbol{J}^{\mathrm{T}}, 0_{\mathbb{R}^3}^{\mathrm{T}}\right)^{\mathrm{T}}. \tag{1.6f}$$

Nous constatons que la matrice $\boldsymbol{A}_t$ est symétrique définie positive et que les matrices $\boldsymbol{A}_i$ sont symétriques. Nous sommes donc bien en présence d'un système de Friedrichs.

La matrice $\boldsymbol{A}_i n_i$ de la définition 1 est donnée par :

$$\boldsymbol{A}_i n_i = \begin{pmatrix} 0 & -\boldsymbol{n}\times \\ \boldsymbol{n}\times & 0 \end{pmatrix}, \text{ avec } \boldsymbol{n}\times = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}. \tag{1.7}$$

Les valeurs propres de cette matrice sont $\|\boldsymbol{n}\|^2$, 0 et $-\|\boldsymbol{n}\|^2$, chacune de multiplicité 2.

Nous pouvons aussi énoncer la définition plus générale d'un système hyperbolique dans $\mathbb{R}^3$:

Définition 2. *Soit $\boldsymbol{u}(x,t) = (u_1, \ldots, u_m)^{\mathrm{T}}$ un vecteur de m fonctions solution du système d'équations aux dérivées partielles suivant :*

$$\partial_t \boldsymbol{u} + \partial_i f^i(\boldsymbol{u}) = 0 \tag{1.8}$$

*avec $f^i \in \mathcal{C}^1(\mathbb{R}^m, \mathbb{R}^m)$ des fonctions continûment dérivables non nécessairement linéaires. Soient aussi les matrices $\boldsymbol{A}_i = (\partial_{u_k} f^i_j)_{j,k\in[\![1\,;m]\!]}$. Ce système est dit **hyperbolique** si pour toute direction $\boldsymbol{n}$ de $\mathbb{R}^3$ la matrice $\boldsymbol{A} = \boldsymbol{A}_i n_i$ est diagonalisable dans $\mathbb{R}$.*

Proposition 2. *Un système de Friedrichs est hyperbolique.*

Démonstration. Un système de Friedrichs est hyperbolique si pour toute direction $\boldsymbol{n}$ de $\mathbb{R}^3$ la matrice $\boldsymbol{A}_t^{-1}\boldsymbol{A}_i n_i$ est diagonalisable dans $\mathbb{R}$. Or, la matrice $\boldsymbol{A}_t$ est diagonale définie positive et la matrice $\boldsymbol{A}_i n_i$ est symétrique et donc diagonalisable dans $\mathbb{R}$. Il est donc aisé de démontrer que le produit $\boldsymbol{A}_t^{-1}\boldsymbol{A}_i n_i$ est diagonalisable et à valeurs propres réelles. Le système étudié est donc un système hyperbolique. □

1.2.2 Condition de saut

Nous allons maintenant examiner les possibles discontinuités apparaissant dans les solutions du système hyperbolique (1.5). Les discontinuités apparaissant dans une solution d'un tel système vérifient les conditions de saut de Rankine-Hugoniot [63]. Appliquons ces conditions aux équations de Maxwell.

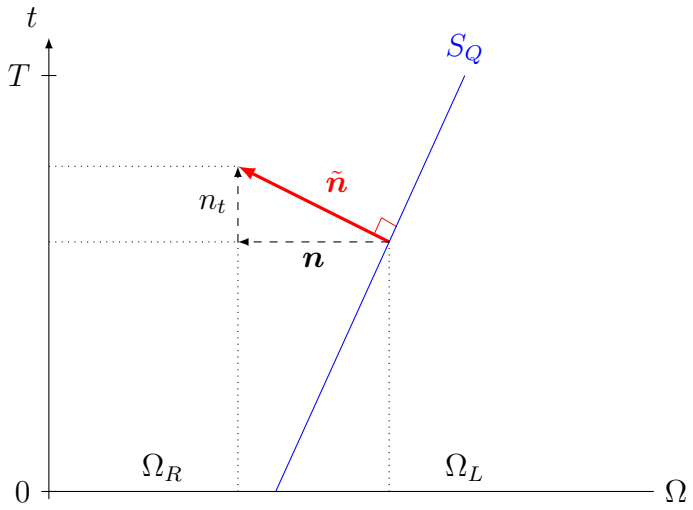
Relation de Rankine-Hugoniot

Soit Ω un ouvert de $\mathbb{R}^3$ et T un réel positif. Considérons une surface spatio-temporelle régulière $S_Q \subset Q = \Omega \times [0\,;T]$ orientée par une normale notée $\tilde{\boldsymbol{n}} = (\boldsymbol{n}^{\mathrm{T}}, n_t)^{\mathrm{T}}$ (figure 1.1). Le vecteur unitaire $\boldsymbol{n} \in \mathbb{R}^3$ est la partie spatiale de la normale et $n_t \in \mathbb{R}$ est la partie temporelle de la normale.

Cette surface S_Q sépare Ω en deux ouverts Ω_L et Ω_R. Nous supposons que $\boldsymbol{n}$ est orienté de Ω_L vers Ω_R. Notons $\boldsymbol{w}_L$, respectivement $\boldsymbol{w}_R$, la restriction des champs électromagnétiques $\boldsymbol{w}$ à Ω_L, respectivement Ω_R. $\boldsymbol{w}_L$ et $\boldsymbol{w}_R$ sont solutions $\mathcal{C}^1$ des équations de Maxwell et $\boldsymbol{w}$ vérifie la condition de Rankine-Hugoniot sur la discontinuité S_Q.

Dans la suite de cette section, l'indice L, respectivement R, apposé à une notation existante fait référence au côté Ω_L, respectivement Ω_R, de la discontinuité.

FIGURE 1.1 – Schéma représentant une discontinuité spatio-temporelle S_Q dans $Q = \Omega \times [0\,;T]$.



Définition 3. *Soit $(\boldsymbol{n}^{\mathrm{T}}, n_t)^{\mathrm{T}}$, avec $\boldsymbol{n}$ un vecteur unitaire, la normale au support d'une discontinuité de la solution du système hyperbolique (1.5). La*

relation de Rankine-Hugoniot satisfaite sur cette discontinuité est donnée par l'équation :

$$n_t \left((\boldsymbol{A}_t)_L \boldsymbol{w}_L - (\boldsymbol{A}_t)_R \boldsymbol{w}_R \right) = \left((\boldsymbol{A}_i n_i)_L \boldsymbol{w}_L - (\boldsymbol{A}_i n_i)_R \boldsymbol{w}_R \right), \tag{1.9}$$

avec n_t la vitesse de propagation de la discontinuité.

En appliquant cette relation aux équations de Maxwell, nous déduisons la propriété suivante :

Proposition 3. *De part et d'autre de la discontinuité, les champs vérifient :*

$$n_t(\varepsilon_L \boldsymbol{E}_L - \varepsilon_R \boldsymbol{E}_R) = -\boldsymbol{n} \times (\boldsymbol{H}_L - \boldsymbol{H}_R), \tag{1.10a}$$
$$n_t(\mu_L \boldsymbol{H}_L - \mu_R \boldsymbol{H}_R) = \boldsymbol{n} \times (\boldsymbol{E}_L - \boldsymbol{E}_R). \tag{1.10b}$$

Interprétation

En milieu homogène ($\varepsilon_L = \varepsilon_R = \varepsilon$ et $\mu_L = \mu_R = \mu$), les équations précédentes peuvent être écrites sous la forme matricielle :

$$n_t \begin{pmatrix} \boldsymbol{E}_L - \boldsymbol{E}_R \\ \boldsymbol{H}_L - \boldsymbol{H}_R \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{\varepsilon}\boldsymbol{n}\times \\ \frac{1}{\mu}\boldsymbol{n}\times & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{E}_L - \boldsymbol{E}_R \\ \boldsymbol{H}_L - \boldsymbol{H}_R \end{pmatrix}. \tag{1.11}$$

Le vecteur $((\boldsymbol{E}_L - \boldsymbol{E}_R)^{\mathrm{T}}, (\boldsymbol{H}_L - \boldsymbol{H}_R)^{\mathrm{T}})^{\mathrm{T}}$ est donc vecteur propre associé à la valeur propre n_t de cette matrice. D'autre part, les valeurs propres de cette matrice peuvent être calculées et sont $v = (\varepsilon\mu)^{-\frac{1}{2}}$, 0 et $-v$, chacune de multiplicité 2. Avec v, la vitesse de la lumière dans le milieu considéré.

Dans le cas où $n_t = \pm v$, la discontinuité se propage à la vitesse de la lumière. Ce type de discontinuité apparaît dans le cas où la condition initiale est discontinue ou pour des sources électromagnétiques de type mesure de Dirac. Les méthodes d'ordre élevé ont du mal à reproduire correctement ces solutions (apparition d'oscillations de Gibbs). En pratique, ces deux cas de figure ne se présenteront pas, puisque physiquement ce type de discontinuité n'apparaît pas spontanément.

Dans le cas où $n_t = 0$, la discontinuité est fixe. L'égalité obtenue se réduit alors à une condition de continuité des champs tangents à S_Q. Il n'est donc pas possible d'observer des solutions discontinues sauf si la condition initiale est elle-même discontinue. Or, à l'instant initial, les conditions de divergence (1.1c) et (1.1d) réduisent les possibilités d'observer des solutions discontinues. En effet, les égalités suivantes sont vérifiées sur la discontinuité :

$$\varepsilon \boldsymbol{n} \cdot (\boldsymbol{E}_L - \boldsymbol{E}_R) = \rho_L - \rho_R, \tag{1.12a}$$
$$\mu \boldsymbol{n} \cdot (\boldsymbol{H}_L - \boldsymbol{H}_R) = 0. \tag{1.12b}$$

Ainsi, si la charge ρ est assez régulière, cela impose que les sauts des composantes normales de $\boldsymbol{E}$ et $\boldsymbol{H}$ sont nuls à l'instant initial et donc à chaque instants de la solution d'après la proposition 1. Par conséquent, les champs ne peuvent pas présenter de sauts tangentiels ou normaux et ne peuvent être que continus.

Dans le cas où les paramètres du milieu sont discontinus, la situation est différente. À l'instant initial, nous avons :

$$\boldsymbol{n} \cdot (\varepsilon_L \boldsymbol{E}_L - \varepsilon_R \boldsymbol{E}_R) = 0, \tag{1.13a}$$
$$\boldsymbol{n} \cdot (\mu_L \boldsymbol{H}_L - \mu_R \boldsymbol{H}_R) = 0. \tag{1.13b}$$

La solution initiale est donc discontinue au niveau des changements de matériaux. Ainsi, pour des matériaux inhomogènes, sur une discontinuité fixe, les champs électromagnétiques tangents à la discontinuité sont continus et leurs composantes normales sont discontinues. La méthode GD prend naturellement en compte ces discontinuités fixes.

1.2.3 Problème d'évolution

Soit Ω un ouvert de $\mathbb{R}^3$ et T un réel positif. Pour que le problème d'évolution lié au système de Friedrichs (1.5) soit bien posé dans le domaine spatio-temporel $Q = \Omega \times [0\,;T]$, il faut y ajouter une condition initiale, notée $\boldsymbol{w}_0$, et des conditions aux limites locales et linéaires sous forme d'une matrice, notée $\boldsymbol{M}(x)$, et définie sur le bord $\partial Q = \partial\Omega \times [0\,;T]$:

$$\partial_t \boldsymbol{A}_t \boldsymbol{w} + \boldsymbol{C}\boldsymbol{w} + \boldsymbol{A}_i \partial_i \boldsymbol{w} = \boldsymbol{S} \quad \text{sur } Q, \tag{1.14a}$$
$$\boldsymbol{w}(x,0) = \boldsymbol{w}_0(x) \quad \text{sur } \Omega, \tag{1.14b}$$
$$\boldsymbol{M}\boldsymbol{w} = 0 \quad \text{sur } \partial Q. \tag{1.14c}$$

Soit aussi l'espace vectoriel V_b défini sur le bord $\partial\Omega$ tel que :

$$V_b = \ker \boldsymbol{M}, \tag{1.15}$$

et donc :

$$\boldsymbol{w} \in V_b \iff \boldsymbol{M}\boldsymbol{w} = 0. \tag{1.16}$$

Nous pouvons définir une énergie associée aux solutions du système (1.14) :

Définition 4. *L'énergie $\mathcal{E}$ associée aux champs $\boldsymbol{w}$ à un instant t est définie par l'intégrale :*

$$\mathcal{E} = \frac{1}{2} \int_\Omega (\boldsymbol{A}_t \boldsymbol{w}) \cdot \boldsymbol{w} dx. \tag{1.17}$$

Unicité de la solution

Supposons, pour simplifier, le système sans source ($\boldsymbol{C} = 0$ et $\boldsymbol{S} = 0$), alors en multipliant le système hyperbolique (1.5) par $\boldsymbol{w}$ puis en intégrant par parties, nous obtenons le bilan d'énergie :

$$\partial_t \mathcal{E} = -\frac{1}{2} \int_{\partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}) \cdot \boldsymbol{w} ds. \tag{1.18}$$

Ainsi, dans le cas d'un second membre nul, la décroissance de l'énergie au cours du temps assure l'unicité de la solution.

Théorème 1. *Si :*

$$\forall \boldsymbol{w} \in V_b, (\boldsymbol{A}_i n_i \boldsymbol{w}) \cdot \boldsymbol{w} \geq 0, \tag{1.19}$$

alors la solution du problème d'évolution (1.14) *est unique.*

Existence de la solution

Les conditions d'existence ont d'abord été analysées par Lax et Phillips [51] puis étendues par Rauch [60]. Leurs travaux assurent l'existence de la solution par l'unicité de la solution du problème adjoint avec condition finale qui s'écrit :

$$\partial_t \boldsymbol{A}_t \boldsymbol{w} + \boldsymbol{C}^{\mathrm{T}} \boldsymbol{w} - \boldsymbol{A}_i \partial_i \boldsymbol{w} = 0 \quad \text{sur } Q, \tag{1.20a}$$

$$\boldsymbol{w}(x, T) = \boldsymbol{w}_T(x) \quad \text{sur } \Omega, \tag{1.20b}$$

$$\boldsymbol{w} \in V_b^{\#} \quad \text{sur } \partial Q. \tag{1.20c}$$

Ce problème adjoint est associé à une condition aux limites adjointe (1.20c) avec $V_b^{\#} = (\boldsymbol{A}_i n_i V_b)^{\perp}$, *i.e.* :

$$\forall \boldsymbol{w} \in V_b, \ \forall \boldsymbol{w}^{\#} \in V_b^{\#}, \ (\boldsymbol{A}_i n_i \boldsymbol{w}) \cdot \boldsymbol{w}^{\#} = 0. \tag{1.21}$$

Cette condition implique que la dimension de l'espace V_b doit être égale au nombre de valeurs propres positives ou nulles de $\boldsymbol{A}_i n_i$ en comptant leur multiplicité, soit $\dim V_b = 4$. La justification de cette assertion est donnée en annexe A. Nous sommes donc amenés à introduire les définitions suivantes des parties positives et négatives de $\boldsymbol{A}_i n_i$.

Définition 5. *Soit* $\boldsymbol{D} = (d_{i,j})_{i,j \in \mathbb{N}^\star}$ *une matrice réelle* ***diagonale****. La* ***valeur absolue*** *de* $\boldsymbol{D}$ *est définie par* $|\boldsymbol{D}| = (|d_{i,j}|)$.

Définition 6. *Soit* $\boldsymbol{A}$ *une matrice carrée réelle et diagonalisable sur* $\mathbb{R}$. *Soient* $\boldsymbol{P}$ *une matrice inversible et* $\boldsymbol{D}$ *une matrice diagonale telles que* $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$. *La* ***partie positive*** *de* $\boldsymbol{A}$, *notée* $\boldsymbol{A}^+$, *est définie par :*

$$\boldsymbol{A}^+ = \frac{1}{2}\boldsymbol{P}\left(\boldsymbol{D} + |\boldsymbol{D}|\right)\boldsymbol{P}^{-1}. \tag{1.22}$$

La ***partie négative*** *de* $\boldsymbol{A}$, *notée* $\boldsymbol{A}^-$, *est définie par :*

$$\boldsymbol{A}^- = \frac{1}{2}\boldsymbol{P}\left(\boldsymbol{D} - |\boldsymbol{D}|\right)\boldsymbol{P}^{-1}. \tag{1.23}$$

Théorème 2. *Le problème d'évolution* (1.14) *admet une unique solution si en tout point du bord, on a :*

$$\forall \boldsymbol{w} \in V_b,\ (\boldsymbol{A}_i n_i \boldsymbol{w}) \cdot \boldsymbol{w} \geq 0 \tag{1.24}$$

et

$$\dim(V_b) = \dim(\ker(\boldsymbol{A}_i n_i{}^-)). \tag{1.25}$$

La condition (1.24) est appelée condition de dissipation. La condition (1.25) est appelée condition de maximalité.

1.3 Méthode Galerkin Discontinue

Dans cette section, nous allons construire à partir d'un système de Friedrichs bien posé (1.14) une formulation faible de type Galerkin Discontinue (GD) du problème d'évolution. Nous verrons que la condition de Lax-Phillips (théorème 2) se traduit dans la formulation faible par un choix de flux numérique frontière vérifiant les propriétés de dissipation et de maximalité.

Soit Ω un ouvert de $\mathbb{R}^3$. Soient des ouverts $(\Omega_k)_{k\in[\![1\,;K]\!]}$ tels que sur chacun d'eux les paramètres physiques du milieu soient continus et :

$$\bigcup_{k=1}^{K} \overline{\Omega_k} = \overline{\Omega} \text{ et } \forall k \neq k', \Omega_k \cap \Omega_{k'} = \emptyset. \tag{1.26}$$

Nous avons vu précédemment (section 1.2.2) que les discontinuités stationnaires de la solution vont coïncider avec les discontinuités des matériaux

diélectriques. La solution du système admet donc des discontinuités uniquement sur l'union des frontières des Ω_k.

Nous ajoutons alors des discontinuités fictives à la solution. Pour cela nous définissons un ensemble fini d'ouverts $\mathcal{M} = (L_i)_{i\in[\![1\,;N]\!]}$ inclus dans Ω tels que :

$$\bigcup_{i=1}^{N} \overline{L_i} = \overline{\Omega},\ \forall i \neq i', L_i \cap L_{i'} = \emptyset \text{ et } \forall i, \exists k : L_i \subset \Omega_k. \tag{1.27}$$

En particulier, l'union des frontières des Ω_k est incluse dans l'union des frontières des L_i :

$$\bigcup_{k=1}^{K} \partial\Omega_k \subset \bigcup_{i=1}^{N} \partial L_i = \Sigma. \tag{1.28}$$

Ces ouverts L_i correspondent dans la suite aux éléments du maillage, ou mailles. L'union de ces ouverts est notée $\tilde{\Omega}$. L'union des frontières des mailles est notée Σ. Par cette méthode de construction, Σ, *i.e.* les faces des mailles, coïncide avec les discontinuités fixes de la solution. La solution du problème d'évolution est alors continue sur $\tilde{\Omega}$. Nous supposons aussi que la solution est continûment différentiable par rapport au temps et que sa restriction à L_i est continûment différentiable en espace sur chaque ensemble $\overline{L_i}$.

1.3.1 Formulation faible

Soit $\boldsymbol{w}$ une solution du système de lois de conservation (1.14). Nous utilisons la notation $\langle f, \psi\rangle$ pour le crochet distributionnel d'une distribution $f \in \mathcal{D}'(\Omega)$ et d'une fonction test $\psi \in \mathcal{D}(\Omega)$. Rappelons que dans le cas où $f \in \mathrm{L}^1_{\mathrm{loc}}$, ce crochet vérifie :

$$\langle f, \psi\rangle = \int_\Omega f\psi dx. \tag{1.29}$$

D'autre part, la dérivée partielle ∂_i de f au sens des distributions est définie par :

$$\langle \partial_i f, \psi\rangle := -\langle f, \partial_i\psi\rangle\,. \tag{1.30}$$

Supposons alors que nous cherchons une solution $\boldsymbol{w}$ du problème (1.14) au sens des distributions. Afin de simplifier l'exposé, nous supposerons que

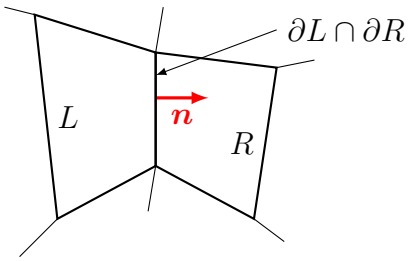
la matrice $\boldsymbol{A}_t$ est égale à l'identité. Nous avons alors :

$$\forall \psi \in \mathcal{C}^1(\overline{\Omega}),\ \langle \partial_t \boldsymbol{w}, \psi \rangle + \langle \boldsymbol{A}_i \partial_i \boldsymbol{w}, \psi \rangle + \int_{\partial\Omega} \boldsymbol{M}\boldsymbol{w}\psi ds = 0. \tag{1.31}$$

Le vecteur de fonctions $\boldsymbol{w}$ est discontinu sur $\Sigma \setminus \partial\Omega$. Ainsi, en appliquant la formule des sauts [63], nous pouvons écrire l'équation précédente sous la forme :

$$\begin{aligned} \forall \psi \in \mathcal{C}^1(\overline{\Omega}),\ \int_{\Omega} \partial_t \boldsymbol{w}\psi dx + \int_{\tilde{\Omega}} \boldsymbol{A}_i \partial_i \boldsymbol{w}\psi dx \\ + \int_{\Sigma\setminus\partial\Omega} \boldsymbol{A}_i n_i (\boldsymbol{w}_R - \boldsymbol{w}_L)\psi ds \\ + \int_{\partial\Omega} \boldsymbol{M}\boldsymbol{w}\psi ds = 0. \end{aligned} \tag{1.32}$$

FIGURE 1.2 – Représentation de deux mailles voisines notées L et R. La normale à l'interface entre les deux mailles est dirigée de L vers R.



Nous rappelons que les indices L et R font références aux ouverts situés de chaque côté de Σ (figure 1.2). La surface Σ est orientée de telle sorte que le vecteur normal unitaire à Σ, noté $\boldsymbol{n}$, soit orienté de L vers R. Les champs $\boldsymbol{w}_L$, respectivement $\boldsymbol{w}_R$, sont la restriction à Σ des champs provenant du côté L, respectivement R.

1.3.2 Formulation GD

Maintenant que nous avons établi une formulation faible du problème, nous pouvons en déduire une formulation GD. Nous allons étendre la formulation faible à des fonctions tests pouvant également présenter une discontinuité sur les interfaces entre ouverts et ayant la même régularité que la solution. Pour cela, nous introduisons un flux numérique, dépendant des

états $\boldsymbol{w}_L$ et $\boldsymbol{w}_R$ afin de donner un sens à l'intégrale sur $\Sigma \setminus \partial\Omega$.

Rappelons que dans le cas des équations de Maxwell, la matrice $\boldsymbol{A}_i n_i$ est symétrique et vaut :

$$\boldsymbol{A}_i n_i = \begin{pmatrix} 0 & -\boldsymbol{n}\times \\ \boldsymbol{n}\times & 0 \end{pmatrix}, \text{ avec } \boldsymbol{n}\times = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}. \tag{1.33}$$

Les valeurs propres de cette matrice sont $\pm \|\boldsymbol{n}\|^2$ et 0. Dans le cas où le vecteur $\boldsymbol{n}$ est unitaire, les matrices $\boldsymbol{A}_i {n_i}^+$ et $\boldsymbol{A}_i {n_i}^-$ sont elles aussi symétriques et valent :

$$\boldsymbol{A}_i {n_i}^+ = \frac{1}{2}\begin{pmatrix} -(\boldsymbol{n}\times)^2 & -\boldsymbol{n}\times \\ \boldsymbol{n}\times & -(\boldsymbol{n}\times)^2 \end{pmatrix}, \tag{1.34a}$$

$$\boldsymbol{A}_i {n_i}^- = \frac{1}{2}\begin{pmatrix} (\boldsymbol{n}\times)^2 & -\boldsymbol{n}\times \\ \boldsymbol{n}\times & (\boldsymbol{n}\times)^2 \end{pmatrix}. \tag{1.34b}$$

La matrice $(\boldsymbol{n}\times)^2$ correspond à l'application, sur un vecteur $\boldsymbol{u}$, de l'opération $\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{u})$. Nous avons :

$$(\boldsymbol{n}\times)^2 = \begin{pmatrix} -n_2^2 - n_3^2 & n_1 n_2 & n_3 n_1 \\ n_1 n_2 & -n_3^2 - n_1^2 & n_2 n_3 \\ n_3 n_1 & n_2 n_3 & -n_1^2 - n_2^2 \end{pmatrix}. \tag{1.35}$$

Nous introduisons alors ces matrices dans la formulation faible (1.32) et obtenons ainsi la formulation GD. Pour toute fonction test ψ éventuellement discontinue sur Σ :

$$\begin{aligned} \int_\Omega \partial_t \boldsymbol{w}\psi dx + \int_{\tilde{\Omega}} \boldsymbol{A}_i \partial_i \boldsymbol{w}\psi dx \\ + \int_{\Sigma\setminus\partial\Omega} \boldsymbol{A}_i {n_i}^-(\boldsymbol{w}_R - \boldsymbol{w}_L)\psi_L + \boldsymbol{A}_i {n_i}^+(\boldsymbol{w}_R - \boldsymbol{w}_L)\psi_R ds \\ + \int_{\partial\Omega} \boldsymbol{M}\boldsymbol{w}_L\psi_L ds = 0. \end{aligned} \tag{1.36}$$

Il est important de remarquer ici que cette formulation faible n'a plus de sens dans l'espace des distributions $\mathcal{D}(\Omega)$ puisque les fonctions tests sont maintenant discontinues. Cette formulation a du sens par exemple lorsque $\boldsymbol{w}$ est de classe $\mathcal{C}^1$ sur toute cellule $\overline{L}$ du maillage $\mathcal{M}$. Nous pouvons donc introduire l'espace de fonctions suivant :

$$\mathcal{H} = \{f \in \mathrm{L}^2(\Omega) : \forall L \in \mathcal{M}, f_{|L} \in \mathcal{C}^1(\overline{L})\}. \tag{1.37}$$

Nous cherchons alors la solution $\boldsymbol{w}$ dans $\mathcal{C}^1([0\,;T],\mathcal{H}^m)$. Lorsque les fonctions tests s'annulent en dehors de $\overline{L}$, la formulation GD (1.36) sur L devient :

$$\begin{aligned}\int_L \partial_t \boldsymbol{w}\psi dx + \int_L \boldsymbol{A}_i \partial_i \boldsymbol{w}\psi dx \\ + \int_{\partial L\setminus\partial\Omega} \boldsymbol{A}_i n_i{}^- (\boldsymbol{w}_R - \boldsymbol{w}_L)\psi_L ds \\ + \int_{\partial L\cap\partial\Omega} \boldsymbol{M}\boldsymbol{w}_L\psi_L ds = 0.\end{aligned} \tag{1.38}$$

En appliquant ensuite la formule de Green sur le second terme :

$$\int_L \nabla\phi\psi = -\int_L \nabla\psi\phi + \int_{\partial L} \boldsymbol{n}\phi\psi, \tag{1.39}$$

l'équation précédente devient :

$$\begin{aligned}\int_L \partial_t \boldsymbol{w}\psi dx - \int_L \boldsymbol{A}_i \partial_i \psi \boldsymbol{w} dx \\ + \int_{\partial L\setminus\partial\Omega} (\boldsymbol{A}_i n_i{}^+ \boldsymbol{w}_L + \boldsymbol{A}_i n_i{}^- \boldsymbol{w}_R)\psi_L ds \\ + \int_{\partial L\cap\partial\Omega} (\boldsymbol{A}_i n_i + \boldsymbol{M})\boldsymbol{w}_L\psi_L ds = 0.\end{aligned} \tag{1.40}$$

Nous introduisons alors le flux numérique :

$$F(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}) = \boldsymbol{A}_i n_i{}^+ \boldsymbol{w}_L + \boldsymbol{A}_i n_i{}^- \boldsymbol{w}_R \tag{1.41}$$

et le flux de bord :

$$F_b(\boldsymbol{w}, \boldsymbol{n}) = (\boldsymbol{A}_i n_i + \boldsymbol{M})\boldsymbol{w}. \tag{1.42}$$

Ce flux numérique est un flux décentré. Nous avons choisi ce flux car il introduit une dissipation numérique qui améliore la stabilité du schéma selon les travaux de Castel *et al.* [12], Cohen *et al.* [15] et Hesthaven *et al.* [37].

Pour obtenir une approximation convergente vers la bonne solution, le flux doit vérifier les propriétés de consistance :

$$F(\boldsymbol{w}, \boldsymbol{w}, \boldsymbol{n}) = \boldsymbol{A}_i n_i \boldsymbol{w} \tag{1.43}$$

et de conservation :

$$F(\boldsymbol{w}_L, \boldsymbol{w}_R, -\boldsymbol{n}) = -F(\boldsymbol{w}_R, \boldsymbol{w}_L, \boldsymbol{n}). \tag{1.44}$$

Il est possible de choisir d'autres flux numériques. Dans le cas des équations de Maxwell, nous pouvons donner une famille de flux numériques plus généraux, en fonction d'un paramètre $\alpha \geq 0$, qui garantissent la stabilité du schéma numérique :

$$F_\alpha(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}) = \begin{pmatrix} -\frac{1}{2}\boldsymbol{n} \times (\boldsymbol{H}_L + \boldsymbol{H}_R) + \alpha(\boldsymbol{n}\times)^2(\boldsymbol{E}_R - \boldsymbol{E}_L) \\ \frac{1}{2}\boldsymbol{n} \times (\boldsymbol{E}_L + \boldsymbol{E}_R) + \alpha(\boldsymbol{n}\times)^2(\boldsymbol{H}_R - \boldsymbol{H}_L) \end{pmatrix}. \tag{1.45}$$

Le paramètre α définit le décentrage du flux numérique et la dissipation de l'énergie induite. Lorsque $\alpha = \frac{1}{2}$, le flux est décentré. Lorsque $\alpha = 0$, le flux est centré.

Proposition 4. *La famille de flux F_α vérifie les propriétés de consistance (1.43) et de conservation (1.44).*

Démonstration. Consistance :

$$F_\alpha(\boldsymbol{w}, \boldsymbol{w}, \boldsymbol{n}) = \begin{pmatrix} -\boldsymbol{n} \times \boldsymbol{H} \\ \boldsymbol{n} \times \boldsymbol{E} \end{pmatrix} = \begin{pmatrix} 0 & -\boldsymbol{n}\times \\ \boldsymbol{n}\times & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{H} \end{pmatrix} = \boldsymbol{A}_i n_i \boldsymbol{w}. \tag{1.46}$$

Conservation :

$$\begin{aligned} &-F_\alpha(\boldsymbol{w}_L, \boldsymbol{w}_R, -\boldsymbol{n}) \\ &\quad = \begin{pmatrix} \frac{1}{2}(-\boldsymbol{n}) \times (\boldsymbol{H}_L + \boldsymbol{H}_R) - \alpha(-\boldsymbol{n}\times)^2(\boldsymbol{E}_R - \boldsymbol{E}_L) \\ -\frac{1}{2}(-\boldsymbol{n}) \times (\boldsymbol{E}_L + \boldsymbol{E}_R) - \alpha(-\boldsymbol{n}\times)^2(\boldsymbol{H}_R - \boldsymbol{H}_L) \end{pmatrix} \\ &\quad = \begin{pmatrix} -\frac{1}{2}\boldsymbol{n} \times (\boldsymbol{H}_R + \boldsymbol{H}_L) + \alpha(\boldsymbol{n}\times)^2(\boldsymbol{E}_L - \boldsymbol{E}_R) \\ \frac{1}{2}\boldsymbol{n} \times (\boldsymbol{E}_R + \boldsymbol{E}_L) + \alpha(\boldsymbol{n}\times)^2(\boldsymbol{H}_L - \boldsymbol{H}_R) \end{pmatrix} \\ &\quad = F_\alpha(\boldsymbol{w}_R, \boldsymbol{w}_L, \boldsymbol{n}). \end{aligned} \tag{1.47}$$

□

Nous obtenons finalement la formulation GD de notre problème : trouver une solution $\boldsymbol{w} \in \mathcal{C}^1([0\,;T], \mathcal{H}^m)$ telle que :

$$\begin{aligned} &\forall L \in \mathcal{M}, \forall \psi \in \mathcal{C}^1(\overline{L}), \\ &\qquad \int_L \partial_t \boldsymbol{w} \psi dx - \int_L F\left(\boldsymbol{w}, \boldsymbol{w}, \nabla\psi\right) dx \\ &\qquad\qquad + \int_{\partial L \setminus \partial\Omega} F\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) \psi_L ds \\ &\qquad\qquad + \int_{\partial L \cap \partial\Omega} F_b\left(\boldsymbol{w}_L, \boldsymbol{n}\right) \psi_L ds = 0. \end{aligned} \tag{1.48}$$

En ajoutant à cette formulation un état initial $\boldsymbol{w}(0) = \boldsymbol{w}_0$, le problème d'évolution ainsi posé admet une unique solution.

1.3.3 Stabilité de la méthode

Nous nous intéressons à présent à la stabilité de cette méthode. Plaçons-nous dans le cas où aucune source d'énergie n'est placée dans le domaine. La méthode sera stable si l'énergie du système décroît au cours du temps. Afin d'alléger les écritures, supposons, sans perte de généralité, que la matrice $\boldsymbol{A}_t$ est égale à l'identité. Dans la définition 4, nous avons défini une énergie associée à la solution $\boldsymbol{w}$. En différenciant son expression par rapport au temps nous obtenons :

$$\partial_t \mathcal{E} = \frac{1}{2} \int_\Omega \partial_t \boldsymbol{w} \cdot \boldsymbol{w} dx. \tag{1.49}$$

En prenant $\psi = \boldsymbol{w}$ dans la formulation GD (1.36) nous pouvons exprimer la dérivée temporelle de l'énergie :

$$\begin{aligned} \int_\Omega \partial_t \boldsymbol{w} \cdot \boldsymbol{w} dx = &- \int_{\tilde{\Omega}} (\boldsymbol{A}_i \partial_i \boldsymbol{w}) \cdot \boldsymbol{w} dx \\ &- \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i {n_i}^-(\boldsymbol{w}_R - \boldsymbol{w}_L)) \cdot \boldsymbol{w}_L ds \\ &- \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i {n_i}^+(\boldsymbol{w}_R - \boldsymbol{w}_L)) \cdot \boldsymbol{w}_R ds \\ &- \int_{\partial\Omega} (\boldsymbol{M} \boldsymbol{w}_L) \cdot \boldsymbol{w}_L ds. \end{aligned} \tag{1.50}$$

Le premier terme du membre de droite peut être décomposé comme suit :

$$\begin{aligned} \int_{\tilde{\Omega}} (\boldsymbol{A}_i \partial_i \boldsymbol{w}) \cdot \boldsymbol{w} dx &= \frac{1}{2} \int_{\tilde{\Omega}} \partial_i ((\boldsymbol{A}_i \boldsymbol{w}) \cdot \boldsymbol{w}) dx \\ &= \frac{1}{2} \int_\Sigma (\boldsymbol{A}_i n_i \boldsymbol{w}_L) \cdot \boldsymbol{w}_L ds - \frac{1}{2} \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}_R) \cdot \boldsymbol{w}_R ds \\ &= \frac{1}{2} \int_{\partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}_L) \cdot \boldsymbol{w}_L ds + \frac{1}{2} \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}_L) \cdot \boldsymbol{w}_L ds \\ &\quad - \frac{1}{2} \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}_R) \cdot \boldsymbol{w}_R ds \\ &= \frac{1}{2} \int_{\partial\Omega} (\boldsymbol{A}_i n_i \boldsymbol{w}_L) \cdot \boldsymbol{w}_L ds \\ &\quad - \frac{1}{2} \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i n_i (\boldsymbol{w}_R - \boldsymbol{w}_L)) \cdot \boldsymbol{w}_L ds \\ &\quad - \frac{1}{2} \int_{\Sigma \setminus \partial\Omega} (\boldsymbol{A}_i n_i (\boldsymbol{w}_R - \boldsymbol{w}_L)) \cdot \boldsymbol{w}_R ds. \end{aligned} \tag{1.51}$$

De plus,

$$\begin{aligned}\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(\boldsymbol{A}_i n_i(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_L ds-\int_{\Sigma\setminus\partial\Omega}(\boldsymbol{A}_i {n_i}^-(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_L ds\\ =\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(|\boldsymbol{A}_i n_i|\,(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_L ds\end{aligned} \tag{1.52}$$

et

$$\begin{aligned}\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(\boldsymbol{A}_i n_i(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_R ds-\int_{\Sigma\setminus\partial\Omega}(\boldsymbol{A}_i {n_i}^+(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_R ds\\ =-\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(|\boldsymbol{A}_i n_i|\,(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot\boldsymbol{w}_R ds\end{aligned} \tag{1.53}$$

où $|\boldsymbol{A}_i n_i| = \boldsymbol{A}_i {n_i}^+ - \boldsymbol{A}_i {n_i}^-$ est une matrice dont les valeurs propres sont des réels positifs.

Ainsi, nous obtenons une expression de l'évolution de l'énergie ne faisant intervenir que des intégrales surfaciques :

$$\begin{aligned}\int_\Omega \partial_t\boldsymbol{w}\cdot\boldsymbol{w}dx=&-\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(|\boldsymbol{A}_i n_i|\,(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot(\boldsymbol{w}_R-\boldsymbol{w}_L)ds\\ &-\int_{\partial\Omega}\left(\left(\frac{1}{2}\boldsymbol{A}_i n_i+\boldsymbol{M}\right)\boldsymbol{w}_L\right)\cdot\boldsymbol{w}_L ds.\end{aligned} \tag{1.54}$$

Cette expression de l'énergie nous permet alors d'énoncer un théorème donnant une condition suffisante de stabilité portant sur les conditions aux limites.

Théorème 3. *L'énergie $\mathcal{E}$ du système décroît si la matrice $\frac{1}{2}\boldsymbol{A}_i n_i + \boldsymbol{M}$ est positive.*

Démonstration. La dérivée en temps de l'énergie vérifie

$$\begin{aligned}2\partial_t\mathcal{E}=&-\frac{1}{2}\int_{\Sigma\setminus\partial\Omega}(|\boldsymbol{A}_i n_i|\,(\boldsymbol{w}_R-\boldsymbol{w}_L))\cdot(\boldsymbol{w}_R-\boldsymbol{w}_L)ds\\ &-\int_{\partial\Omega}\left(\left(\frac{1}{2}\boldsymbol{A}_i n_i+\boldsymbol{M}\right)\boldsymbol{w}_L\right)\cdot\boldsymbol{w}_L ds.\end{aligned} \tag{1.55}$$

Puisque la matrice $|\boldsymbol{A}_i n_i|$ est une matrice positive, si la matrice $\frac{1}{2}\boldsymbol{A}_i n_i + \boldsymbol{M}$ est aussi positive, les intégrales de cette équation sont positives pour tout $\boldsymbol{w} \in \mathcal{H}^m$. Dans ce cas la dérivée en temps de $\mathcal{E}$ est négative et donc $\mathcal{E}$ décroît. □

Remarque 1. *Cette condition de stabilité porte sur l'expression des conditions aux limites utilisées dans la formulation GD. En particulier, ce théorème nous permettra de déduire d'une condition « physique », la (ou les) matrice(s) $\boldsymbol{M}$ la représentant et garantissant la stabilité du schéma en découlant.*

Remarque 2. *Si la condition de stabilité est vérifiée, alors le théorème assurant l'unicité de la solution (théorème 1) est vérifié. En effet,*

$$\forall \boldsymbol{w} \in \ker \boldsymbol{M},\ 0 \leq \left(\left(\frac{1}{2} \boldsymbol{A}_i n_i + \boldsymbol{M} \right) \boldsymbol{w} \right) \cdot \boldsymbol{w} = \frac{1}{2} (\boldsymbol{A}_i n_i \boldsymbol{w}) \cdot \boldsymbol{w}. \tag{1.56}$$

1.3.4 Formulation semi-discrète

Supposons maintenant que nous avons construit un maillage $\mathcal{M}$ de Ω dont les bords des mailles correspondent exactement aux discontinuités stationnaires de la solution exacte. Nous approximons alors la solution $\boldsymbol{w}$ du système de Friedrichs (1.5) par une solution discrète $\mathbf{w}$ dont les composantes à chaque instant t sont polynomiales de degré d dans chaque maille. Ces fonctions polynomiales forment un espace vectoriel de dimension p engendré par la base $(\psi_{L,i})_{i \in [\![0\,;\,p-1]\!]}$. Par exemple, cette base pourrait être constituée de produits tensoriels de polynômes de degré au plus d, nous décrirons plus loin un tel espace d'approximation. Notons alors cet espace d'approximation :

$$\mathcal{H}_h = \mathrm{vect}\{\psi_{L,i} : L \in \mathcal{M},\ i \in [\![0\,;\,p-1]\!]\}. \tag{1.57}$$

Le paramètre h représente un paramètre de finesse du maillage, par exemple le diamètre maximal des mailles. Les fonctions $\psi_{L,i}$ sont nulles en dehors de la maille L. Elles constituent une base de dimension finie de fonctions définies sur L.

Sur la maille L, nous avons :

$$\forall x \in L,\ \boldsymbol{w}(x,t) \approx \mathbf{w}(x,t) = \sum_{i=0}^{p-1} \mathbf{w}_{L,i}(t) \psi_{L,i}(x). \tag{1.58}$$

Le problème semi-discret s'écrit alors comme suit.

Trouver une solution $\mathbf{w} \in \mathcal{C}^1([0\,;T], \mathcal{H}_h^m)$ telle que :

$$\begin{aligned} &\forall L \in \mathcal{M},\ \forall \psi \in \mathcal{H}_h : \mathrm{supp}(\psi) \subset \overline{L}, \\ &\int_L \partial_t \mathbf{w} \psi dx - \int_L F\left(\mathbf{w}, \mathbf{w}, \nabla\psi\right) dx \\ &\qquad + \int_{\partial L \setminus \partial\Omega} F\left(\mathbf{w}_L, \mathbf{w}_R, \boldsymbol{n}\right) \psi_L ds \\ &\qquad + \int_{\partial L \cap \partial\Omega} F_b\left(\mathbf{w}_L, \boldsymbol{n}\right) \psi_L ds = 0. \end{aligned} \tag{1.59}$$

Théorème 4. *La solution du problème semi-discret* (1.59) *converge vers la solution du problème continu* (1.14) *lorsque h tend vers zéro.*

Nous ne démontrerons pas ce théorème ici. Dans un souci de complétude, la démonstration est donnée en annexe B.

À présent que le théorème de convergence du problème semi-discret est énoncé, nous pouvons identifier la solution discrète à la solution exacte dans la suite en utilisant l'abus de notation $\mathbf{w} = \boldsymbol{w}$.

Conclusion

Dans ce premier chapitre, nous avons rappelé les équations de Maxwell et montré qu'elles constituent un système hyperbolique dit de Friedrichs. En plus de ce système hyperbolique, le modèle de Maxwell donne une condition initiale sur la divergence des champs.

Nous avons ensuite présenté la méthode GD, particulièrement bien adaptée à la résolution de champs réguliers présentant des discontinuités sur des interfaces stationnaires. Cette méthode peut être appliquée à tout système de Friedrichs. Elle fait intervenir un flux numérique que nous avons choisi décentré afin d'améliorer la stabilité du schéma. Nous avons également caractérisé les conditions aux limites permettant d'assurer cette stabilité.

Enfin, nous avons déduit de cette formulation GD une formulation semi-discrète dont la solution converge vers celle du problème continu. De plus, cette formulation prend naturellement en compte les discontinuités de la solution.

Dans le chapitre suivant, nous appliquerons la condition de stabilité du théorème 3 afin de valider différentes conditions aux limites présentes dans

la littérature et associées aux équations de Maxwell. Nous donnerons aussi quelques modèles mathématiques de simulation fréquemment utilisés et dont nous aurons besoin dans nos cas d'application.

Chapitre 2

Modèles mathématiques et conditions aux limites

Dans le chapitre précédent, nous avons décrit un schéma Galerkin Discontinu (GD) semi-discret (1.59) dont la convergence est assurée par le théorème 4. Ce schéma nous permettra de résoudre les équations de Maxwell (1.1) dans un domaine borné donné. Nous allons présenter les principaux modèles mathématiques nécessaires à la résolution des problèmes qui nous intéressent.

Nous commençons par décrire des conditions aux limites respectant la décroissance de l'énergie associée à la solution. Décroissance requise afin de garantir l'unicité de la solution. De telles conditions sont aussi nécessaires afin de limiter, informatiquement parlant, la taille du problème. Car sans ces conditions de bord, le rayon du domaine de calcul devrait être supérieur à la distance parcourue par une onde électromagnétique en autant de temps que le temps physique simulé. Nous présentons une technique permettant de résoudre un problème en domaine infini par l'ajout de couches absorbantes — appelées PML pour « *Perfectly Matched Layer* » — à la périphérie du domaine de calcul.

Dans un second temps, nous présentons des modèles représentant de fines plaques de matériaux afin de nous permettre de remplacer leur représentation géométrique en 3 dimensions par de simples surfaces. Nous verrons dans la suite (section 3.3.3) que la présence de ces fines plaques produirait un malus important en temps de simulation. De plus, de par leur nature, elles compliquent la phase de maillage.

Enfin, dans le but de traiter les matériaux diélectriques dont la permittivité varie en fonction de la fréquence, nous décrivons aussi un modèle volumique prenant en compte ces variations. Ce type de modèle est intéressant dans le cas de simulations faisant intervenir le corps humain qui présente ce

type de matériaux.

Ces modèles sont souvent inspirés de ceux implémentés dans les schémas de type différences finies d'un point de vue formel. Une étude des conditions aux limites admissibles pour les équations de Maxwell a été faite par Bourdel *et al.* [7]. Voir aussi [18, 23].

2.1 Conditions aux limites

2.1.1 Conducteur électrique parfait

Cette condition de bord représente un conducteur électrique parfait (PEC pour « *Perfect Electric Conductor* »). Elle correspond à un matériau de résistivité nulle (ou conductivité infinie) et est généralement utilisée pour représenter les métaux qui ont une résistivité proche de 0.

Cette condition est appliquée sur le bord du domaine de calcul afin de simuler les ondes à l'intérieur d'une boîte métallique. Elle est aussi utilisée dans la modélisation des dispositifs rayonnants, et permet de représenter un plan de masse ou tout autre composant conducteur tel que les lignes microruban (figure 2.2). Physiquement, cette condition est donnée par :

$$\boldsymbol{n} \times \boldsymbol{E} = 0, \tag{2.1}$$

où $\boldsymbol{n}$ est le vecteur normal unitaire sortant du domaine de calcul.

Soit $\boldsymbol{M}_{\mathrm{PEC}}$ la matrice associée à cette condition. Son noyau est l'ensemble $\ker(\boldsymbol{M}_{\mathrm{PEC}}) = \{\boldsymbol{w} = (\boldsymbol{E}^{\mathrm{T}}, \boldsymbol{H}^{\mathrm{T}})^{\mathrm{T}} : \boldsymbol{n} \times \boldsymbol{E} = 0\}$. La condition d'existence (1.25) de la solution impose que cet ensemble soit de dimension 4, ce qui est le cas ici : l'annulation du champ électrique tangent est composé de deux contraintes. La forme générique de la matrice $\boldsymbol{M}_{\mathrm{PEC}}$ est donnée par :

$$\boldsymbol{M}_{\mathrm{PEC}} = \begin{pmatrix} a(\boldsymbol{n}\times)^2 + b\boldsymbol{n}\times & 0 \\ c(\boldsymbol{n}\times)^2 + d\boldsymbol{n}\times & 0 \end{pmatrix} \tag{2.2}$$

où a, b, c et d sont des réels non tous nuls.

Pour une telle matrice $\boldsymbol{M}_{\mathrm{PEC}}$, la condition de stabilité du théorème 3

requiert la positivité de la matrice $\frac{1}{2}\boldsymbol{A}_i n_i + \boldsymbol{M}_{\text{PEC}}$, *i.e.* :

$$\begin{aligned}
0 \leq & \int_{\partial\Omega} \left(\left(\frac{1}{2}\boldsymbol{A}_i n_i + \boldsymbol{M}_{\text{PEC}} \right) \boldsymbol{w} \right) \cdot \boldsymbol{w} ds \\
= & \int_{\partial\Omega} \frac{1}{2}((-\boldsymbol{n} \times \boldsymbol{H}) \cdot \boldsymbol{E} + (\boldsymbol{n} \times \boldsymbol{E}) \cdot \boldsymbol{H}) ds \\
& + \int_{\partial\Omega} (a\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E}) + b\boldsymbol{n} \times \boldsymbol{E}) \cdot \boldsymbol{E} ds \\
& + \int_{\partial\Omega} (c\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{H}) + d\boldsymbol{n} \times \boldsymbol{H}) \cdot \boldsymbol{E} ds \\
= & \int_{\partial\Omega} a(\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E})) \cdot \boldsymbol{E} ds \\
& + \int_{\partial\Omega} c(\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{H})) \cdot \boldsymbol{E} ds \\
& + \int_{\partial\Omega} (1+d)(\boldsymbol{n} \times \boldsymbol{H}) \cdot \boldsymbol{E} ds.
\end{aligned} \tag{2.3}$$

Une condition suffisante pour que cette inégalité soit vraie est que, pour tous champs $\boldsymbol{E}$ et $\boldsymbol{H}$:

$$a(\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E})) \cdot \boldsymbol{E} \geq 0, \tag{2.4a}$$
$$c(\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{H})) \cdot \boldsymbol{E} \geq 0, \tag{2.4b}$$
$$(1+d)(\boldsymbol{n} \times \boldsymbol{H}) \cdot \boldsymbol{E} \geq 0. \tag{2.4c}$$

Cette condition est réalisée si $a \leq 0$, $c = 0$ et $d = -1$.

Nous remarquerons que la valeur de b n'a aucune incidence sur la stabilité de cette condition de bord, nous pouvons donc choisir $b = 0$ pour simplifier les calculs. De plus, le réel a se révèle être un paramètre permettant de conditionner la dissipation numérique de cette condition.

Les résultats obtenus au cours de la thèse de Strub [69] montrent que le choix du paramètre $a = -1$ augmente la dissipation de cette condition et ainsi améliore la stabilité dans des cas où les mailles du maillage sont très déformées. La matrice $\boldsymbol{M}_{\text{PEC}}$ s'écrit alors :

$$\boldsymbol{M}_{\text{PEC}} = \begin{pmatrix} -(\boldsymbol{n}\times)^2 & 0 \\ -\boldsymbol{n}\times & 0 \end{pmatrix}. \tag{2.5}$$

Cette formulation correspond donc à l'application d'un champ fictif $\boldsymbol{w}^\star$, défini en fonction de $\boldsymbol{w}$, imposé du côté extérieur au domaine de calcul :

$$\boldsymbol{w}^\star = \begin{pmatrix} \boldsymbol{E}^\top - \boldsymbol{E}^\perp \\ \boldsymbol{H}^\perp - \boldsymbol{H}^\top \end{pmatrix}, \tag{2.6}$$

où $\boldsymbol{E}^{\perp}$ et $\boldsymbol{H}^{\perp}$, respectivement $\boldsymbol{E}^{\top}$ et $\boldsymbol{H}^{\top}$, sont les composantes orthogonales, respectivement tangentielles, à la surface des champs $\boldsymbol{E}$ et $\boldsymbol{H}$.

Dans ce cas, le flux associé à cette condition s'écrit :

$$F_{\mathrm{PEC}}\left(\boldsymbol{w}, \boldsymbol{n}\right) = F\left(\boldsymbol{w}, \boldsymbol{w}^{\star}, \boldsymbol{n}\right) \tag{2.7}$$

où F est le flux numérique décentré.

2.1.2 Conducteur magnétique parfait

Dans la section précédente, nous avons présenté la condition de conducteur électrique parfait. En suivant le même raisonnement nous pouvons définir une condition de bord représentant un conducteur magnétique parfait (PMC).

Cette condition peut être utilisée de manière combinée avec le PEC pour résoudre des problèmes de propagation d'ondes planes dans un espace de dimension 3, notamment dans le but de tester des modèles de matériaux.

Physiquement, cette condition est donnée par :

$$\boldsymbol{n} \times \boldsymbol{H} = 0, \tag{2.8}$$

où $\boldsymbol{n}$ est le vecteur normal unitaire sortant du domaine de calcul. Nous obtenons alors la matrice correspondante, avec dissipation :

$$\boldsymbol{M}_{\mathrm{PMC}} = \begin{pmatrix} 0 & -(\boldsymbol{n}\times)^2 \\ 0 & -\boldsymbol{n}\times \end{pmatrix}. \tag{2.9}$$

2.1.3 Condition d'impédance de surface

Cette condition de bord a initialement été définie par Senior [66] sous l'acronyme SIBC (pour « *Surface Impedance Boundary Condition* »). Il s'agit d'une formulation générale dont nous présenterons deux cas particuliers dans la suite : la condition de type Silver-Müller et le modèle de Bérenger.

Le principe de cette condition est de lier les champs électrique et magnétique tangents au bord du domaine par une relation faisant intervenir une impédance de surface, notée $z \in \mathbb{R}^{+}$. Nous nous plaçons dans le cas où l'impédance ne dépend pas de la fréquence.

La formule associée à cette condition est connue sous le nom de « relation de Léontovitch » :

$$\boldsymbol{n} \times \boldsymbol{E} + z\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{H}) = 0, \tag{2.10}$$

où $\boldsymbol{n}$ est le vecteur normal unitaire sortant du domaine de calcul.

En effectuant le produit vectoriel de cette équation par $\boldsymbol{n}$, nous obtenons une formulation équivalente nous permettant de déduire une matrice $\boldsymbol{M}_{\text{SIBC}}$ décrivant cette condition et donnée par :

$$\boldsymbol{M}_{\text{SIBC}} = \begin{pmatrix} a(\boldsymbol{n}\times)^2 & -az\boldsymbol{n}\times \\ b\boldsymbol{n}\times & bz(\boldsymbol{n}\times)^2 \end{pmatrix} \tag{2.11}$$

où a et b sont des réels, dont l'un au moins est non nul.

La condition d'existence (1.25) est toujours vérifiée pour une matrice de cette forme. La condition de stabilité du théorème 3 requiert la positivité de la matrice $\boldsymbol{B} = \frac{1}{2}\boldsymbol{A}_i n_i + \boldsymbol{M}_{\text{SIBC}}$. Soit $\boldsymbol{w} = (\boldsymbol{E}^{\mathrm{T}}, \boldsymbol{H}^{\mathrm{T}})^{\mathrm{T}}$, alors :

$$\begin{aligned} \boldsymbol{w}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{w} =& (b + az + 1)\boldsymbol{H} \cdot \boldsymbol{n} \times \boldsymbol{E} \\ &+ a\boldsymbol{E} \cdot (\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E})) \\ &+ bz\boldsymbol{H} \cdot (\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{H})). \end{aligned} \tag{2.12}$$

Or, pour tout vecteur $\boldsymbol{v} \in \mathbb{R}^3$, la forme quadratique qui à $\boldsymbol{v}$ associe $\boldsymbol{v} \cdot (\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{v}))$ est négative, en effet :

$$\begin{aligned} \boldsymbol{v} \cdot (\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{v})) &= \boldsymbol{v} \cdot ((\boldsymbol{n} \cdot \boldsymbol{v})\boldsymbol{n} - (\boldsymbol{n} \cdot \boldsymbol{n})\boldsymbol{v}) \\ &= (\boldsymbol{n} \cdot \boldsymbol{v})^2 - (\boldsymbol{n} \cdot \boldsymbol{n})(\boldsymbol{v} \cdot \boldsymbol{v}) \leq 0 \end{aligned} \tag{2.13}$$

d'après l'inégalité de Cauchy-Schwartz. Ainsi, la matrice $\boldsymbol{B}$ est positive si a et b sont négatifs et liés par la relation $b + az + 1 = 0$. Ce qui équivaut à :

$$-\frac{1}{z} \leq a \leq 0, \tag{2.14a}$$

$$b = -1 - az. \tag{2.14b}$$

Nous remarquons alors que la positivité de l'impédance z est essentielle : dans le cas contraire la condition (2.14a) n'est jamais vérifiée.

La matrice $\boldsymbol{M}_{\text{SIBC}}$ s'écrit alors :

$$\boldsymbol{M}_{\text{SIBC}} = \begin{pmatrix} a(\boldsymbol{n}\times)^2 & -az\boldsymbol{n}\times \\ -(1+az)\boldsymbol{n}\times & -(1+az)z(\boldsymbol{n}\times)^2 \end{pmatrix}. \tag{2.15}$$

Le paramètre a permet d'augmenter la dissipation numérique de cette condition aux limites. Lorsqu'il est nul, la matrice $\boldsymbol{B}$ est dégénérée et un minimum d'énergie est dissipé.

2.1.4 Condition de Silver-Müller

La condition de Silver-Müller est un cas particulier de la condition SIBC vue dans la section 2.1.3 et a notamment été étudiée par Barucq *et al.* [4] et Bendali *et al.* [5] et se traduit par une condition aux limites de type SIBC avec une impédance de surface égale à celle du vide, donnée par :

$$z_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}}. \tag{2.16}$$

Dans le cas des équations de Maxwell adimensionnées que nous résoudrons dans le solveur implémenté, l'impédance du vide est égale à 1. Nous nous plaçons donc dans ce cas afin de simplifier les écritures.

En reprenant les résultats de la condition SIBC appliquée au cas particulier que nous présentons ici, nous obtenons la matrice de condition aux limites suivante :

$$\begin{pmatrix} a(\boldsymbol{n}\times)^2 & -a\boldsymbol{n}\times \\ -(1+a)\boldsymbol{n}\times & -(1+a)(\boldsymbol{n}\times)^2 \end{pmatrix}, \tag{2.17}$$

avec a le paramètre de dissipation tel que $-1 \leq a \leq 0$.

En choisissant $a = -\frac{1}{2}$, nous obtenons la matrice $\boldsymbol{M}_{\mathrm{SM}} = -\boldsymbol{A}_i {n_i}^-$. Dans ce cas, le flux associé à cette condition vaut :

$$F_{\mathrm{SM}}(\boldsymbol{w}, \boldsymbol{n}) = (\boldsymbol{A}_i n_i + \boldsymbol{M}_{\mathrm{SM}})\boldsymbol{w} = \boldsymbol{A}_i {n_i}^+ \boldsymbol{w}. \tag{2.18}$$

Considérons maintenant une condition de Silver-Müller inhomogène. Cette condition aux limites permet d'injecter dans le domaine de calcul un champ incident que nous notons $\boldsymbol{w}_I$:

$$\boldsymbol{M}_{\mathrm{SM}}(\boldsymbol{w} - \boldsymbol{w}_I) = 0. \tag{2.19}$$

Le flux associé à cette condition inhomogène s'exprime en fonction du flux obtenu dans le cas homogène :

$$\begin{aligned} F_{\mathrm{SMI}}(\boldsymbol{w}, \boldsymbol{n}) &= F_{\mathrm{SM}}(\boldsymbol{w}, \boldsymbol{n}) - \boldsymbol{M}_{\mathrm{SM}} \boldsymbol{w}_I \\ &= \boldsymbol{A}_i {n_i}^+ \boldsymbol{w} + \boldsymbol{A}_i {n_i}^- \boldsymbol{w}_I \\ &= F(\boldsymbol{w}, \boldsymbol{w}_I, \boldsymbol{n}). \end{aligned} \tag{2.20}$$

Nous retrouvons ainsi le flux numérique décentré F.

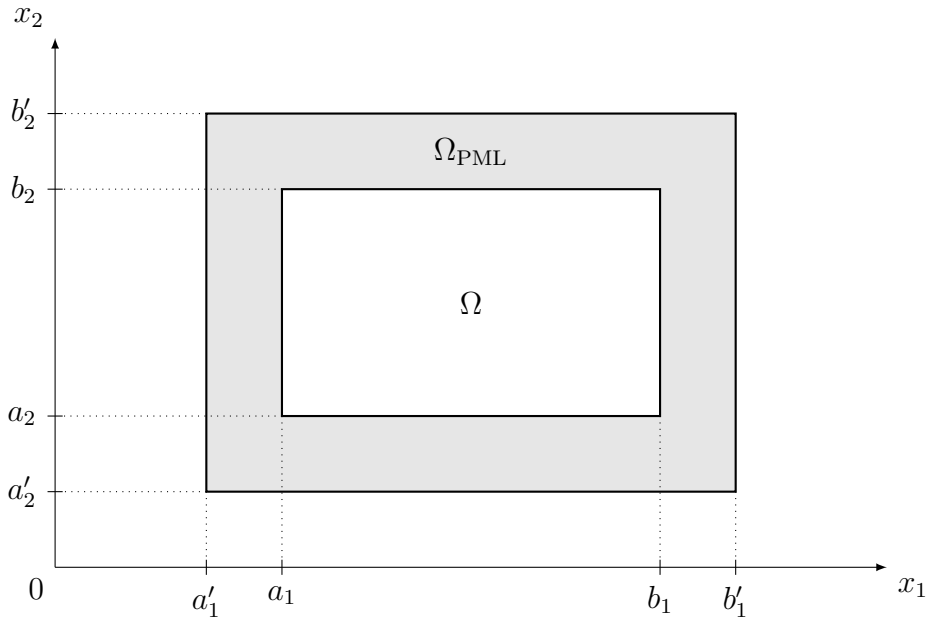
2.1.5 Condition aux limites absorbante

Pour calculer la solution des équations de Maxwell en espace libre, il est nécessaire de simuler cet espace libre à l'aide d'une condition aux limites absorbante. Sans une telle condition de bord, le domaine de calcul devrait être suffisamment grand pour s'assurer qu'aucune onde réfléchie ne revienne polluer la scène électromagnétique étudiée.

La condition aux limites de Silver-Müller vue dans la section 2.1.4, est connue pour sa facilité d'implémentation et son faible coût de calcul. Cependant, cette condition provoque des réflexions des ondes sortantes contrairement au modèle PML couramment utilisé en simulation électromagnétique.

Le modèle PML a d'abord été introduit par Bérenger [6] puis étendu et analysé théoriquement dans une vaste littérature [21, 28, 48, 49, 55–57, 70, 75]. La construction de ce modèle utilise un changement de variable complexe effectué sur la forme harmonique des équations de Maxwell. Ce modèle s'applique dans un volume englobant le domaine de calcul et consiste en l'ajout d'une conductivité électrique fictive qui a pour effet de dissiper les ondes électromagnétiques dans la dimension imaginaire du modèle.

FIGURE 2.1 – Représentation en 2 dimensions du domaine Ω_{PML} considéré, englobant le domaine de calcul Ω.



Considérons les équations de Maxwell dans le domaine de calcul $\Omega = [a_1\,;b_1] \times [a_2\,;b_2] \times [a_3\,;b_3]$ inclus dans $\mathbb{R}^3$. Soit aussi le volume englobant $\Omega_{\text{PML}} = [a'_1\,;b'_1] \times [a'_2\,;b'_2] \times [a'_3\,;b'_3]$ défini tel que dans chaque direction $a_i > a'_i$ et $b_i < b'_i$ (figure 2.1). Il est possible d'établir des modèles PML pour des domaines de formes plus complexes comme par exemple dans les travaux de Laurens [49].

Dans le domaine Ω, après transformée de Fourier-Laplace en temps du système sans sources ($\boldsymbol{C} = 0$ et $\boldsymbol{S} = 0$), nous obtenons l'expression fréquentielle des équations de Maxwell :

$$p\varepsilon\tilde{\boldsymbol{E}} - \nabla \times \tilde{\boldsymbol{H}} = 0, \tag{2.21a}$$

$$p\mu\tilde{\boldsymbol{H}} + \nabla \times \tilde{\boldsymbol{E}} = 0, \tag{2.21b}$$

avec $p = j\omega$ où j est la notation classique en physique du nombre complexe défini tel que $j^2 = -1$, f la fréquence et $\omega = 2\pi f$ la pulsation.

Remarque 3. *Le fait de considérer le système sans sources est légitime puisque nous nous situons en bordure de domaine de calcul où nous cherchons à simuler un espace libre infini.*

Nous introduisons le changement de variable complexe suivant dans le domaine Ω_{PML} :

$$\tau(x) = \hat{x} : \hat{x}_i = x_i + \frac{1}{p}\int_0^{x_i} \sigma_i(s)ds, \tag{2.22}$$

avec $\hat{x}$ un point de l'espace complexe $\mathbb{C}^3$ et $(\sigma_i)_{i\in[\![1\,;3]\!]}$ des fonctions réelles homogènes à des conductivités. Cette opération se justifie théoriquement grâce à des outils avancés de géométrie différentielle sur des variétés complexes décrits dans les travaux de Lassas *et al.* [48], Laurens [49] et Mazet *et al.* [56].

Remarquons que la matrice jacobienne τ' de τ est diagonale :

$$\tau' = \text{diag}(s_1, s_2, s_3), \text{ avec } s_i = 1 + \frac{1}{p}\sigma_i(x_i). \tag{2.23}$$

Les conductivités « fictives » σ_i sont supposées nulles sur Ω et de progression régulière sur Ω_{PML} afin d'absorber progressivement les ondes sortantes du domaine Ω. Un argument de type développement de Taylor, ou développement analytique, permet de conclure que jusqu'à un certain ordre, la solution calculée est bien la restriction à un domaine borné de la solution en milieu libre.

Nous exprimons aussi les opérateurs $\hat{\partial}_i$ de dérivation complexe en fonction des opérateurs ∂_i en effectuant le changement de variable formel suivant :

$$\hat{\partial}_i = \frac{1}{s_i}\partial_i. \tag{2.24}$$

Nous pouvons alors exprimer les équations de Maxwell sur Ω_{PML} avec les dérivées complexes formelles :

$$p\varepsilon\tilde{\boldsymbol{E}} - \hat{\nabla} \times \tilde{\boldsymbol{H}} = 0, \tag{2.25a}$$
$$p\mu\tilde{\boldsymbol{H}} + \hat{\nabla} \times \tilde{\boldsymbol{E}} = 0, \tag{2.25b}$$

qui coïncide avec le système (2.21) sur Ω. En utilisant l'expression du rotationnel complexe d'un champ $\boldsymbol{u}$:

$$\hat{\nabla} \times \boldsymbol{u} = \operatorname{jac}(\tau)^{-1} \tau'(\nabla\times)\tau'\boldsymbol{u}, \tag{2.26}$$

puis, en injectant cette formule dans le système de Maxwell complexe, nous obtenons le système perturbé :

$$p\varepsilon \operatorname{jac}(\tau)(\tau')^{-2}\tilde{\boldsymbol{E}}_s - \nabla \times \tilde{\boldsymbol{H}}_s = 0, \tag{2.27a}$$
$$p\mu \operatorname{jac}(\tau)(\tau')^{-2}\tilde{\boldsymbol{H}}_s + \nabla \times \tilde{\boldsymbol{E}}_s = 0, \tag{2.27b}$$

avec $\tilde{\boldsymbol{E}}_s = \tau'\tilde{\boldsymbol{E}}$ et $\tilde{\boldsymbol{H}}_s = \tau'\tilde{\boldsymbol{H}}$.

La matrice $\operatorname{jac}(\tau)(\tau')^{-2}$ est une matrice diagonale :

$$\operatorname{jac}(\tau)(\tau')^{-2} = \operatorname{diag}\left(\frac{s_2 s_3}{s_1}, \frac{s_3 s_1}{s_2}, \frac{s_1 s_2}{s_3}\right). \tag{2.28}$$

Nous introduisons alors les matrices suivantes :

$$\boldsymbol{\Sigma}_1 = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3, \sigma_1, \sigma_2, \sigma_3), \tag{2.29a}$$
$$\boldsymbol{\Sigma}_2 = \operatorname{diag}(\sigma_2, \sigma_3, \sigma_1, \sigma_2, \sigma_3, \sigma_1), \tag{2.29b}$$
$$\boldsymbol{\Sigma}_3 = \operatorname{diag}(\sigma_3, \sigma_1, \sigma_2, \sigma_3, \sigma_1, \sigma_2), \tag{2.29c}$$

et :

$$\boldsymbol{S}_1 = \operatorname{diag}(s_1, s_2, s_3, s_1, s_2, s_3), \tag{2.30a}$$
$$\boldsymbol{S}_2 = \operatorname{diag}(s_2, s_3, s_1, s_2, s_3, s_1), \tag{2.30b}$$
$$\boldsymbol{S}_3 = \operatorname{diag}(s_3, s_1, s_2, s_3, s_1, s_2), \tag{2.30c}$$

telles que :

$$\boldsymbol{S}_i = \boldsymbol{I} + \frac{1}{p}\boldsymbol{\Sigma}_i. \tag{2.31}$$

Nous pouvons alors écrire le système perturbé (2.27) sous la forme :

$$p\boldsymbol{A}_t(\boldsymbol{S}_1)^{-1}\boldsymbol{S}_2\boldsymbol{S}_3\tilde{\boldsymbol{w}}_s + \boldsymbol{A}_i\partial_i\tilde{\boldsymbol{w}}_s = 0, \tag{2.32}$$

avec $\tilde{\boldsymbol{w}}_s = \left(\tilde{\boldsymbol{E}}_s^{\mathrm{T}}, \tilde{\boldsymbol{H}}_s^{\mathrm{T}}\right)^{\mathrm{T}}$.

En réécrivant la matrice du premier terme :

$$p(\boldsymbol{S}_1)^{-1}\boldsymbol{S}_2\boldsymbol{S}_3 = (p\boldsymbol{I} + \boldsymbol{\Sigma}_1)^{-1}(p\boldsymbol{I} + \boldsymbol{\Sigma}_2)(p\boldsymbol{I} + \boldsymbol{\Sigma}_3), \tag{2.33}$$

et puisque toutes ces matrices sont diagonales, nous pouvons réécrire ce produit afin d'obtenir une décomposition en éléments simples. Ainsi, pour le premier terme de chaque diagonale, nous obtenons :

$$\begin{aligned}
(p+\sigma_1)^{-1}(p+\sigma_2)(p+\sigma_3) &= \frac{(p+\sigma_2)(p+\sigma_3)}{p+\sigma_1} \\
&= \frac{p^2 + (\sigma_2+\sigma_3)p + \sigma_2\sigma_3}{p+\sigma_1} \\
&= p + \frac{(\sigma_2+\sigma_3-\sigma_1)p + \sigma_2\sigma_3}{p+\sigma_1} \\
&= p + (\sigma_2+\sigma_3-\sigma_1) + \frac{-(\sigma_2+\sigma_3-\sigma_1)\sigma_1 + \sigma_2\sigma_3}{p+\sigma_1} \\
&= p + (\sigma_2+\sigma_3-\sigma_1) + \frac{\sigma_1^2 - \sigma_1\sigma_2 - \sigma_1\sigma_3 + \sigma_2\sigma_3}{p+\sigma_1} \\
&= p + (\sigma_2+\sigma_3-\sigma_1) + \frac{(\sigma_1-\sigma_2)(\sigma_1-\sigma_3)}{p+\sigma_1},
\end{aligned} \tag{2.34}$$

d'où, en raisonnant de manière analogue sur les autres termes des diagonales :

$$p(\boldsymbol{S}_1)^{-1}\boldsymbol{S}_2\boldsymbol{S}_3 = p\boldsymbol{I} + \boldsymbol{C}' + \boldsymbol{R}_1(p\boldsymbol{I} + \boldsymbol{\Sigma}_1)^{-1}\boldsymbol{R}_2, \tag{2.35}$$

avec :

$$\boldsymbol{C}' = \boldsymbol{\Sigma}_2 + \boldsymbol{\Sigma}_3 - \boldsymbol{\Sigma}_1, \tag{2.36a}$$
$$\boldsymbol{R}_1 = \boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2, \tag{2.36b}$$
$$\boldsymbol{R}_2 = \boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_3. \tag{2.36c}$$

Nous appliquons alors une transformée de Fourier-Laplace inverse au système (2.32) et obtenons un système intégro-différentiel en temps. Nous introduisons donc une variable intermédiaire $\boldsymbol{v}$ pour nous ramener à un système d'EDP en espace-temps :

$$\boldsymbol{A}_t\partial_t\boldsymbol{w} + \boldsymbol{A}_t\boldsymbol{C}'\boldsymbol{w} + \boldsymbol{A}_t\boldsymbol{R}_1\boldsymbol{v} + \boldsymbol{A}_i\partial_i\boldsymbol{w} = 0, \tag{2.37a}$$
$$\partial_t\boldsymbol{v} + \boldsymbol{\Sigma}_1\boldsymbol{v} - \boldsymbol{R}_2\boldsymbol{w} = 0. \tag{2.37b}$$

Ce système est un système de Friedrichs (définition 1) et s'écrit sous la forme :

$$\hat{\boldsymbol{A}}_t\partial_t\hat{\boldsymbol{w}} + \hat{\boldsymbol{C}}\hat{\boldsymbol{w}} + \hat{\boldsymbol{A}}_i\partial_i\hat{\boldsymbol{w}} = 0, \tag{2.38}$$

avec :

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{w}^{\mathrm{T}}, \boldsymbol{v}^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{2.39a}$$

$$\hat{\boldsymbol{A}}_t = \begin{pmatrix} \boldsymbol{A}_t & 0 \\ 0 & \boldsymbol{I} \end{pmatrix}, \tag{2.39b}$$

$$\hat{\boldsymbol{C}} = \begin{pmatrix} \boldsymbol{A}_t \boldsymbol{C}' & \boldsymbol{A}_t \boldsymbol{R}_1 \\ -\boldsymbol{R}_2 & \boldsymbol{\Sigma}_1 \end{pmatrix}, \tag{2.39c}$$

$$\hat{\boldsymbol{A}}_i = \begin{pmatrix} \boldsymbol{A}_i & 0 \\ 0 & 0 \end{pmatrix}. \tag{2.39d}$$

L'intérêt de cette formulation est qu'elle est bien adaptée au solveur que nous développons puisque ce dernier est en mesure de résoudre tout système de Friedrichs. Ce système de Friedrichs associé au modèle PML fait intervenir 12 variables conservatives (et donc des matrices carrées de taille 12) contre seulement 6 pour les équations de Maxwell seules. L'existence et l'unicité de la solution dans le domaine PML est donnée par Kato [44] et Laurens [50] dans le cas harmonique des équations de Maxwell.

De manière générale, l'application du modèle PML au bord du domaine de calcul s'effectue de manière régulière, en ajoutant un domaine englobant d'épaisseur totale $\delta_{\max} = |a_i - a_i'| = |b_i - b_i'|$ dans chaque direction sur les bords du domaine de calcul Ω (figure 2.1). Ce domaine englobant est noté Ω_{PML}.

Le choix des conductivités fictives σ_i introduites dans la construction du modèle PML (2.22) est fait afin d'obtenir une certaine régularité dans l'absorption des ondes sur le domaine Ω_{PML}. Cette régularité est importante pour assurer la « transparence » des PML.

Rappelons que ces conductivités sont nulles dans le domaine de calcul Ω. Dans le domaine Ω_{PML}, elles sont données par la formule :

$$\sigma_i(x_i) = \sigma_{\max} \left(\frac{\delta_i(x_i)}{\delta_{\max}} \right)^d \tag{2.40}$$

avec $\sigma_{\max}$ la conductivité électrique maximale du modèle PML, $d \in \mathbb{N}$ l'ordre polynomial des conductivités, $\delta_{\max}$ l'épaisseur du volume englobant et $\delta_i(x_i)$ la distance de $x_i \in \Omega_{\mathrm{PML}}$ au domaine Maxwell :

$$\delta_i(x_i) = \begin{cases} a_i - x_i & \text{si } x_i < a_i, \\ x_i - b_i & \text{si } x_i > b_i, \\ 0 & \text{sinon.} \end{cases} \tag{2.41}$$

La conductivité maximale $\sigma_{\max}$ est atteinte pour chaque composante, dans sa direction associée, uniquement aux bords du domaine Ω_{PML}. En général, nous utilisons ce modèle avec les paramètres suivants qui génèrent un très faible taux de réflexion : $d = 2$, $\sigma_{\max} = 2000$ et $\delta_{\max} = \lambda/2$, avec λ la longueur d'onde du signal, subdivisé en 5 couches de mailles hexaédriques. La condition de bord appliquée à la frontière extérieure du domaine Ω_{PML} est une condition de conducteur électrique parfait présenté dans la section 2.1.1.

2.2 Modèles de plaques minces

Les modèles de plaques minces sont utilisés en simulation numérique afin de reproduire le comportement des matériaux de faible épaisseur. Cela dans le but de supprimer du maillage les petites mailles qui produiraient un malus important en temps de simulation (section 3.3.3) et qui compliquent fortement la phase de maillage. Ces mailles fines sont alors remplacées par des faces sur lesquelles un flux spécifique est appliqué.

La validité de ces modèles dépend de la fréquence de coupure f_c des matériaux (2.44). Nous avons déjà présenté un tel modèle dans la section 2.1.3 : la condition aux limites d'impédance de surface SIBC. Nous allons le présenter ici dans sa version dépendante de la fréquence. Ce modèle est valide en hautes fréquences, pour des fréquences supérieures à $20 f_c$. Nous présentons deux autres modèles permettant de couvrir une bande de fréquence plus étendue : le modèle de plaque mince de Bérenger valide pour des fréquences inférieures à la fréquence de coupure et la condition BIBC valide sur la bande de fréquences de $f_c/10$ à $20 f_c$.

2.2.1 Plaque mince de Bérenger

Le modèle de Bérenger [10] permet de remplacer efficacement les plaques minces de matériaux pour une fréquence allant jusqu'à la fréquence de coupure f_c de ce dernier.

Soit S la surface représentant la plaque mince de matériau. Cette surface est orientée par un vecteur normal unitaire $\boldsymbol{n}$, dirigé de la maille L vers la maille R. Nous notons $\boldsymbol{E}_L^{\top}$, respectivement $\boldsymbol{E}_R^{\top}$, la composante tangentielle à la surface S du champ électrique du côté L, respectivement R. Soient aussi z_s et z_t les impédances de surface et de transfert de la plaque. Les champs

électrique et magnétique vérifient alors la relation de Léontovitch :

$$\boldsymbol{E}_L^\top + \boldsymbol{n} \times (z_s \boldsymbol{H}_L + z_t \boldsymbol{H}_R) = 0, \tag{2.42a}$$
$$\boldsymbol{E}_R^\top - \boldsymbol{n} \times (z_s \boldsymbol{H}_R + z_t \boldsymbol{H}_L) = 0. \tag{2.42b}$$

En prenant pour hypothèse que l'épaisseur de peau [1] est supérieure à l'épaisseur de la plaque, nous pouvons approcher les impédances de surface et de transfert par :

$$z_s = z_t = \frac{1}{\sigma d}, \tag{2.43}$$

avec d l'épaisseur de la plaque et σ sa conductivité. Cette approximation reste valide pour des fréquences inférieures à la fréquence de coupure :

$$f_c = \frac{1}{2\pi\mu\sigma d^2}. \tag{2.44}$$

Considérons dans un premier temps une plaque placée sur le bord du domaine de calcul et munie d'une normale orientée vers l'extérieur. En remarquant que :

$$\forall \boldsymbol{v} \in \mathbb{R}^3, \boldsymbol{v}^\top = -\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{v}), \tag{2.45}$$

la relation de Léontovitch précédente s'écrit :

$$\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E}) - z_s \boldsymbol{n} \times \boldsymbol{H} = 0. \tag{2.46}$$

Nous retrouvons alors la condition aux limites SIBC (2.15) et nous en déduisons la matrice $\boldsymbol{M}_{\mathrm{BER}}$ représentant la présente condition :

$$\boldsymbol{M}_{\mathrm{BER}} = \begin{pmatrix} a(\boldsymbol{n}\times)^2 & -az_s\boldsymbol{n}\times \\ -(1+az_s)\boldsymbol{n}\times & -(1+az_s)z_s(\boldsymbol{n}\times)^2 \end{pmatrix}. \tag{2.47}$$

La condition inhomogène s'écrit alors, pour un champ incident $\boldsymbol{w}_I$:

$$\begin{aligned} 0 &= \boldsymbol{M}_{\mathrm{BER}}(\boldsymbol{w} - \boldsymbol{w}_I) \\ &= \begin{pmatrix} a\left((\boldsymbol{n}\times)^2(\boldsymbol{E} - \boldsymbol{E}_I) - z_s \boldsymbol{n} \times (\boldsymbol{H} - \boldsymbol{H}_I)\right) \\ -(1+az_s)\left(\boldsymbol{n} \times (\boldsymbol{E} - \boldsymbol{E}_I) + z_s(\boldsymbol{n}\times)^2(\boldsymbol{H} - \boldsymbol{H}_I)\right) \end{pmatrix}, \end{aligned} \tag{2.48}$$

avec $\boldsymbol{w}_I = \left(\boldsymbol{E}_I^{\mathrm{T}}, \boldsymbol{H}_I^{\mathrm{T}}\right)^{\mathrm{T}}$.

1. L'épaisseur de la zone où se concentre le courant dans un conducteur, en surface.

Nous pouvons alors choisir le paramètre a assurant la stabilité du schéma :

$$-\frac{1}{z_s} \leq a \leq 0. \tag{2.49}$$

Par exemple, en prenant $a = 0$, nous obtenons un flux non dissipatif associé au modèle de Bérenger :

$$\begin{aligned} F_{\text{BER}}\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) &= \boldsymbol{A}_i n_i \boldsymbol{w}_L + \boldsymbol{M}_{\text{BER}}(\boldsymbol{w}_L - \boldsymbol{w}_R) \\ &= \begin{pmatrix} -\boldsymbol{n} \times \boldsymbol{H}_L \\ \boldsymbol{n} \times \boldsymbol{E}_R + z_s(\boldsymbol{n}\times)^2(\boldsymbol{H}_R - \boldsymbol{H}_L) \end{pmatrix}. \end{aligned} \tag{2.50}$$

Il est également possible d'ajouter une dissipation numérique dans le but de stabiliser le schéma en prenant a non nul.

2.2.2 Impédance de surface

Nous avons vu la condition SIBC comme condition de bord dans la section 2.1.3. Cette condition peut être implémentée pour prendre en compte une impédance de surface variable en fonction de la fréquence.

Pour cela, nous utilisons la technique du *vector fitting* [32] dans le but d'approcher l'impédance de surface par une somme de fractions rationnelles dans le domaine fréquentiel :

$$z_s(\omega) = z_s^\infty \left(1 + \sum_{k=1}^{N_p} \frac{r_k}{j\omega - p_k}\right), \tag{2.51}$$

avec z_s^∞ l'impédance de surface à la fréquence infinie, N_p le nombre de pôles, $p_k \in \mathbb{R}$ les pôles et $r_k \in \mathbb{R}$ les résidus.

Cette approximation est alors injectée dans la relation de Léontovitch (2.10) et nous obtenons la condition SIBC de la section 2.1.3 appliquée avec l'impédance à l'infini z_s^∞, à laquelle nous ajoutons N_p courants annexes :

$$\tilde{\boldsymbol{J}}_k = \frac{r_k}{j\omega - p_k} \tilde{\boldsymbol{H}}_L, \text{ pour } k \in [\![1\,;\, N_p]\!]. \tag{2.52}$$

Ces courants s'appliquent uniquement du côté considéré de la plaque, l'impédance de transfert étant nulle dans ce cas.

En revenant dans le domaine temporel en appliquant une transformée de Fourier-Laplace inverse, nous obtenons N_p équations différentielles ordinaires annexes, pour $k \in [\![1\,;\, N_p]\!]$:

$$\partial_t \boldsymbol{J}_k - p_k \boldsymbol{J}_k - r_k \boldsymbol{H}_L = 0, \tag{2.53a}$$

$$\boldsymbol{J}_k(x, 0) = 0. \tag{2.53b}$$

Ainsi, les champs $\boldsymbol{w}_L = (\boldsymbol{E}_L^{\mathrm{T}}, \boldsymbol{H}_L^{\mathrm{T}})^{\mathrm{T}}$ vérifient le système :

$$\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E}_L) - z_s^\infty \boldsymbol{n} \times \boldsymbol{H}_L = z_s^\infty \boldsymbol{n} \times \sum_{k=1}^{N_p} \boldsymbol{J}_k, \tag{2.54a}$$

$$\partial_t \boldsymbol{J}_k - p_k \boldsymbol{J}_k - r_k \boldsymbol{H}_L = 0, \text{ pour } k \in [\![1\,;\, N_p]\!]. \tag{2.54b}$$

Nous pouvons alors définir le flux SIBC à partir du flux de Bérenger inhomogène pour l'impédance z_s^∞ :

$$F_{\mathrm{SIBC}}\left(\boldsymbol{w}_L, \boldsymbol{n}\right) = F_{\mathrm{BER}}\left(\boldsymbol{w}_L, \begin{pmatrix} 0_{\mathbb{R}^3} \\ -\sum_{k=1}^{N_p} \boldsymbol{J}_k \end{pmatrix}, \boldsymbol{n}\right). \tag{2.55}$$

2.2.3 Impédance de surface bilatérale

La condition d'impédance de surface bilatérale (ou BIBC pour « *Bilateral Impedance Boundary Condition* ») est une condition SIBC pour laquelle l'impédance de transfert n'est pas nulle. Nous utilisons donc là aussi la technique du *vector fitting* [32] dans le but d'approcher les impédances de surface et de transfert par une somme de fractions rationnelles dans le domaine fréquentiel :

$$z_s(\omega) = z_s^\infty \left(1 + \sum_{k=1}^{N_p} \frac{r_k}{j\omega - p_k}\right), \tag{2.56a}$$

$$z_t(\omega) = z_t^\infty \left(1 + \sum_{k=1}^{N_p'} \frac{r_k'}{j\omega - p_k'}\right), \tag{2.56b}$$

avec z_s^∞ l'impédance de surface à la fréquence infinie, z_t^∞ l'impédance de transfert à la fréquence infinie, N_p et N_p' le nombre de pôles pour chaque approximation, $p_k, p_k' \in \mathbb{R}$ les pôles et $r_k, r_k' \in \mathbb{R}$ les résidus.

Ces approximations sont injectées dans la relation de Léontovitch (2.42a) pour obtenir les $N_p + N_p'$ courants annexes :

$$\tilde{\boldsymbol{J}}_k = \frac{r_k}{j\omega - p_k} \tilde{\boldsymbol{H}}_L, \text{ pour } k \in [\![1\,;\, N_p]\!] \tag{2.57}$$

et

$$\tilde{\boldsymbol{J}}_k' = \frac{r_k'}{j\omega - p_k'} \tilde{\boldsymbol{H}}_R, \text{ pour } k \in [\![1\,;\, N_p']\!]. \tag{2.58}$$

En revenant dans le domaine temporel en appliquant une transformée de Fourier-Laplace inverse, nous obtenons $N_p + N'_p$ équations différentielles ordinaires annexes, pour $k \in [\![1\,;\,N_p]\!]$:

$$\partial_t \boldsymbol{J}_k - p_k \boldsymbol{J}_k - r_k \boldsymbol{H}_L = 0, \tag{2.59a}$$
$$\boldsymbol{J}_k(x,0) = 0, \tag{2.59b}$$

et pour $k \in [\![1\,;\,N'_p]\!]$

$$\partial_t \boldsymbol{J}'_k - p'_k \boldsymbol{J}'_k - r'_k \boldsymbol{H}_R = 0, \tag{2.60a}$$
$$\boldsymbol{J}'_k(x,0) = 0. \tag{2.60b}$$

Ainsi, les champs $\boldsymbol{w}_L = (\boldsymbol{E}_L^{\mathrm{T}}, \boldsymbol{H}_L^{\mathrm{T}})^{\mathrm{T}}$ et $\boldsymbol{w}_R = (\boldsymbol{E}_R^{\mathrm{T}}, \boldsymbol{H}_R^{\mathrm{T}})^{\mathrm{T}}$ vérifient le système :

$$\boldsymbol{n} \times (\boldsymbol{n} \times \boldsymbol{E}_L) - z_s^\infty \boldsymbol{n} \times \boldsymbol{H}_L = z_s^\infty \boldsymbol{n} \times \hat{\boldsymbol{J}}, \tag{2.61a}$$
$$\hat{\boldsymbol{J}} = \frac{z_t^\infty}{z_s^\infty} \boldsymbol{H}_R + \frac{z_t^\infty}{z_s^\infty} \sum_{k=1}^{N'_p} \boldsymbol{J}'_k + \sum_{k=1}^{N_p} \boldsymbol{J}_k, \tag{2.61b}$$
$$\partial_t \boldsymbol{J}_k - p_k \boldsymbol{J}_k - r_k \boldsymbol{H}_L = 0, \text{ pour } k \in [\![1\,;\,N_p]\!], \tag{2.61c}$$
$$\partial_t \boldsymbol{J}'_k - p'_k \boldsymbol{J}'_k - r'_k \boldsymbol{H}_R = 0, \text{ pour } k \in [\![1\,;\,N'_p]\!]. \tag{2.61d}$$

Nous pouvons alors définir le flux BIBC à partir du flux de Bérenger inhomogène pour l'impédance z_s^∞ :

$$F_{\mathrm{BIBC}}(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}) = F_{\mathrm{BER}}\left(\boldsymbol{w}_L, \begin{pmatrix} 0_{\mathbb{R}^3} \\ -\hat{\boldsymbol{J}} \end{pmatrix}, \boldsymbol{n}\right). \tag{2.62}$$

2.3 Injection d'ondes

La méthode GD que nous avons présentée dans le chapitre 1 permet de résoudre les équations de Maxwell (1.1) à partir d'un état initial. De manière générale, lorsque nous cherchons la solution d'un problème d'application industrielle (émission d'un signal par une antenne), la valeur des champs à l'instant initial est nulle. Or, les équations de Maxwell préservent les états constants (une solution initiale nulle reste nulle au cours du temps). Nous avons donc besoin d'outils permettant de perturber les champs présents dans la scène.

2.3.1 Injection d'un champ incident

Nous commençons par présenter une condition surfacique permettant l'injection d'un champ incident dans le volume de calcul. Taflove et Hagness [71] font une description de ce type de source électromagnétique dans le cadre de la méthode des différences finies. Définissons dans un premier temps, les solutions dites d'ondes planes des équations de Maxwell.

Proposition 5. *Soit $g : [0\,;T] \to \mathbb{R}$ une fonction $\mathcal{C}^1$ définissant l'amplitude d'une onde plane au cours du temps. Soient $\boldsymbol{E}_0$, $\boldsymbol{H}_0$ et $\boldsymbol{k}$ trois vecteurs formant une base orthogonale directe $(O, \boldsymbol{E}_0, \boldsymbol{H}_0, \boldsymbol{k})$ de $\mathbb{R}^3$ tels que $\|\boldsymbol{k}\| = 1$ et $\|\boldsymbol{H}_0\| = \sqrt{\varepsilon/\mu}\,\|\boldsymbol{E}_0\|$. Les champs électrique et magnétique dans cette base :*

$$\boldsymbol{E}(x', t) = g(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{E}_0, \tag{2.63a}$$

$$\boldsymbol{H}(x', t) = g(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{H}_0, \tag{2.63b}$$

sont solutions des équations de Maxwell sans sources ($\sigma = \sigma^\star = 0$ et $\boldsymbol{J} = 0$).

Démonstration. Par définition de l'orthogonalité, nous avons :

$$\boldsymbol{H}_0 \times \boldsymbol{k} = \sqrt{\frac{\varepsilon}{\mu}}\boldsymbol{E}_0. \tag{2.64}$$

Considérons l'équation de Maxwell-Ampère (1.1a) :

$$\begin{aligned}
&\varepsilon\partial_t \boldsymbol{E} - \nabla \times \boldsymbol{H} \\
&\quad = \varepsilon\partial_t g(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{E}_0 - \nabla \times \left(g(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{H}_0\right) \\
&\quad = \varepsilon g'(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{E}_0 - \nabla g(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'}) \times \boldsymbol{H}_0 \\
&\quad = \varepsilon g'(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{E}_0 + \sqrt{\varepsilon\mu} g'(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})\boldsymbol{k} \times \boldsymbol{H}_0 \\
&\quad = g'(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'})(\varepsilon\boldsymbol{E}_0 + \sqrt{\varepsilon\mu}\boldsymbol{k} \times \boldsymbol{H}_0) \\
&\quad = g'(t - \sqrt{\varepsilon\mu}\boldsymbol{k} \cdot \overrightarrow{Ox'}) \left(\varepsilon\boldsymbol{E}_0 - \sqrt{\varepsilon\mu}\sqrt{\frac{\varepsilon}{\mu}}\,\|\boldsymbol{E}_0\|\right) \\
&\quad = 0.
\end{aligned} \tag{2.65}$$

De manière analogue, l'équation de Maxwell-Faraday (1.1b) est vérifiée. □

Soit S une surface fermée, incluse dans le domaine de calcul Ω. Nous appelons zone de champ total le volume délimité par cette surface et zone de champ diffracté son complémentaire. La surface S qui sépare les deux zones est appelée surface de Huygens.

Nous définissons alors un flux numérique appliqué sur cette surface et permettant d'injecter une onde plane dans la zone de champ total du domaine et laissant sortir les ondes diffractées. En appliquant le principe de superposition, nous définissons deux flux s'appliquant de part et d'autre de la surface de Huygens :

$$F_{\text{interne}}\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) = F\left(\boldsymbol{w}_L, \boldsymbol{w}_R + \boldsymbol{w}_I, \boldsymbol{n}\right), \tag{2.66a}$$

$$F_{\text{externe}}\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) = F\left(\boldsymbol{w}_L, \boldsymbol{w}_R - \boldsymbol{w}_I, \boldsymbol{n}\right), \tag{2.66b}$$

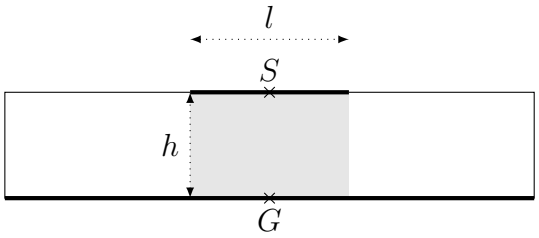
avec $\boldsymbol{w}_I$ le champ incident associé à l'onde plane injectée.

2.3.2 Générateur de Thévenin

Le modèle de Thévenin représente un générateur de tension parfait. Un tel générateur est déterminé par la donnée d'une tension notée V_T et une résistance notée R_T.

Ce modèle est utilisé pour injecter un courant au niveau d'une antenne de type ligne microruban (figure 2.2). Nous injectons ce courant en générant une tension entre la ligne microruban (S pour « *strip* ») et le plan de masse (G pour « *ground* ») de l'antenne.

FIGURE 2.2 – Schéma d'une antenne de type ligne microruban. Vue du côté de l'alimentation électrique.



La surface du générateur comprise entre la ligne microruban et le plan de masse possède une impédance réelle dont la valeur est donnée par la résistance R_T du générateur de Thévenin :

$$z_s = \frac{l R_T}{h}. \tag{2.67}$$

Nous représentons cette impédance à l'aide du modèle de plaque mince de Bérenger (section 2.2.1).

À cette impédance s'ajoute la tension V_T entre les deux surfaces métalliques de l'antenne représentées par la condition PEC (section 2.1.1). La direction du vecteur champ électrique appliqué entre ces deux bandes est donnée par :

$$\boldsymbol{E}_0 = \frac{\overrightarrow{GS}}{hl}, \tag{2.68}$$

qui correspond à l'application d'une tension de 1 V sur la surface, puisque appliquée sur toute la largeur l. Le champ électrique incident, tangent à la surface, est alors donné par :

$$\boldsymbol{w}_I = \left(\frac{V_T}{R_T}\boldsymbol{E}_0^{\mathrm{T}}, 0_{\mathbb{R}^3}^{\mathrm{T}}\right)^{\mathrm{T}}. \tag{2.69}$$

Nous injectons alors ce champ sur la surface à l'aide du flux non dissipatif associé au modèle de Bérenger :

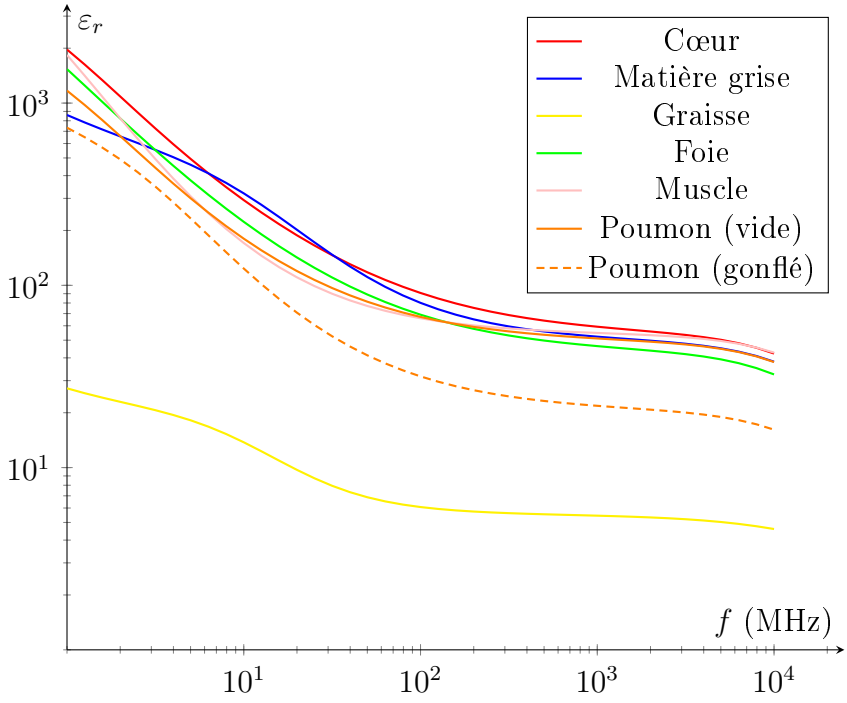
$$F_{\mathrm{THE}}\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) = F_{\mathrm{BER}}\left(\boldsymbol{w}_L, \boldsymbol{w}_R + \boldsymbol{w}_I, \boldsymbol{n}\right). \tag{2.70}$$

2.4 Permittivité et matériaux dispersifs

La permittivité diélectrique ε d'un matériau présente dans les équations de Maxwell influe sur la propagation d'une onde électromagnétique en affectant sa vitesse de propagation et en causant un phénomène de réfraction-réflexion aux interfaces entre deux matériaux de permittivité distincte. Dans le cas d'un matériau linéaire, homogène, isotrope, et avec réponse instantanée, la permittivité peut être considérée comme étant une constante réelle. C'est le cas dans le vide et, par convention, nous noterons sa permittivité $\varepsilon_0 = 8,854187 \cdot 10^{-12}$ F.m^{-1}. Mais de manière générale, la permittivité est fonction de nombreux paramètres physiques comme la fréquence du champ électromagnétique considéré ou bien la température et l'humidité du matériau.

Les travaux de Gabriel [25–27] mettent en évidence l'influence de la fréquence f sur la permittivité des tissus humains dans la bande de 10 Hz à 20 GHz. Ces travaux sont très intéressants en ce qui concerne les objets connectés qui rayonnent généralement dans une bande de 1 à 3 GHz.

FIGURE 2.3 – Permittivité relative de différents tissus humains sur la bande de fréquences de 1 MHz à 10 GHz. Données issues de la base de données IFAC-CNR.



Dans la suite, nous ferons référence à la permittivité relative (figure 2.3) d'un matériau $\varepsilon_r(f) : \mathbb{R}^+ \to \mathbb{R}^+$ en fonction de la permittivité absolue $\varepsilon(f) : \mathbb{R}^+ \to \mathbb{R}^+$ dont l'expression est donnée par la formule :

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_0}. \tag{2.71}$$

Remarque 4. *De manière analogue, la perméabilité magnétique relative μ_r d'un matériau est donnée en fonction de sa perméabilité absolue et de la perméabilité du vide $\mu_0 = 4\pi \cdot 10^{-7}$ H.m^{-1} par la formule :*

$$\mu_r = \frac{\mu}{\mu_0}. \tag{2.72}$$

Remarque 5. *Si nous notons $c = 2.99792458 \cdot 10^8$ m.s^{-1} la vitesse de propagation d'une onde électromagnétique dans le vide, alors c_r, la vitesse de propagation dans un matériau diélectrique, est donnée par :*

$$c_r = \frac{c}{\sqrt{\varepsilon_r \mu_r}}. \tag{2.73}$$

Le solveur `teta-clac` résout les équations de Maxwell dans le domaine temporel. Nous ne pouvons donc pas directement prendre en compte les permittivités relatives dépendantes de la fréquence dans les calculs. Ainsi, nous utilisons un modèle décomposant ces permittivités en sommes de fractions rationnelles tel que les conditions SIBC (section 2.2.2) et BIBC (section 2.2.3) pour lesquelles nous décomposons l'impédance en somme de fractions rationnelles. Le modèle *Complex-Conjugate Poles-Residues* (CCPR) a été choisi [34, 59]. Il s'agit d'une généralisation des modèles de Debye et Lorentz [71]. L'expression du modèle CCPR est la suivante :

$$\varepsilon(\omega) = \varepsilon_0 \varepsilon_r(\omega) = \varepsilon_0 \varepsilon_r^\infty + \varepsilon_0 \sum_{k=1}^{N_p} \frac{r_k}{j\omega - p_k} + \frac{\overline{r_k}}{j\omega - \overline{p_k}} \tag{2.74}$$

avec ε_r^∞ la permittivité relative à la fréquence infinie, N_p le nombre de pôles, $p_k \in \mathbb{C}$ les pôles, $r_k \in \mathbb{C}$ les résidus et $\overline{[.]}$ l'opérateur de conjugaison complexe. Les pôles et résidus sont obtenus à l'aide de la technique du *vector fitting* [32] appliquée à la fonction permittivité relative du matériau.

En écrivant l'équation de Maxwell-Ampère (1.1a) dans le domaine fréquentiel, nous avons :

$$j\omega\varepsilon(\omega)\tilde{\boldsymbol{E}} + \sigma\tilde{\boldsymbol{E}} - \nabla \times \tilde{\boldsymbol{H}} = 0. \tag{2.75}$$

Puis, en substituant $\varepsilon(\omega)$ dans cette équation par l'expression du CCPR (2.74), nous obtenons :

$$j\omega\varepsilon_0 \left(\varepsilon_r^\infty + \sum_{k=1}^{N_p} \frac{r_k}{j\omega - p_k} + \frac{\overline{r_k}}{j\omega - \overline{p_k}} \right) \tilde{\boldsymbol{E}} + \sigma\tilde{\boldsymbol{E}} - \nabla \times \tilde{\boldsymbol{H}} = 0, \tag{2.76}$$

soit encore :

$$j\omega\varepsilon_0\varepsilon_r^\infty \tilde{\boldsymbol{E}} + \left(\sigma + \varepsilon_0 \sum_{k=1}^{N_p} (r_k + \overline{r_k}) \right) \tilde{\boldsymbol{E}} - \nabla \times \tilde{\boldsymbol{H}} = -\sum_{k=1}^{N_p} (\tilde{\boldsymbol{J}}_k + \tilde{\boldsymbol{J}}'_k) \tag{2.77}$$

avec, pour $k \in [\![1\,;\, N_p]\!]$:

$$\tilde{\boldsymbol{J}}_k = \frac{\varepsilon_0 r_k p_k}{j\omega - p_k} \tilde{\boldsymbol{E}}, \text{ et } \tilde{\boldsymbol{J}}'_k = \frac{\varepsilon_0 \overline{r_k p_k}}{j\omega - \overline{p_k}} \tilde{\boldsymbol{E}}. \tag{2.78}$$

En réécrivant les équations (2.77) et (2.78) dans le domaine temporel (par transformée de Fourier-Laplace inverse), et en remarquant que les champs

électromagnétiques sont toujours réels (les courants annexes $\boldsymbol{J}_k$ et $\boldsymbol{J}'_k$ vérifient $\boldsymbol{J}_k = \boldsymbol{J}'_k$), nous obtenons après calculs le système différentiel suivant :

$$\partial_t \varepsilon_0 \varepsilon_r^\infty \boldsymbol{E} + (\sigma + 2\varepsilon_0 \sum_{k=1}^{N_p} \Re(r_k))\boldsymbol{E} - \nabla \times \boldsymbol{H} = -2 \sum_{k=1}^{N_p} \Re(\boldsymbol{J}_k), \tag{2.79a}$$

$$\partial_t \boldsymbol{J}_k - p_k \boldsymbol{J}_k - \varepsilon_0 r_k p_k \boldsymbol{E} = 0, \text{ pour } k \in [\![1\,;\, N_p]\!], \tag{2.79b}$$

avec $\Re$ la fonction partie réelle.

Nous obtenons alors un couplage entre le schéma GD et N_p courants vectoriels complexes annexes vérifiant chacun une équation différentielle ordinaire. Ce modèle est bien défini et stable si les parties réelles des pôles p_k sont négatives.

Conclusion

Dans ce second chapitre, nous avons présenté plusieurs modèles mathématiques de simulation. Nous avons commencé par décrire des conditions de bord respectant la condition de décroissance de l'énergie associée à la solution assurant l'existence de la solution. Nous avons aussi présenté la construction d'une condition aux limites simulant un domaine infini par l'adjonction d'un domaine englobant sur lequel les équations de Maxwell sont résolues dans l'espace complexe. Ce problème complexe est aussi un système de Friedrichs.

Ensuite, nous avons énuméré un ensemble de modèles permettant de simuler les plaques minces de matériaux. Ces modèles permettent de remplacer les volumes fins par des surfaces et sont complémentaires sur une large bande de fréquences. Le remplacement des mailles fines par des surfaces va nous permettre de réduire les temps de simulation et faciliter les phases de maillage.

Nous avons aussi présenté des modèles permettant d'injecter un signal dans la scène simulée, notamment utilisé dans les simulations faisant intervenir une antenne.

Enfin, dans le but de traiter les matériaux diélectriques dont la permittivité varie en fonction de la fréquence, nous avons donné un modèle volumique prenant en compte ces variations.

Dans le chapitre suivant, nous décrivons une implémentation sur mailles hexaédriques de la formulation semi-discrète présentée au chapitre 1. Cette implémentation a nécessité le choix d'un espace d'approximation dont nous décrivons les propriétés et les simplifications qu'elles induisent.

Chapitre 3

Implémentation

Dans les chapitres précédents, nous avons commencé par décrire la méthode Galerkin Discontinue (GD) appliquée aux équations de Maxwell écrites sous forme d'un système de Friedrichs. Rappelons que cette méthode s'applique à tout système de lois de conservation physique. Nous avons ensuite présenté des conditions aux limites ainsi que des modèles mathématiques de simulation permettant de modéliser des matériaux spécifiques tels que les conducteurs parfaits (métaux) et les plaques minces de diélectriques, ou de calculer la solution des équations de Maxwell en domaine infini simulé à l'aide de conditions aux limites absorbantes.

Cette méthode peut théoriquement être appliquée à tout type de maillage. Nous étudions ici le cas des maillages composés de mailles hexaédriques à arêtes droites. L'utilisation des hexaèdres a par exemple été étudiée par Cohen *et al.* [15]. Une autre possibilité est l'utilisation des mailles tétraédriques qui a notamment été étudiée par Hesthaven *et al.* [38] et Lanteri *et al.* [53]. Les hexaèdres permettent diverses optimisations de la formulation GD et sont bien adaptés à la parallélisation sur processeur graphique.

Dans ce chapitre, nous introduisons l'espace d'approximation dans lequel nous chercherons la solution numérique de notre problème. Cet espace est engendré par des fonctions de base à support inclus dans une unique maille. Le choix de ces fonctions de base est varié. Nous pouvons citer par exemple les travaux de Kopriva *et al.* [46], Lu *et al.* [54], Cheong *et al.* [13] et Cohen *et al.* [15]. Dans le cas du solveur `teta-clac` que nous avons implémenté, nous utilisons une base de polynômes dite nodale. Cette base est constituée de produits tensoriels de polynômes d'interpolation de Lagrange. Les points d'interpolation choisis sont les points d'intégration de Gauss-Legendre.

Avec l'utilisation combinée des hexaèdres, ce choix de polynômes et de points d'interpolation permet plusieurs optimisations (simplifications au sein

des intégrales) qui rendent la méthode GD particulièrement efficace. Dans le cas du solveur `schnaps` que nous évoquerons au chapitre 6, les points d'intégration choisis sont ceux de Gauss-Lobatto.

Nous finissons par décrire les schémas d'intégration en temps actuellement les plus souvent utilisés par le solveur : les schémas de type Runge-Kutta. Nous donnons aussi 2 diagnostics permettant d'étudier la stabilité du schéma couplé espace-temps.

3.1 Élément de référence

Dans un premier temps, nous construisons l'espace d'approximation dans lequel nous cherchons la solution numérique de notre problème. La construction de cet espace s'appuie sur la définition d'un élément de référence, ici le cube unité, noté $\hat{H} = [0\,;1]^3$. Nous notons $\hat{x} = (\hat{x}_1, \hat{x}_2, \hat{x}_3)$ un point de cet espace. Une transformation géométrique nous permettra de passer du cube de référence à une maille hexaédrique L.

Les fonctions de base sont alors constituées de compositions entre cette transformation et des fonctions polynomiales définies sur $\hat{H}$. Ces fonctions polynomiales sont des produits tensoriels de polynômes de Lagrange associés aux points d'intégration de Gauss-Legendre. Ces derniers nous serviront également à définir des formules de quadrature permettant d'estimer les intégrales volumiques et surfaciques apparaissant dans la méthode GD.

Ces choix de fonctions de base et de formules de quadrature permettent diverses simplifications de la formulation GD discrète.

Dans la suite, nous noterons $d \in \mathbb{N}^\star$, l'ordre d'approximation du schéma.

Remarque 6. *Dans cette section, nous ferons commencer la numérotation des indices à partir de l'indice 0. Ce choix permet certaines simplifications d'écriture et correspond aussi à la numérotation du langage C++ utilisé pour programmer le solveur.*

3.1.1 Points d'intégration

Soient $(\xi_p)_{p\in[\![0\,;d]\!]}$ et $(\lambda_p)_{p\in[\![0\,;d]\!]}$ les points et poids de quadrature de Gauss-Legendre de degré d sur l'intervalle $[0\,;1]$. Nous allons passer de la dimension 1 à la dimension 3 par des techniques de produits tensoriels. À tout multi-indice (p_0, p_1, p_2) d'entiers compris entre 0 et d nous pouvons associer de manière bijective un unique entier $i = p_0 + (d+1)(p_1 + (d+1)p_2)$ compris

entre 0 et $(d+1)^3-1$. Des divisions euclidiennes permettent de retrouver le multi-indice (p_0, p_1, p_2) de manière unique à partir de i.

Nous pouvons alors définir $(d+1)^3$ points d'intégration volumiques :

$$\hat{g}_i = (\xi_{p_0}, \xi_{p_1}, \xi_{p_2}) \in \hat{H} \tag{3.1}$$

et les $(d+1)^3$ poids d'intégration associés :

$$\omega_i = \lambda_{p_0}\lambda_{p_1}\lambda_{p_2} \in \mathbb{R}^{\star+}. \tag{3.2}$$

Nous approcherons alors les intégrales sur $\hat{H}$ par la formule de quadrature en dimension 3 découlant directement de la formule de quadrature sur $[0\,;1]$:

$$\int_{\hat{H}} \psi(\hat{x})d\hat{x} \approx \sum_{i=0}^{(d+1)^3-1} \omega_i \psi(\hat{g}_i). \tag{3.3}$$

En découle aussi la propriété suivante :

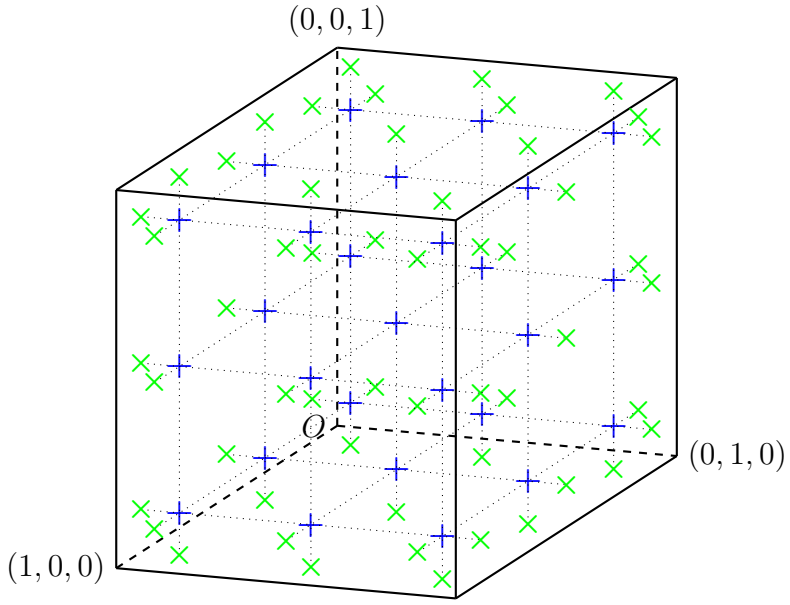
Proposition 6. *La formule de quadrature* (3.3) *associée aux points* $(\hat{g}_i)$ *et poids* (ω_i) *est exacte pour des produits tensoriels de polynômes de degré au plus* $2d+1$.

Ajoutons à ces points d'intégration, des points d'intégration situés sur les faces du cube $\hat{H}$. Notons $(\hat{Q}_f)_{f\in[\![0\,;5]\!]}$ les 6 faces quadrangulaires du cube $\hat{H}$. Nous avons choisi la numérotation suivante des faces du cube de référence :

$$\begin{aligned}
\hat{Q}_0 &= [0\,;1] \times \{0\} \times [0\,;1],\\
\hat{Q}_1 &= \{1\} \times [0\,;1] \times [0\,;1],\\
\hat{Q}_2 &= [0\,;1] \times \{1\} \times [0\,;1],\\
\hat{Q}_3 &= \{0\} \times [0\,;1] \times [0\,;1],\\
\hat{Q}_4 &= [0\,;1] \times [0\,;1] \times \{0\},\\
\hat{Q}_5 &= [0\,;1] \times [0\,;1] \times \{1\}.
\end{aligned} \tag{3.4}$$

Nous définissons sur la face $\hat{Q}_f$, $(d+1)^2$ points d'intégration surfaciques notés $(\hat{g}_{f,i})$. Ces points sont numérotés de 0 à $(d+1)^2-1$ sur chaque face. Comme précédemment cette numérotation permet de retrouver facilement les indices des points unidimensionnels à partir de divisions euclidiennes. Ces points sont les projetés sur les faces des points d'intégration dans le volume (figure 3.1).

FIGURE 3.1 – Points d'intégration volumiques (« + » bleus) et surfaciques (« × » verts) de l'élément de référence $\hat{H}$ pour $d = 2$. Les alignements entre ces points sont représentés par les lignes pointillées.



Afin de les définir rigoureusement, nous commençons par considérer le tableau suivant, donnant pour chaque face les directions des deux axes tangents à la face puis de l'axe normal à la face. La dernière colonne du tableau correspond à la valeur sur la face de la dernière coordonnée :

$$\sigma_{\text{ori}} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 2 & 1 \end{pmatrix}. \tag{3.5}$$

Par exemple, pour la face 0 (*i.e.* la première ligne du tableau), les axes tangents correspondent à ceux des composantes 0 et 2, la direction normale correspond à l'axe de la composante 1 et la coordonnée sur cet axe normal est $\hat{x}_1 = 0$.

Considérons maintenant l'indice i compris entre 0 et $(d+1)^2 - 1$ d'un point de face. Nous pouvons trouver de façon unique deux indices p_0 et p_1 compris entre 0 et d tels que $i = p_0 + (d+1)p_1$. Alors le point correspondant

sur la face f est donné par :

$$\hat{g}_{f,i} = (\alpha_0, \alpha_1, \alpha_2) \tag{3.6}$$

avec

$$\alpha_{\sigma_{\text{ori}}(f,0)} = \xi_{p_0}, \quad \alpha_{\sigma_{\text{ori}}(f,1)} = \xi_{p_1}, \quad \alpha_{\sigma_{\text{ori}}(f,2)} = \sigma_{\text{ori}}(f,3). \tag{3.7}$$

À ce point d'intégration correspond le poids d'intégration suivant :

$$\omega_{f,i} = \lambda_{p_0}\lambda_{p_1}. \tag{3.8}$$

Ces points et poids d'intégration définissent une formule de quadrature permettant d'approcher les intégrales sur les faces de l'élément $\hat{H}$:

$$\int_{\hat{Q}_f} \psi(\hat{x})d\hat{x} \approx \sum_{i=0}^{(d+1)^2-1} \omega_{f,i}\psi(\hat{g}_{f,i}). \tag{3.9}$$

Grâce aux propriétés de quadrature de Gauss-Legendre, nous avons la propriété suivante :

Proposition 7. *La formule de quadrature* (3.9) *associée aux points* $(\hat{g}_{f,i})$ *et poids* $(\omega_{f,i})$ *est exacte pour des produits tensoriels de polynômes de degrés au plus* $2d+1$.

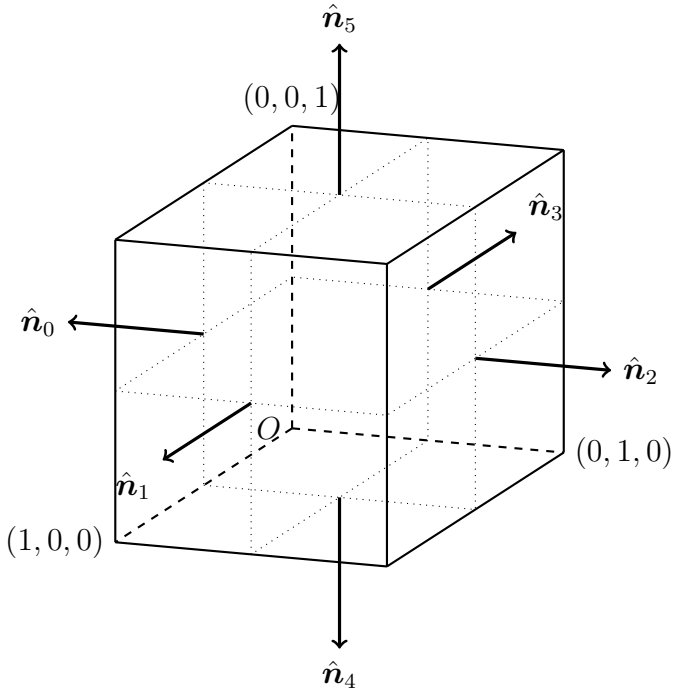
Nous pouvons encore définir le projeté sur la face $\hat{Q}_f$ d'un point d'intégration volumique $\hat{g}_i$, avec $i = p_0 + (d+1)(p_1 + (d+1)p_2)$. Cette projection nous permettra d'exprimer des simplifications qui apparaîtront lors de l'extrapolation des champs sur les faces. Il s'agit du point $\hat{g}_{f,j}$ dont l'indice j peut être calculé à l'aide de l'application π définie telle que :

$$\begin{aligned} j &= \pi(f,i) \\ &= \pi(f, p_0 + (d+1)(p_1 + (d+1)p_2)) \\ &= p_{\sigma_{\text{ori}}(f,0)} + (d+1)p_{\sigma_{\text{ori}}(f,1)}. \end{aligned} \tag{3.10}$$

Enfin, nous munissons les faces de l'élément de référence d'une normale unitaire sortante notée $\hat{\boldsymbol{n}}$ (figure 3.2) :

$$\hat{\boldsymbol{n}} = \begin{cases} (0,-1,0)^{\mathrm{T}} & \text{sur la face } 0, \\ (1,0,0)^{\mathrm{T}} & \text{sur la face } 1, \\ (0,1,0)^{\mathrm{T}} & \text{sur la face } 2, \\ (-1,0,0)^{\mathrm{T}} & \text{sur la face } 3, \\ (0,0,-1)^{\mathrm{T}} & \text{sur la face } 4, \\ (0,0,1)^{\mathrm{T}} & \text{sur la face } 5. \end{cases} \tag{3.11}$$

FIGURE 3.2 – Orientation des normales des faces de l'élément de référence. La normale à la face $\hat{Q}_f$ est notée $\hat{\boldsymbol{n}}_f$.



Après avoir défini l'espace de référence, nous allons le munir d'une base de fonctions qui va nous permettre d'exprimer la solution et de la stocker sous forme discrète.

3.1.2 Fonctions de base

Considérons maintenant les polynômes de Lagrange $(l_p)_{p\in[\![0\,;d]\!]}$ associés aux points $(\xi_p)_{p\in[\![0\,;d]\!]}$. Ces polynômes sont donnés par :

$$l_p(\xi) = \prod_{\substack{q=0 \\ q\neq p}}^{d} \frac{\xi - \xi_q}{\xi_p - \xi_q} \tag{3.12}$$

et vérifient :

$$l_p(\xi_q) = \delta_{p,q} \tag{3.13}$$

où $\delta_{p,q}$ est le symbole de Kronecker qui vaut 1 si $p = q$ et 0 sinon.

Nous définissons la base d'approximation sur $\hat{H}$ comme étant formée des produits tensoriels des polynômes (l_p). Plus précisément, ces fonctions de base sont données par :

$$\begin{aligned} \hat{\psi}_i : \hat{H} &\to \mathbb{R} \\ \hat{x} &\mapsto l_{p_0}(\hat{x}_0) l_{p_1}(\hat{x}_1) l_{p_2}(\hat{x}_2) \end{aligned} \tag{3.14}$$

où $i = p_0 + (d+1)(p_1 + (d+1)p_2)$.

Proposition 8. *La formule de quadrature (3.3) est exacte pour des combinaisons linéaires de produits de fonctions de base décrites plus haut. De plus,*

$$\int_{\hat{H}} \sum_{i=0}^{(d+1)^3-1} \sum_{j=0}^{(d+1)^3-1} a_{i,j} \hat{\psi}_i \hat{\psi}_j d\hat{x} = \sum_{i=0}^{(d+1)^3-1} a_{i,i} \omega_i. \tag{3.15}$$

Remarque 7. *La formule de quadrature (3.3) est exacte, car nous utilisons des points de Gauss-Legendre. Si nous utilisions des points de Gauss-Lobatto, la proposition précédente ne serait plus vraie.*

La proposition suivante nous permet de simplifier l'extrapolation des champs sur les points d'intégration surfaciques. Elle nous donne le sous-ensemble des $d+1$ fonctions de base nécessaires à l'extrapolation des champs sur un point de face donné. Sans cette simplification, l'extrapolation des champs nécessiterait d'utiliser les $(d+1)^3$ fonctions de base de la maille pour chaque point d'intégration surfacique.

Proposition 9. *Une fonction de base donnée est nulle en tous points d'intégration surfaciques d'une face, excepté celui correspondant au projeté du point volumique correspondant à cette fonction de base. Autrement dit,*

$$\hat{\psi}_i(\hat{g}_{f,j}) = \begin{cases} l_{p_{\sigma_{\text{ori}}(f,2)}}(\sigma_{\text{ori}}(f,3)) & \text{si } j = \pi(f,i), \\ 0 & \text{sinon.} \end{cases} \tag{3.16}$$

Démonstration. Soit $\hat{\psi}_i$ une fonction de base et $\hat{g}_i = (\xi_{p_0}, \xi_{p_1}, \xi_{p_2})$ le point d'interpolation correspondant. Soit $\hat{g}_{f,j}$ un point d'intégration sur la face f avec $j = q_0 + (d+1)q_1$. Alors, en ce point la fonction de base vaut :

$$\begin{aligned} \hat{\psi}_i(\hat{g}_{f,j}) &= l_{p_{\sigma_{\text{ori}}(f,0)}}(\xi_{q_0}) l_{p_{\sigma_{\text{ori}}(f,1)}}(\xi_{q_1}) l_{p_{\sigma_{\text{ori}}(f,2)}}(\sigma_{\text{ori}}(f,3)) \\ &= \delta_{p_{\sigma_{\text{ori}}(f,0)},q_0} \delta_{p_{\sigma_{\text{ori}}(f,1)},q_1} l_{p_{\sigma_{\text{ori}}(f,2)}}(\sigma_{\text{ori}}(f,3)) \\ &= \delta_{\pi(f,i),j} l_{p_{\sigma_{\text{ori}}(f,2)}}(\sigma_{\text{ori}}(f,3)). \end{aligned} \tag{3.17}$$

□

Nous allons maintenant définir la transformation géométrique qui nous permettra de passer de la base d'approximation de référence que nous venons de définir aux mailles physiques présentes dans le maillage étudié.

3.1.3 Transformation géométrique

Soit $(\hat{S}_i)_{i\in[\![0\,;7]\!]}$ les huit sommets du cube unité $\hat{H}$ donnés par :

$$\begin{aligned} \hat{S}_0 &= (0,0,0)\,, & \hat{S}_1 &= (1,0,0)\,, \\ \hat{S}_2 &= (0,1,0)\,, & \hat{S}_3 &= (1,1,0)\,, \\ \hat{S}_4 &= (0,0,1)\,, & \hat{S}_5 &= (1,0,1)\,, \\ \hat{S}_6 &= (0,1,1)\,, & \hat{S}_7 &= (1,1,1)\,. \end{aligned} \tag{3.18}$$

Nous définissons les huit fonctions de forme associées à chacun de ces sommets :

$$\begin{aligned} \phi_0(\hat{x}) &= (1-\hat{x}_0)(1-\hat{x}_1)(1-\hat{x}_2), \\ \phi_1(\hat{x}) &= \hat{x}_0(1-\hat{x}_1)(1-\hat{x}_2), \\ \phi_2(\hat{x}) &= (1-\hat{x}_0)\hat{x}_1(1-\hat{x}_2), \\ \phi_3(\hat{x}) &= \hat{x}_0\hat{x}_1(1-\hat{x}_2), \\ \phi_4(\hat{x}) &= (1-\hat{x}_0)(1-\hat{x}_1)\hat{x}_2, \\ \phi_5(\hat{x}) &= \hat{x}_0(1-\hat{x}_1)\hat{x}_2, \\ \phi_6(\hat{x}) &= (1-\hat{x}_0)\hat{x}_1\hat{x}_2, \\ \phi_7(\hat{x}) &= \hat{x}_0\hat{x}_1\hat{x}_2. \end{aligned} \tag{3.19}$$

Chacune des ces fonctions de forme prend une valeur nulle sur chacun des sommets du cube unité excepté au point lui correspondant, *i.e.* :

$$\phi_i(\hat{S}_j) = \delta_{i,j}. \tag{3.20}$$

Soit L un hexaèdre appartenant au maillage $\mathcal{M}$. Cet hexaèdre est défini par ses huit sommets $(S_i)_{i\in[\![0\,;7]\!]}$. Nous appellerons point physique et noterons x un point situé dans la maille L et nous appellerons point de référence et noterons $\hat{x}$ un point de $\hat{H}$.

Nous définissons alors la transformation géométrique τ_L telle que l'image du cube unité soit l'hexaèdre s'appuyant sur les huit sommets $(S_i)_{i\in[\![0\,;7]\!]}$. Cette transformation est donnée par la formule classique, faisant intervenir les fonctions de forme :

$$\tau_L(\hat{x}) = \sum_{i=0}^{7} \phi_i(\hat{x}) S_i. \tag{3.21}$$

Nous supposerons que cette transformation est une bijection entre $\hat{H}$ et L, et que son jacobien

$$\mathrm{jac}\,(\tau_L) = \det\,(\tau_L') \tag{3.22}$$

est strictement positif en tout point de $\hat{H}$.

Remarque 8. *Contrairement aux maillages en tétraèdres, il peut arriver qu'un mailleur produise exceptionnellement des hexaèdres dégénérés, c'est-à-dire que le jacobien de τ_L s'annule ou change de signe en certains points de L. Il est important de vérifier, avant de lancer un calcul, que le maillage ne présente pas de maille dégénérée. Pour cela, nous pouvons calculer le jacobien $\mathrm{jac}\,(\tau_L)$ aux points d'intégration et vérifier qu'il ne s'annule pas. C'est pourquoi, nous supposerons dans la suite que toutes les mailles considérées ne sont pas dégénérées et que la numérotation de leurs sommets garantit la positivité du jacobien des transformations associées.*

La transformation τ_L permet de passer explicitement de l'élément de référence $\hat{H}$ à la maille physique L. L'inverse de τ_L n'est en général pas calculable explicitement. Cependant, nous n'aurons à calculer l'antécédent d'un point physique par cette transformation que dans certains cas particuliers. Pour cela, nous utiliserons la méthode de Newton.

Nous définissons alors sur L une base d'approximation composée des fonctions $(\psi_{L,i})_{i\in[\![0\,;(d+1)^3-1]\!]}$. Ces fonctions sont transportées sur la maille physique à partir des fonctions de base définies sur l'élément de référence :

$$\forall \hat{x} \in \hat{H},\ \psi_{L,i} \circ \tau_L(\hat{x}) = \hat{\psi}_i(\hat{x}), \tag{3.23}$$

soit encore,

$$\forall x \in L,\ \psi_{L,i}(x) = \hat{\psi}_i \circ \tau_L^{-1}(x). \tag{3.24}$$

En général, les fonctions de base ainsi définies ne sont pas polynomiales dans l'espace physique. Cependant, les approximations des intégrales restent valables pour des maillages non dégénérés. Par ailleurs, nous prolongeons par 0 ces fonctions en dehors de la maille L :

$$\forall x \notin L,\ \psi_{L,i}(x) = 0. \tag{3.25}$$

Rappelons que puisque nous utilisons une interpolation nodale, à chaque instant t, l'approximation $\boldsymbol{w}$ est identifiée par ses valeurs sur les points d'intégration des mailles L. Notons $\boldsymbol{w}_i$ les composantes vectorielles de $\boldsymbol{w}$ dans

la base d'approximation relative à L. $\boldsymbol{w}$ s'exprime en un point d'intégration $g_{L,i} = \tau_L(\hat{g}_i)$ de L par :

$$\begin{aligned} \boldsymbol{w}(g_{L,i}, t) &= \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}_j(t)\psi_{L,j}(g_{L,i}) \\ &= \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}_j(t)\hat{\psi}_j(\hat{g}_i) \\ &= \boldsymbol{w}_i(t). \end{aligned} \tag{3.26}$$

Ainsi, la solution est entièrement définie par ses valeurs aux points d'intégration volumiques. Nous stockerons donc en mémoire ces valeurs.

Remarque 9. *Les formules de quadrature volumique* (3.3) *et surfacique* (3.9) *restent exactes sur la maille physique si la transformation géométrique τ_L est affine. Dans le cas contraire, l'intégration numérique devient approximative.*

Remarque 10. *Bien que les formules de quadrature soient approximatives lorsque la transformation géométrique est quelconque, l'intégration par parties* (1.39) *utilisée pour obtenir la formulation GD reste exacte : l'erreur commise est de l'ordre de 10^{-14} à l'ordre d'interpolation $d = 2$. Les erreurs commises et la maille tordue testée sont présentées dans la figure 3.3.*

Nous allons maintenant décrire la formulation discrète obtenue en utilisant l'espace d'approximation que nous venons de construire. Plus précisément, nous examinerons les approximations des intégrales présentes dans la formulation GD (1.48) obtenues en appliquant les formules de quadrature volumique (3.3) et surfacique (3.9).

3.2 Approximation des intégrales

En partant d'une formulation faible de notre problème de départ, nous avons établi une formulation semi-discrète. Nous allons maintenant détailler le calcul des différentes intégrales apparaissant dans la formulation semi-discrète (1.59) lorsque nous discrétisons la solution dans l'espace d'approximation décrit dans la section précédente.

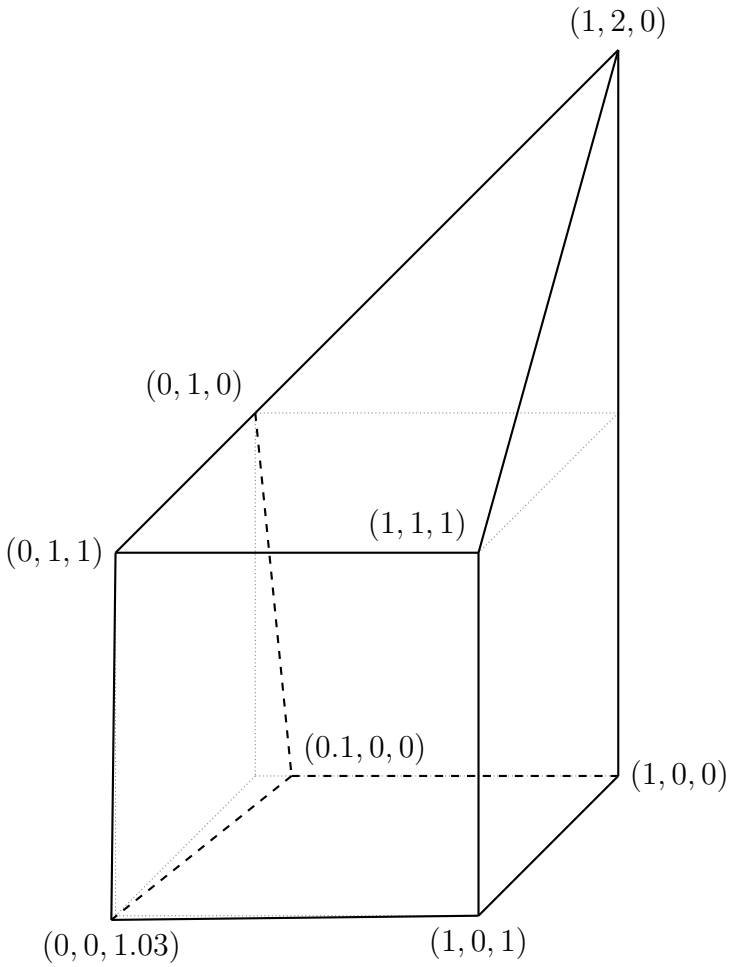
Pour cela, nous estimerons ces intégrales au moyen des formules de quadrature volumique (3.3) et surfacique (3.9). Nous verrons alors diverses simplifications de ces estimations introduites par Cohen *et al.* [15].

FIGURE 3.3 – Erreur commise au cours de l'intégration par parties discrète sur un hexaèdre quelconque (non affine).

(a) Erreurs.

d	Erreur L^2
0	$4.23 \cdot 10^{-16}$
1	$4.75 \cdot 10^{-15}$
2	$1.52 \cdot 10^{-14}$
3	$6.63 \cdot 10^{-14}$
4	$1.04 \cdot 10^{-13}$

(b) Hexaèdre.



Notons les décompositions des champs $\boldsymbol{w}$ et $\partial_t \boldsymbol{w}$ dans la base d'approximation :

$$\boldsymbol{w}(x,t) = \sum_{L\in\mathcal{M}} \sum_{i=0}^{(d+1)^3-1} \boldsymbol{w}_{L,i}(t)\psi_{L,i}(x), \text{ avec } \boldsymbol{w}_{L,i} \in \mathbb{R}^6 \text{ et } x \in L, \quad (3.27a)$$

$$\partial_t \boldsymbol{w}(x,t) = \sum_{L\in\mathcal{M}} \sum_{i=0}^{(d+1)^3-1} \boldsymbol{w}'_{L,i}(t)\psi_{L,i}(x), \text{ avec } \boldsymbol{w}'_{L,i} \in \mathbb{R}^6 \text{ et } x \in L. \quad (3.27b)$$

Dans ce qui suit, nous nous intéresserons au calcul des termes de la formulation semi-discrète (1.59) pour une fonction test $\psi_{L,i}$ fixée.

3.2.1 Terme de masse

Le terme de « masse » est donné par l'intégrale :

$$\begin{aligned}\int_L \partial_t \boldsymbol{w}\psi_{L,i}dx &= \int_L \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}'_{L,j}\psi_{L,j}\psi_{L,i}dx \\ &= \int_{\hat{H}} \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}'_{L,j}\hat{\psi}_j\hat{\psi}_i \mathrm{jac}\,(\tau_L)\,d\hat{x} \\ &\approx \omega_i \boldsymbol{w}'_{L,i}\mathrm{jac}\,(\tau_L)\,(\hat{g}_i).\end{aligned} \quad (3.28)$$

Ainsi, la matrice de masse obtenue est diagonale et ne dépend que des poids d'intégration et du jacobien de la transformation géométrique τ_L sur les points d'intégration.

3.2.2 Terme de rigidité

Le terme de « rigidité » est donné par l'intégrale :

$$\begin{aligned}&\int_L F\,(\boldsymbol{w}, \boldsymbol{w}, \nabla\psi_{L,i})\,dx \\ &\quad = \int_{\hat{H}} F\,(\boldsymbol{w}\circ\tau_L, \boldsymbol{w}\circ\tau_L, \nabla\psi_{L,i}\circ\tau_L)\,\mathrm{jac}\,(\tau_L)\,d\hat{x}.\end{aligned} \quad (3.29)$$

Nous pouvons écrire le gradient $\nabla\psi_{L,i}\circ\tau_L$ sous la forme $(\tau_L'^{\mathrm{T}})^{-1}\hat{\nabla}\hat{\psi}_i$ où τ'_L est la matrice jacobienne de τ_L. Nous introduisons alors la comatrice $\mathrm{com}\,(\tau'_L)$ de τ'_L définie par :

$$\mathrm{com}\,(\tau'_L)^{\mathrm{T}}\,\tau'_L = \det\,(\tau'_L)\,I. \quad (3.30)$$

Ainsi :

$$\begin{aligned}
&\int_L F\left(\boldsymbol{w}, \boldsymbol{w}, \nabla\psi_{L,i}\right) dx \\
&\quad = \int_{\hat{H}} F\left(\boldsymbol{w}\circ\tau_L, \boldsymbol{w}\circ\tau_L, \operatorname{com}\left(\tau_L'\right)\hat{\nabla}\hat{\psi}_i\right) d\hat{x} \\
&\quad \approx \sum_{j=0}^{(d+1)^3-1} \omega_j F\left(\boldsymbol{w}_{L,j}, \boldsymbol{w}_{L,j}, \operatorname{com}\left(\tau_L'\right)\hat{\nabla}\hat{\psi}_i(\hat{g}_j)\right).
\end{aligned} \tag{3.31}$$

Plusieurs termes de la somme précédente sont nuls. Nous pouvons les déterminer en appliquant les propriétés de fonctions de base. Cela nous permettra de diminuer significativement le volume de calcul. Par définition des fonctions de base, le gradient de la fonction de base $\hat{\nabla}\hat{\psi}_i$, avec $i = p_0 + (d+1)(p_1 + (d+1)p_2)$, s'écrit :

$$\hat{\nabla}\hat{\psi}_i(\hat{x}) = \begin{pmatrix} l'_{p_0}(\hat{x}_0) l_{p_1}(\hat{x}_1) l_{p_2}(\hat{x}_2) \\ l_{p_0}(\hat{x}_0) l'_{p_1}(\hat{x}_1) l_{p_2}(\hat{x}_2) \\ l_{p_0}(\hat{x}_0) l_{p_1}(\hat{x}_1) l'_{p_2}(\hat{x}_2) \end{pmatrix}. \tag{3.32}$$

Par définition des polynômes de Lagrange, la valeur de la première composante de ce gradient sur un point d'intégration $\hat{g}_j = (\xi_{q_0}, \xi_{q_1}, \xi_{q_2})$ s'écrit :

$$\begin{aligned}
\partial_0\hat{\psi}_i(\hat{g}_j) &= l'_{p_0}(\xi_{q_0}) l_{p_1}(\xi_{q_1}) l_{p_2}(\xi_{q_2}) \\
&= l'_{p_0}(\xi_{q_0}) \delta_{p_1,q_1}\delta_{p_2,q_2}.
\end{aligned} \tag{3.33}$$

Cette composante est donc nulle en tout point d'intégration, excepté sur les points dont les indices q_1 et q_2 correspondent aux indices p_1 et p_2 du point $\hat{g}_i$. Ces points sont les points alignés avec $\hat{g}_i$ dans la direction $(1,0,0)^{\mathrm{T}}$, c'est-à-dire les $d+1$ points de coordonnées :

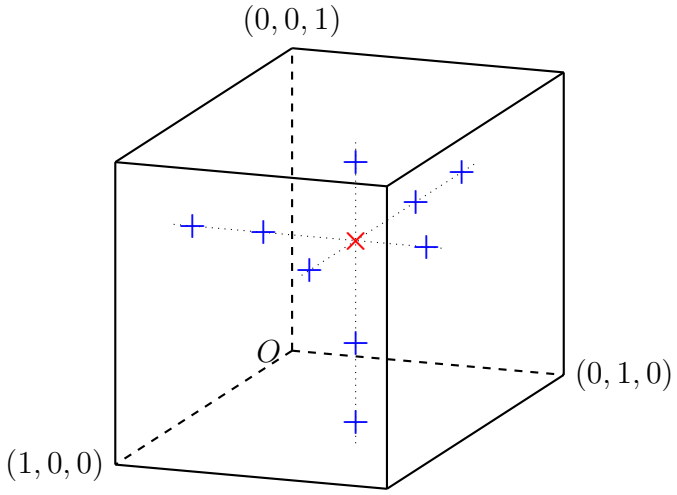
$$\hat{g}_j = \left(\xi_{q_0}, \xi_{p_1}, \xi_{p_2}\right), \text{ avec } q_0 \in [\![0\,;\, d]\!]. \tag{3.34}$$

De la même manière, la seconde, respectivement la troisième, composante s'annule en tout point d'intégration, excepté les points alignés avec la direction $(0,1,0)^{\mathrm{T}}$, respectivement la direction $(0,0,1)^{\mathrm{T}}$.

Afin de calculer le terme de rigidité (3.31), nous n'avons à prendre en compte que les valeurs sur ces $3(d+1)$ points. De plus, en chacun de ces points, excepté $\hat{g}_i$, seule la composante du gradient correspondant à la direction d'alignement est non nulle. Cette propriété permet encore de simplifier la programmation des calculs de ce terme (figure 3.4).

La matrice de rigidité est alors diagonale par bloc, chaque bloc correspondant à une maille du maillage. Le calcul de chacun de ses termes ne nécessite que l'évaluation de $3(d+1)$ termes et est local à chaque maille, *i.e.* il n'y a pas

FIGURE 3.4 – Points d'intégration volumiques (« + » bleus) associés aux fonctions de base intervenant dans le calcul du gradient au point d'intégration considéré (« × » rouge) pour $d = 3$. Seules les fonctions associées aux points alignés interviennent.



d'interaction entre fonctions de base appartenant à des mailles différentes. Cette propriété rend la parallélisation de ce terme aisée.

3.2.3 Terme de flux

Le terme de « flux » permet le couplage entre les mailles voisines. Notons $\boldsymbol{w}_L$ le champ $\boldsymbol{w}$ provenant de la maille L. Le champ $\boldsymbol{w}_R$ désignera le champ $\boldsymbol{w}$ provenant d'une maille R ayant une face en commun (ou simplement une surface de contact) avec la maille L. Notons également $(Q_f)_{f\in[\![0\,;5]\!]}$ les 6 faces de L image par τ_L des faces $(\hat{Q}_f)_{f\in[\![0\,;5]\!]}$ de $\hat{H}$.

Sur une face Q_f ne touchant pas le bord du domaine de calcul, nous avons :

$$\begin{aligned}&\int_{Q_f} F\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) \psi_{L,i} ds \\ &\quad = \int_{\hat{Q}_f} F\left(\boldsymbol{w}_L \circ \tau_L, \boldsymbol{w}_R \circ \tau_L, \operatorname{com}\left(\tau_L'\right) \hat{\boldsymbol{n}}\right) \psi_{L,i} \circ \tau_L d\hat{s}\end{aligned} \tag{3.35}$$

où $\operatorname{com}\left(\tau_L'\right)$ est la comatrice de τ_L' définie précédemment (3.30).

Notons $\boldsymbol{w}_{L,f,i}$ la valeur des champs provenant de la maille L au point d'intégration $g_{L,f,i}$, le i-ième point d'intégration de la face Q_f de la maille L.

Nous avons :

$$\boldsymbol{w}_{L,f,i} = \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}_{L,j} \hat{\psi}_j(\hat{g}_{f,i}). \tag{3.36}$$

Soit f' l'indice de la face de la maille R coïncidant (ou en contact) avec la face f de L. Lorsque les faces et l'ordre d'interpolation des deux mailles coïncident, le point $g_{L,f,i}$ coïncide avec un point d'intégration de la maille R. Le raccord est alors dit conforme et l'extrapolation des champs provenant de la maille voisine est semblable à celle effectuée dans la maille L.

Raccord conforme

Soit σ_{LR}, la permutation qui à l'indice i du point d'intégration sur la face f de L associe l'indice du point d'intégration sur la face f' de R correspondant au point $g_{L,f,i}$, c'est-à-dire :

$$\tau_R(\hat{g}_{f',\sigma_{LR}(i)}) = g_{R,f',\sigma_{LR}(i)} = g_{L,f,i} = \tau_L(\hat{g}_{f,i}). \tag{3.37}$$

Notons alors $\boldsymbol{w}_{R,f',i}$ la valeur des champs provenant de la maille R au point d'intégration $g_{R,f',i}$, le i-ième point d'intégration de la face $Q_{f'}$ de la maille R. Nous avons :

$$\boldsymbol{w}_{R,f',i} = \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}_{R,j} \hat{\psi}_j(\hat{g}_{f',i}). \tag{3.38}$$

Nous approchons alors l'intégrale précédente par :

$$\begin{aligned} &\int_{Q_f} F\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) \psi_{L,i} ds \\ &\quad \approx \sum_{j=0}^{(d+1)^2-1} \omega_{f,j} \hat{\psi}_i(\hat{g}_{f,j}) F\left(\boldsymbol{w}_{L,f,j}, \boldsymbol{w}_{R,f',\sigma_{LR}(j)}, \mathrm{com}\left(\tau_L'\right) \hat{\boldsymbol{n}}\right). \end{aligned} \tag{3.39}$$

De plus, la fonction de base $\hat{\psi}_i$ est non nulle en uniquement un point d'intégration de la face f (proposition 9). Il s'agit du point d'intégration d'indice $j = \pi(f,i)$, projeté sur la face f du point d'intégration $\hat{g}_i$ associé à la fonction de base $\hat{\psi}_i$. La somme précédente se réduit donc à un seul terme :

$$\begin{aligned} &\int_{Q_f} F\left(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n}\right) \psi_{L,i} ds \\ &\quad \approx \omega_{f,\pi(f,i)} \hat{\psi}_i(\hat{g}_{f,\pi(f,i)}) F\left(\boldsymbol{w}_{L,f,\pi(f,i)}, \boldsymbol{w}_{R,f',\sigma_{LR}(\pi(f,i))}, \mathrm{com}\left(\tau_L'\right) \hat{\boldsymbol{n}}\right). \end{aligned} \tag{3.40}$$

Afin de calculer ce flux, nous devons calculer les champs provenant des deux mailles voisines au point $g_{L,f,\pi(f,i)}$. L'extrapolation du champ $\boldsymbol{w}_{L,f,\pi(f,i)}$ n'utilise que les $d+1$ valeurs associées aux fonctions de base des points d'intégration volumiques se projetant sur le point de face $\hat{g}_{f,\pi(f,i)}$, *i.e.* les fonctions de base non nulles en ce point.

Lorsque les ordres d'approximation dans les deux mailles voisines sont égaux et que les faces coïncident, le calcul de $\boldsymbol{w}_{R,f',\sigma_{LR}(\pi(f,i))}$ est semblable à celui de $\boldsymbol{w}_{L,f,\pi(f,i)}$.

Raccord non conforme

Dans le cas contraire et de manière plus générale, lorsque le point $g_{L,f,\pi(f,i)}$ ne coïncide pas avec un point d'intégration de la maille R, l'extrapolation des champs provenant de la maille voisine nécessite l'utilisation de la réciproque de la transformation géométrique τ_R (supposée bijective). Ce cas se présente lorsque les faces ou l'ordre d'interpolation des deux mailles ne coïncident pas. Nous devons alors calculer le champ $\boldsymbol{w}_R$ au point $\tau_L(\hat{g}_{f,\pi(f,i)})$ en appliquant la décomposition dans la base de l'élément de référence :

$$\boldsymbol{w}_R(g_{L,f,\pi(f,i)}) = \sum_{j=0}^{(d+1)^3-1} \boldsymbol{w}_{R,j}\hat{\psi}_j \circ \tau_R^{-1} \circ \tau_L(\hat{g}_{f,\pi(f,i)}). \tag{3.41}$$

Dans cette dernière formule, l'ordre d de la maille R peut être différent de celui de la maille L. De plus, aucune propriété n'affirme que certains termes de cette somme sont nuls. Il faut donc calculer les $(d+1)^3$ termes pour extrapoler les champs en un point de face, contrairement au cas conforme qui ne nécessite que le calcul de $d+1$ termes.

Maille de bord

Enfin, dans le cas d'une face touchant le bord du domaine, le flux numérique est remplacé par un flux de bord traduisant l'application d'une condition aux limites spécifique. L'évaluation de ce terme de flux en un point d'intégration de cette face nécessite comme précédemment l'extrapolation du champ de la maille L en ce point. Afin de garantir la stabilité du schéma, le flux de bord doit satisfaire le théorème 3 garantissant la décroissance de l'énergie du système.

Remarque 11. *Le calcul du terme de flux fait intervenir des champs provenant des mailles voisines. Ainsi, les fonctions de base appartenant à ces mailles interagissent, ce qui a pour effet de rendre la parallélisation du calcul de ce terme plus délicate et plus coûteuse en temps.*

3.3 Schémas temporels

Après avoir donné une formulation discrétisée en espace, il est indispensable de fournir également une technique de discrétisation en temps. Le choix du schéma en temps est crucial. Il faut en effet s'assurer de la précision et de la stabilité du schéma couplé espace-temps.

Nous avons décrit le calcul de la dérivée en temps du champ $\partial_t \boldsymbol{w}$ en fonction du champ $\boldsymbol{w}$ à un instant t donné. Dans cette section, nous noterons $\mathcal{G}$, l'application qui à un champ $\boldsymbol{w} \in \mathcal{H}_h^m$ et un instant t associe sa dérivée en temps obtenue par la méthode GD décrite précédemment :

$$\partial_t \boldsymbol{w} = \mathcal{G}(\boldsymbol{w}, t). \tag{3.42}$$

Soit Δt un réel strictement positif de $[0\,;T]$. Ce réel est appelé « pas de temps ». Notons $\boldsymbol{w}^n$ la solution numérique du problème semi-discret (1.59) au temps $t_n = n\Delta t$. Supposons $\boldsymbol{w}^n$ connu pour un entier n donné. Nous allons décrire comment calculer la solution au pas de temps suivant $\boldsymbol{w}^{n+1}$. La suite $(\boldsymbol{w}^n)_{n\in\mathbb{N}}$ ainsi définie est appelée solution discrète du problème étudié.

3.3.1 Runge-Kutta

En pratique, nous calculons le plus souvent $\boldsymbol{w}^{n+1}$ par la méthode de Runge-Kutta d'ordre 2 (RK2) :

$$\begin{aligned} \boldsymbol{w}^{n+\frac{1}{2}} &= \boldsymbol{w}^n + \frac{\Delta t}{2}\mathcal{G}(\boldsymbol{w}^n, t), \\ \boldsymbol{w}^{n+1} &= \boldsymbol{w}^n + \Delta t \mathcal{G}\left(\boldsymbol{w}^{n+\frac{1}{2}}, t + \frac{\Delta t}{2}\right). \end{aligned} \tag{3.43}$$

La première étape est appelée prédiction. La seconde étape est appelée mise à jour.

Cependant, comme nous le préciserons dans la suite, cette méthode peut s'avérer numériquement instable dans certains cas. Nous avons donc aussi implémenté la méthode de Runge-Kutta d'ordre 3 (RK3) afin de bénéficier, si nécessaire, de sa condition de stabilité moins contraignante.

3.3.2 Condition de stabilité numérique

Pour étudier la stabilité du schéma GD couplé à une intégration en temps de type RK, nous considérons le problème d'évolution (1.14) avec des conditions aux limites impliquant l'existence et l'unicité de la solution.

Soit $\boldsymbol{w}$ la solution approchée de ce système. Notons alors $\boldsymbol{A}$ la matrice liant la dérivée en temps $\partial_t \boldsymbol{w}$ au champ $\boldsymbol{w}$. Le schéma numérique s'écrit, en simplifiant :

$$\partial_t \boldsymbol{w} = \boldsymbol{A}\boldsymbol{w}. \tag{3.44}$$

Dans ce cas, la solution $\boldsymbol{w}$ est donnée par :

$$\boldsymbol{w}(t) = \exp(\boldsymbol{A}t)\boldsymbol{w}_0. \tag{3.45}$$

Un développement de Taylor donne alors l'approximation à l'ordre k :

$$\boldsymbol{w}^{n+1} = \tilde{\boldsymbol{A}}\boldsymbol{w}^n, \text{ avec } \tilde{\boldsymbol{A}} = \sum_{i=0}^{k} \frac{(\Delta t)^i}{i!}\boldsymbol{A}^i. \tag{3.46}$$

Dans le cas particulier linéaire, tous les schémas d'intégration en temps d'ordre k fixé sont équivalents. Un tel schéma est stable si les valeurs propres de la matrice $\tilde{\boldsymbol{A}}$ sont de module inférieur à 1. Ces valeurs propres sont de la forme $\sum_{i=0}^{k} \frac{(\Delta t\lambda)^i}{i!}$, où λ est une valeur propre de $\boldsymbol{A}$.

Proposition 10. *Un schéma d'ordre k est stable pour un pas de temps Δt si toutes les valeurs propres λ de $\boldsymbol{A}$ sont telles que $\Delta t\lambda$ appartienne à :*

$$S_k = \left\{ \mu \in \mathbb{C} : \left| \sum_{i=0}^{k} \frac{\mu^i}{i!} \right| \leq 1 \right\}. \tag{3.47}$$

On peut alors noter C_k le contour de l'ensemble S_k, c'est-à-dire :

$$C_k = \left\{ \mu \in \mathbb{C} : \left| \sum_{i=0}^{k} \frac{\mu^i}{i!} \right| = 1 \right\}. \tag{3.48}$$

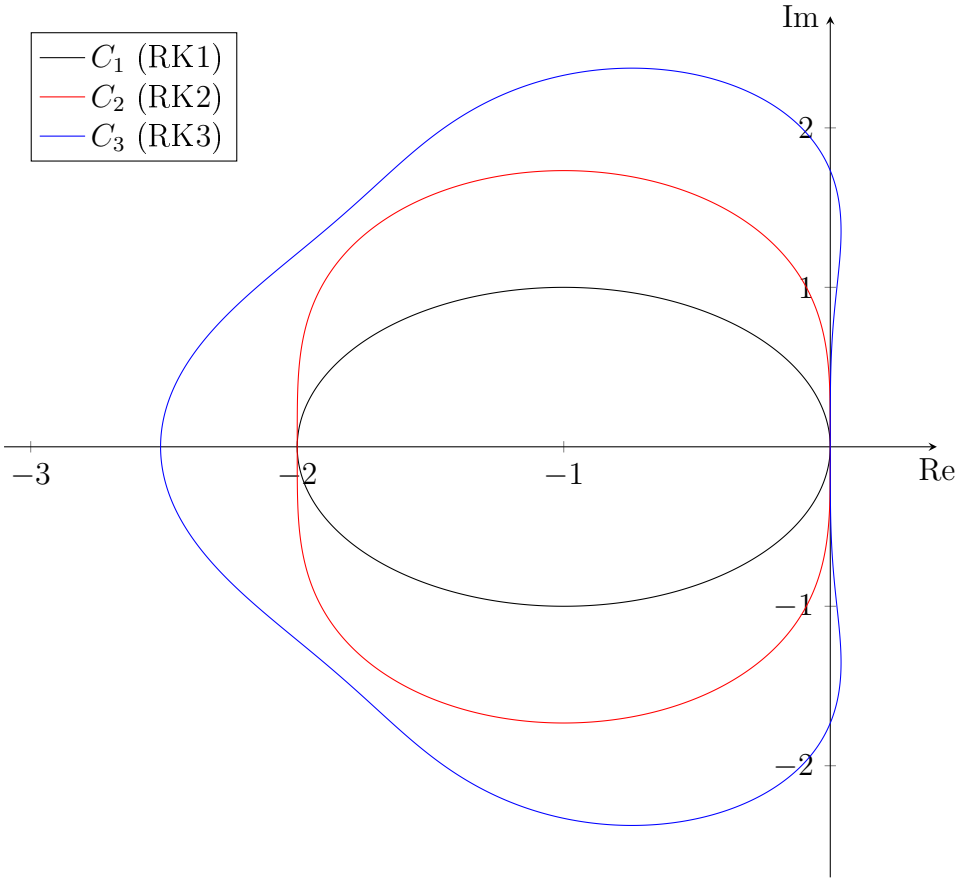
Dans le cas du couplage GD-RK2, la zone de stabilité est donc donnée par l'ensemble :

$$S_2 = \left\{ \mu \in \mathbb{C} : \left| 1 + \mu + \frac{\mu^2}{2} \right| \leq 1 \right\}. \tag{3.49}$$

Cet ensemble est illustré par son contour dans la figure 3.5.

Le calcul des valeurs propres de $\boldsymbol{A}$ permet donc d'assurer la stabilité du schéma couplé en cas d'existence d'un pas de temps Δt vérifiant la proposition 10. Nous constatons que dans le cas des schémas RK1 et RK2, une condition suffisante à l'existence d'un tel pas de temps est que les valeurs propres non nulles de la matrice $\boldsymbol{A}$ aient une partie réelle négative. Le choix d'un pas de

FIGURE 3.5 – Contours des zones de stabilité du schéma couplé pour les schémas temporels d'ordre 1, 2 et 3. La zone de stabilité d'un schéma d'ordre 1 est l'ensemble des points compris dans le cercle unité centré en le point d'affixe $z = -1$.



temps plus petit permet alors de rapprocher les valeurs propres de l'axe réel jusqu'à les inclure dans la zone de stabilité. Dans le cas du schéma RK3, la zone de stabilité est tangente à l'axe imaginaire dans le demi-plan des parties réelles positives, englobant une partie de l'axe imaginaire, ce qui permet de fortement relâcher la contrainte sur le choix du pas de temps.

Cependant, le calcul de la matrice $\boldsymbol{A}$ nécessite d'appliquer le schéma couplé pour chaque point d'approximation du maillage, soit pour N mailles discrétisées à l'ordre d, ce qui représente $6N(d+1)^3$ degrés de liberté dans le cas des équations de Maxwell. De plus, pour des maillages contenant un grand nombre de mailles, le calcul des valeurs propres de cette matrice est coûteux en temps.

Cette méthode de calcul des valeurs propres a cependant été utilisée ponctuellement afin d'évaluer la stabilité d'une maille donnée, par exemple au cours de l'implémentation d'autres méthodes ayant recours à des pas de temps propres à chaque maille (méthodes dites à « pas de temps local »). Nous évoquerons ce type de schémas dans le chapitre 5.

3.3.3 Condition de stabilité *a priori*

Devant la complexité de l'évaluation de la condition de stabilité énoncée précédemment, nous allons présenter un critère de type CFL [16], permettant de déterminer un pas de temps de stabilité *a priori*, en fonction du maillage uniquement.

La définition d'un tel pas de temps par un critère de type CFL a fait l'objet de différents travaux, voir par exemple [22,38,47] pour une estimation de la stabilité du schéma GD associé à une intégration en temps de type saute-mouton. Pour les schémas de type Runge-Kutta, il n'existe pas encore, à notre connaissance, de résultat théorique de stabilité sur des maillages non structurés.

Nous utilisons la condition de stabilité empirique décrite par Hesthaven *et al.* [38] sous la forme :

$$\Delta t \leq C_1 \frac{\min l_H}{v} \frac{1}{2d+1}, \tag{3.50}$$

où $\min l_H$ représente la plus petite arête du maillage hexaédrique considéré, v représente la vitesse de propagation dans le milieu considéré et C_1 est donné par le rapport du volume d'un hexaèdre divisé par la surface de ses faces, soit $C_1 = 1/6$. Les valeurs propres de la matrice $\tilde{\boldsymbol{A}}$ (3.46) obtenues sur un hexaèdre quelconque sont présentées dans la figure 3.6.

Remarque 12. *Dans l'implémentation du solveur* `teta-clac`*, cette condition CFL* (3.50) *a été implémentée avec le dénominateur* $v(d+1)$*. Les simulations d'ordre élevé exécutées avec cette condition plus faible se sont jusqu'alors avérées stables.*

Cette condition de stabilité convient parfaitement aux hexaèdres « relativement » droits. Bien que nous n'ayons pas de critère précis pour quantifier ce degré de relativité. Nous avons néanmoins constaté que pour certains maillages contenant des mailles très déformées, issues du découpage de tétraèdres notamment (figure 4.3), il peut être nécessaire d'abaisser le coefficient de CFL du schéma couplé GD-RK2.

Ces instabilités liées au critère CFL sont principalement apparues au cours de résolutions utilisant un pas de temps local (chapitre 5). En effet, lorsque le pas de temps est uniforme, il est déterminé afin de satisfaire le critère CFL sur la plus petite maille, contrairement aux méthodes à pas de temps local qui tiennent compte de ce critère sur chaque maille du maillage.

Les travaux de Schneider *et al.* [62] et Burman *et al.* [9] donnent une formulation plus précise de la condition CFL, améliorant la stabilité du schéma couplé GD-RK2 :

$$\Delta t \leq C_2 \left(\frac{\min \Delta x}{v} \right)^{\alpha}, \tag{3.51}$$

où $\min \Delta x$ représente la plus petite distance entre deux points d'intégration, $\alpha = 4/3$ et avec C_2 une constante. Ce choix pour α est uniquement préconisé pour le schéma couplé GD-RK2. Dans le cas d'une intégration en temps d'ordre supérieur, nous pouvons choisir $\alpha = 1$.

Le calcul du pas de temps à l'aide du critère (3.51) nécessite de calculer $N(d+1)^3((d+1)^3-1)/2$ distances et est donc plus coûteux que le critère (3.50) qui ne nécessite que le calcul de $28N$ distances. Néanmoins, cette seconde formulation du critère CFL a permis une résolution stable des équations de Maxwell avec un schéma à pas de temps local pour un cas instable avec le premier critère. Ce qui démontre qu'elle est plus adaptée aux déformations des mailles.

3.3.4 Diagnostic de stabilité : Puissance itérée

Une autre technique de diagnostic des instabilités a été implémentée dans le solveur : la méthode de puissance itérée.

Proposition 11. *Soit* $\boldsymbol{A}$ *une matrice carrée de taille* m *et* $(\lambda_i)_{i \in [\![1\,;m]\!]}$ *ses valeurs propres telles que :*

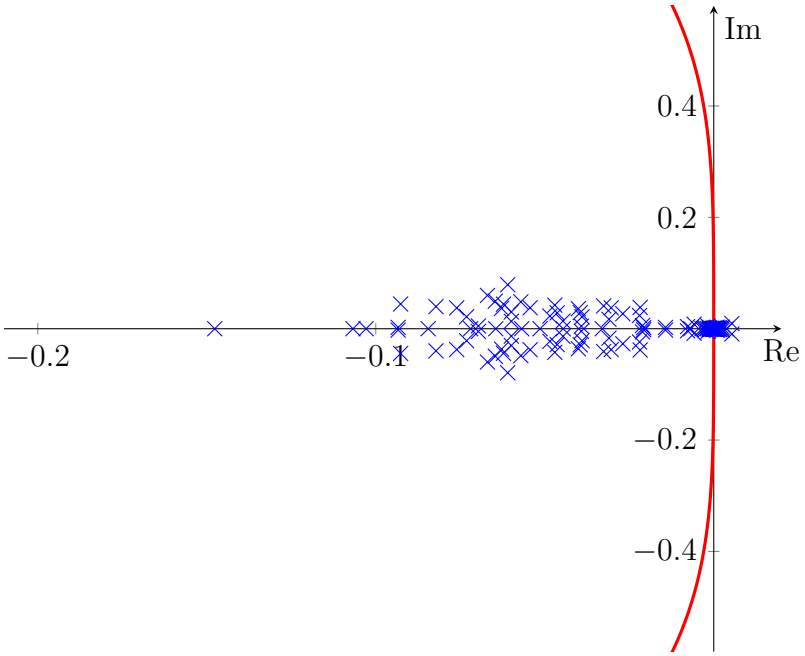
$$|\lambda_1| > \max(|\lambda_2|, \ldots, |\lambda_m|). \tag{3.52}$$

Soit aussi les sous-espaces caractéristiques $(E_i)_{i \in [\![1\,;m]\!]}$ *de* $\boldsymbol{A}$ *associés aux valeurs propres* (λ_i).
Alors, si la projection de $\boldsymbol{w}_0$ *sur* E_1 *n'est pas nulle, la suite de vecteurs* $(\boldsymbol{w}^n)$ *définie par la relation de récurrence :*

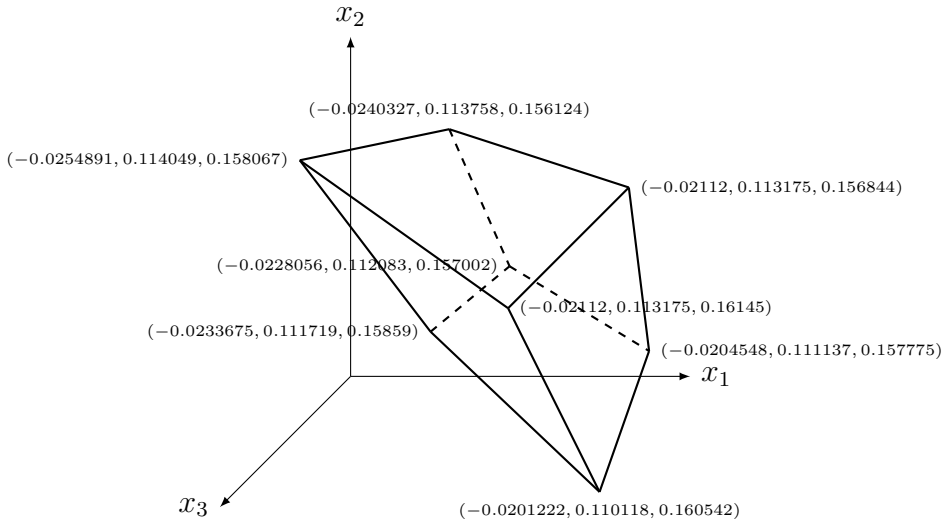
$$\forall n \in \mathbb{N}, \boldsymbol{w}^{n+1} = \frac{1}{\|\boldsymbol{A}\boldsymbol{w}^n\|} \boldsymbol{A}\boldsymbol{w}^n \tag{3.53}$$

FIGURE 3.6 – Valeurs propres de la matrice $\tilde{\boldsymbol{A}}$ (3.46) dans la zone de stabilité GD-RK2 dans le cas d'un hexaèdre du cas d'application 7.2. La condition de stabilité utilisée est la formule (3.50).

(a) Valeurs propres.



(b) Hexaèdre.

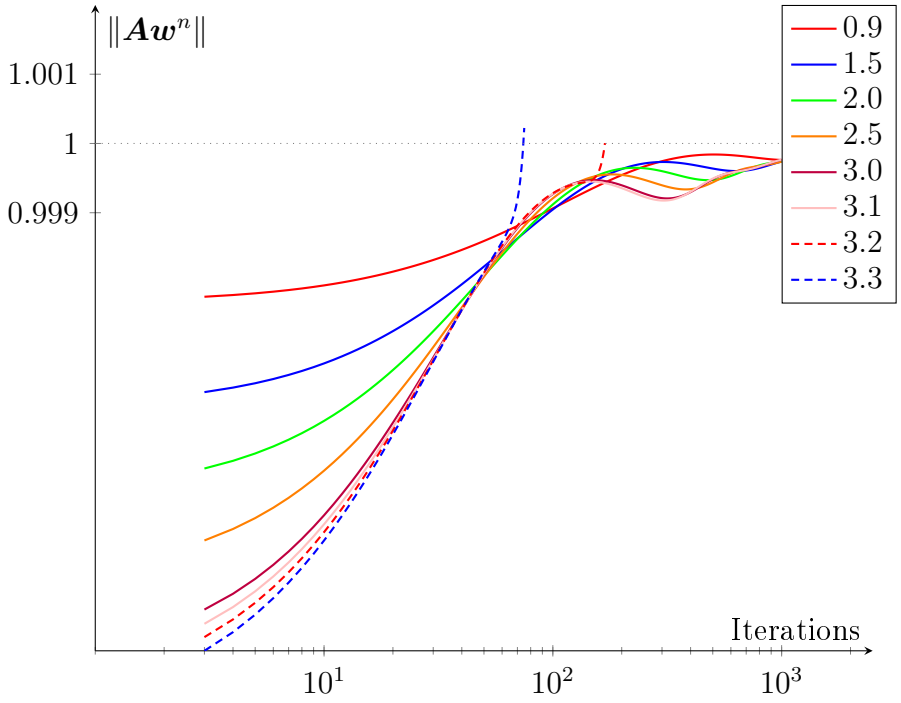


vérifie :

$$\lim_{n\to\infty} \|\boldsymbol{A}\boldsymbol{w}^n\| = |\lambda_1|\,. \tag{3.54}$$

Pour appliquer cette méthode, les champs sont tout d'abord initialisés avec un bruit blanc gaussien. Cette initialisation nous permet d'assurer (avec une très forte probabilité) que la projection du vecteur initial sur l'espace caractéristique de la plus grande valeur propre n'est pas nulle.

FIGURE 3.7 – Résultats d'application du diagnostic de la puissance itérée pour différents coefficients CFL dans le cas de la simulation de la section 7.2 à l'ordre d'interpolation spatiale $d = 2$.



Ensuite, à chaque étape, une normalisation globale des champs est effectuée avant d'appliquer le schéma couplé représenté dans la proposition précédente par la matrice $\boldsymbol{A}$. La norme des champs ainsi calculée à chaque étape tend vers le module de la plus grande valeur propre, ce qui nous donne deux informations : l'évolution de la norme nous indique l'état de la convergence et la limite nous permet de nous positionner par rapport aux zones de stabilité définies dans la proposition 10.

Grâce à cette méthode nous pouvons évaluer la stabilité du schéma pour un pas de temps et un maillage donnés. Nous avons aussi la possibilité d'affi-

ner le calcul du pas de temps en l'augmentant jusqu'à ce que la limite tende vers 1.

L'inconvénient majeur de cette méthode est que la convergence est généralement d'ordre $O(\frac{1}{n})$. Il faut donc plusieurs milliers d'itérations afin de vérifier la stabilité du schéma.

De plus, toute source électromagnétique doit être supprimée pour l'application de ce diagnostic. Ainsi, aucun signal ne doit être injecté, la conductivité des matériaux doit être nulle et le modèle PML (section 2.1.5) doit être remplacé par la condition de Silver-Müller (section 2.1.4).

Nous appliquons ce diagnostic sur le cas d'application étudié dans la section 7.2. La figure 3.7 présente l'évolution de la norme $\|\boldsymbol{A}\boldsymbol{w}^n\|$ en fonction du nombre d'itérations pour différents coefficients CFL. Le diagnostic a été mené jusqu'à 10k itérations dans les cas stables, bien que nous ne présentions que les 1000 premières. Nous constatons que cette simulation est stable pour un coefficient CFL de 3.1 et instable au-delà, pour une interpolation spatiale d'ordre $d = 2$. Pour l'ordre spatial $d = 1$, le coefficient CFL de stabilité maximal ainsi déterminé est de 4.5.

Remarque 13. *En temps normal, un coefficient CFL ne doit pas dépasser* 1. *Cette assertion est vraie lorsque la condition CFL est exacte. Or, la condition* (3.50) *est empirique et fonctionne dans la plupart des cas, mais sans être exacte. Il arrive donc que certaines simulations soient instables avec un coefficient CFL de* 0.9. *Il arrive aussi, comme dans le cas ci-dessus, qu'une simulation soit stable avec un coefficient CFL supérieur à* 1.

Conclusion

Dans ce chapitre, nous avons introduit l'espace d'approximation dans lequel nous chercherons la solution numérique de notre problème. Cet espace est représenté par un élément de référence : le cube unité, et muni de fonctions de base qui vont nous permettre d'exprimer la solution à l'aide de coordonnées discrètes. Ces fonctions de base sont les produits tensoriels des polynômes de Lagrange qui s'appuient sur les points de Gauss-Legendre dans chaque direction. Une transformation géométrique nous permet alors de passer de l'élément de référence à n'importe quelle maille physique. Nous sommes donc en mesure de résoudre tout type de problème d'électromagnétisme modélisé à partir de mailles hexaédriques.

Cet espace d'approximation a été choisi pour ses propriétés de simplification des formules de quadrature par l'annulation du gradient des fonctions

de base en de nombreux points d'interpolation. Ces simplifications n'apparaissent pas dans les mailles tétraédriques.

Nous avons ensuite présenté les schémas de type Runge-Kutta qui vont nous permettre d'intégrer temporellement la solution afin d'avancer cette dernière jusqu'au temps final souhaité. L'implémentation de ces schémas ne présente pas de difficulté. Cependant, leurs conditions de stabilité sont très contraignantes. Nous verrons un autre type de schéma, permettant de réduire l'impact de ces contraintes, dans le chapitre 5.

Dans le chapitre suivant, nous décrivons l'implémentation sur un ou plusieurs accélérateurs, comme des GPU ou des CPU multicœurs, de la méthode GD sur hexaèdres. Les cartes graphiques sont toujours plus utilisées dans la simulation numérique et permettent de paralléliser le traitement de grandes quantités de données. Dans le cas de notre solveur, les simplifications induites par les hexaèdres permettent la parallélisation efficace du code.

Chapitre 4

Parallélisations et optimisations

Dans ce chapitre, nous décrivons l'implémentation sur un ou plusieurs accélérateurs de la méthode Galerkin Discontinue (GD) décrite au chapitre 3. Pour cela, nous utilisons la bibliothèque OpenCL (pour « *Open Computing Language* ») [29] qui permet d'exploiter la puissance de calcul des accélérateurs (cartes graphiques, processeurs multi-cœurs, coprocesseurs, *etc.*) produits par divers constructeurs (NVidia, AMD, Intel) et compatibles avec différents systèmes d'exploitation (Linux, Windows, Mac OS).

La puissance des cartes graphiques récentes permet de réaliser des calculs de façon massivement parallèle. Ces GPGPU (pour « *General Purpose Graphic Processing Unit* »), ou plus simplement GPU, possèdent une puissance crête élevée, de l'ordre de plusieurs téraflops. Ce type de matériel est très utilisé dans la simulation numérique. Nous les retrouvons par exemple dans la résolution des équations de Maxwell [11, 45], de Vlasov-Maxwell [18], dans la simulation de gaz et fluides [19, 35], et bien d'autres domaines. L'implémentation GPU `teta-clac` a initialement été réalisée en C++ en utilisant la bibliothèque OpenCL au cours de la thèse de Strub [69], à partir du code non parallèle FemGD développé à l'ONERA de Toulouse [15].

Afin de pouvoir traiter des volumes de données importants, la méthode GD est également parallélisée sur plusieurs accélérateurs à l'aide d'une implémentation du standard MPI (« *Message Passing Interface* ») [14]. MPI offre la possibilité de lancer plusieurs processus qui effectuent les calculs sur différentes parties du maillage et communiquent par un système de messages.

Cette implémentation GPU est aussi exploitable sur un processeur multi-cœurs (CPU) plus classique. Ces derniers ont beaucoup évolué au cours des années passées pour être aujourd'hui composés de plusieurs dizaines de cœurs logiques cadencés à des fréquences supérieures à celles des GPU. Cependant, de par leur architecture distincte de celle des GPU, une implémentation spécifique aux CPU doit être développée afin de les exploiter au mieux.

Dans un premier temps, nous présenterons la parallélisation OpenCL des calculs sur GPU ainsi que la stratégie asynchrone de parallélisation MPI. Ensuite nous décrirons les adaptations effectuées afin d'en améliorer les performances sur CPU. Enfin, nous présenterons une méthode d'adaptation de l'ordre d'interpolation spatiale appliquée aux mailles du maillage dans le but de réduire le temps de simulation en optimisant l'échantillonnage du domaine de calcul.

4.1 La bibliothèque OpenCL

OpenCL est une interface de programmation (ou API pour « *Application Programming Interface* ») maintenue par le Khronos Group [29], qui gère aussi le standard graphique OpenGL. Cette API a initialement été créée par Apple, puis confiée à Khronos en 2008, dans l'espoir qu'elle devienne un standard. Les différents constructeurs d'accélérateurs (NVidia, AMD, Intel) ont développé et maintiennent des pilotes permettant d'utiliser OpenCL pour paralléliser des calculs sur leurs matériels.

Dans l'abstraction OpenCL, le processeur appelé hôte (ou « *host* ») exécute les instructions du programme principal et assure la soumission des calculs sur les périphériques de calculs ainsi que les tâches annexes (pré- et post-traitement des données, *etc.*).

4.1.1 Les périphériques OpenCL

Les périphériques de calculs (ou accélérateurs, ou « *devices* ») exécutent les calculs parallèles programmés sous forme de fonctions à l'aide d'un langage dérivé du C. Une telle fonction est appelée noyau (ou « *kernel* »). L'exécution d'un kernel par un périphérique ou le transfert de données entre le périphérique et l'hôte sont appelés des tâches. La soumission d'une tâche est effectuée via une file d'attente.

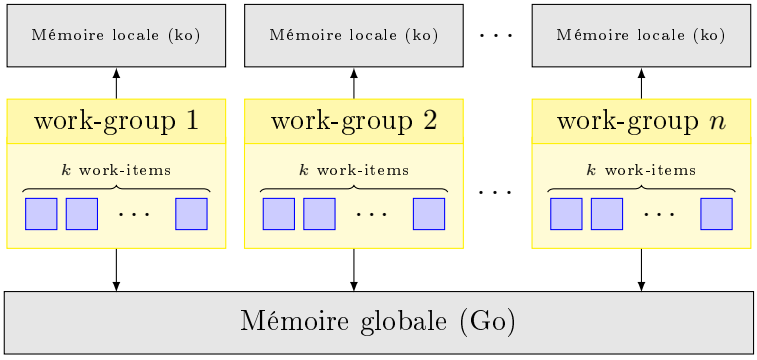
Du point de vue de l'exécution d'un kernel, le périphérique est vu sous la forme d'un ensemble d'unités de travail appelées « *work-items* », rassemblés en groupes de travail appelés « *work-groups* ». Un kernel a aussi accès à plusieurs espaces mémoire (figure 4.1). En particulier, tous les work-items d'un même work-group ont accès à un espace mémoire partagé réservé.

L'exécution des tâches peut être gérée de plusieurs manières : elles peuvent être soumises de manière synchrone ou asynchrone. Dans le cas synchrone,

les tâches sont exécutées dans l'ordre de soumission et le programme hôte est stoppé durant l'exécution d'une tâche. Ce mode de soumission est généralement utilisé en phase de développement pour se soustraire des éventuelles erreurs de synchronisation.

Dans le cas asynchrone, chaque tâche est associée à un événement qui permet de consulter son statut d'exécution : en attente, soumise, en cours d'exécution ou terminée. Contrairement au mode synchrone, le programme hôte continue son exécution sans interruption jusqu'à rencontrer un point de synchronisation demandé par l'utilisateur. Afin de garantir le bon enchaînement des calculs, l'exécution d'une tâche peut être conditionnée par la terminaison d'un ensemble de tâches à l'aide des événements qui leur sont associés. Ces dépendances entre les événements sont représentées au moyen d'un graphe acyclique direct appelé graphe de tâches.

FIGURE 4.1 – Schéma d'un périphérique OpenCL présentant des work-items regroupés en work-groups.



4.1.2 La mémoire OpenCL

Dans l'API OpenCL, la mémoire d'un périphérique est constituée de différentes parties. Par défaut, une variable définie dans un kernel est allouée dans l'espace privé du work-item (aussi identifiable par le préfixe facultatif `__private`). Cette mémoire est allouée pour chaque work-item et demeure la plus rapide.

Nous disposons aussi d'une mémoire constante (identifiée par le préfixe `__constant`). Cette mémoire est d'accès rapide et accessible par tous les work-items. Les données de cet espace sont définies et allouées lors de la

compilation des kernels et en lecture seule : leurs valeurs ne peuvent pas être changées lors de l'exécution.

Ensuite, chaque work-group a accès à une mémoire locale (identifiée par le préfixe `__local`). Les accès à cette mémoire cache sont très rapides, mais les valeurs qui y sont stockées ne sont accessibles que par les work-items du work-group et sont perdues à la fin de l'exécution du kernel. La taille de cette mémoire est petite, de l'ordre de quelques dizaines de kilo-octets, et a pour but d'y stocker les données temporaires partagées au sein du work-group.

Enfin, un périphérique contient une mémoire globale (identifiée par le préfixe `__global`). Celle-ci est de l'ordre de quelques gigaoctets dans le cas d'un GPU et correspond à la mémoire vive du nœud dans le cas d'un CPU. Cette mémoire est accessible par tous les work-items exécutant un kernel. Les lectures et écritures y sont relativement lentes. De plus, l'efficacité des accès à cette mémoire est conditionnée par leur ordonnancement. La façon la plus efficace de procéder est d'y accéder de manière « coalescente » : c'est à dire lorsque les work-items dont les numéros se suivent accèdent à des données rangées de manière contiguë et dans le même ordre.

4.1.3 Les kernels OpenCL

Les sources des kernels peuvent être dynamiquement construites par le programme hôte et compilées au cours de l'exécution. Ceci nous permet d'y inclure des constantes qui facilitent leur programmation.

Lors de l'exécution d'un kernel par le périphérique OpenCL, nous avons aussi accès à des données contextuelles qui permettent de piloter la parallélisation. Nous pouvons par exemple connaitre le numéro du work-group et le numéro du work-item. Ces données permettent alors de paralléliser les calculs de la manière souhaitée.

Le nombre de work-items et la taille des work-groups associés à l'exécution d'un kernel sont spécifiés lors de la soumission de la tâche à la file d'attente. Tous les work-items exécutent alors le même kernel en parallèle. Cependant, dans le cas d'un GPU qui possède une architecture SIMD (pour « *Simple Instruction, Multiple Data* »), contrairement aux CPU qui possèdent une architecture MIMD (pour « *Multiple Instructions, Multiple Data* »), les work-items d'un même work-group partagent la même unité d'instructions, c'est-à-dire que les instructions conditionnelles conduisent à l'exécution successive de toutes les branches et seules les écritures sont désactivées dans les branches rejetées. Ainsi, nous devons limiter au maximum le nombre de branches présentes dans les kernels.

4.2 Implémentation GPU

Initialement, l'implémentation de l'algorithme GD a plus spécifiquement été adaptée aux GPU. Ce type de périphérique possède un volume de mémoire limité. Afin de réduire le volume de données, certains paramètres constants sont donc calculés lorsque requis. De plus, il est nécessaire d'utiliser les ressources de mémoire locale pour y stocker les résultats intermédiaires, mais aussi certaines données auxquelles l'accès est fréquent.

Remarque 14. *Nous présentons dans cette section les kernels optimisés pour hexaèdres. Le solveur possède aussi les kernels génériques qui permettent d'effectuer des calculs sur tétraèdres. Ces kernels génériques sont semblables aux kernels optimisés, mais sans tenir compte des simplifications liées aux propriétés des fonctions de base.*

4.2.1 Zones homogènes

Un GPU est composé de plusieurs milliers de work-items travaillant en parallèle. Afin que leur utilisation soit optimale, l'exécution d'un kernel doit solliciter tous les work-items disponibles. Ce maximum ne peut pas être atteint avec un kernel traitant une seule ou peu de mailles à la fois. Les mailles sont donc réparties dans des zones homogènes. Chaque zone contient une unique géométrie de maille ainsi qu'une même base d'approximation.

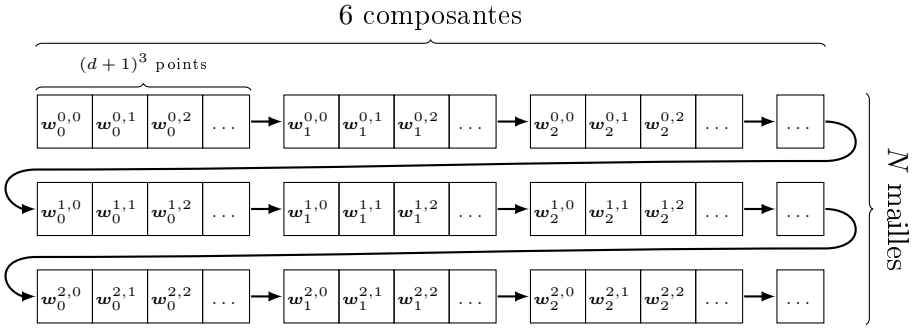
Cette répartition en zones homogènes nous permet de spécifier et simplifier les kernels de calcul afin de limiter au maximum les branches conditionnelles qui dégradent les performances. Chaque zone de maillage est transformée en tableaux de données dans la phase de pré-traitement et à destination des kernels OpenCL. Ainsi, les données géométriques se résument en 3 tableaux : les coordonnées des nœuds (réels), la définition des mailles par les indices de leurs sommets (entiers) et la connectivité entre les faces des mailles voisines (entiers). La transformation géométrique associée à chaque maille, les coordonnées des points d'interpolation et la valeur des fonctions de base sont recalculées dès que nécessaires.

Stockage coalescent

Le champ électromagnétique et sa dérivée en temps sont stockés dans $p + 1$ autres tableaux de réels, où p est généralement l'ordre du schéma en temps. Il existe des implémentations dites « *low storage* » de certains schémas qui permettent de réduire le nombre de tableaux. Les valeurs des champs sont rangées dans les tableaux de façon à ce que les accès en mémoire

soient coalescents comme illustré dans la figure 4.2. Cet ordre est choisi pour optimiser les accès en mémoire globale du kernel calculant le terme de volume.

FIGURE 4.2 – Ordre de stockage des champs électromagnétiques dans les tableaux passés aux kernels de calcul. $\boldsymbol{w}_k^{i,j}$ représente la k-ième composante du champ électromagnétique au j-ième point d'approximation de la i-ième maille.



Géométrie

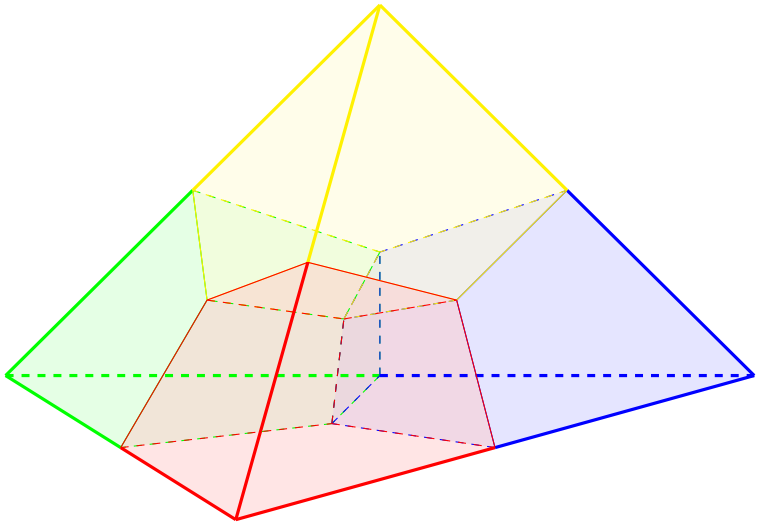
En pratique, seuls les hexaèdres ont été implémentés dans le solveur. Il est toutefois possible de traiter des maillages composés de tétraèdres. Ces derniers sont alors découpés en hexaèdres dans le pré-traitement (figure 4.3). Toutefois, n'importe quel type de géométrie pourrait être implémenté dans le solveur à condition d'être en mesure de fournir les fonctions géométriques associées ainsi qu'un espace d'approximation. Les hexaèdres ont été choisis pour leurs propriétés qui nous permettent d'optimiser et paralléliser le code de manière plus aisée.

Les kernels de calcul sont alors appliqués zone par zone dans l'ordre suivant pour déterminer la dérivée en temps du champ électromagnétique :

— terme de volume (3.2.2) ;
— terme de flux (3.2.3) ;
— terme de masse (3.2.1).

Dans la plupart des schémas temporels, la mise à jour des champs est effectuée avec une étape de type Euler. Le graphe des tâches du schéma couplé GD-RK2 est donné dans la figure 4.4.

FIGURE 4.3 – Découpage classique d'un tétraèdre en 4 hexaèdres. Les sommets des hexaèdres sont les sommets du tétraèdre ainsi que les centres de gravité de ses arêtes, de ses faces et de ses sommets.



4.2.2 Étape de type Euler

La mise à jour des champs est effectuée avec une étape de type Euler dont la parallélisation est aisée. En effet, chaque valeur du tableau des dérivées en temps est utilisée pour mettre à jour la valeur du champ située à la même position dans un autre tableau ordonné à l'identique.

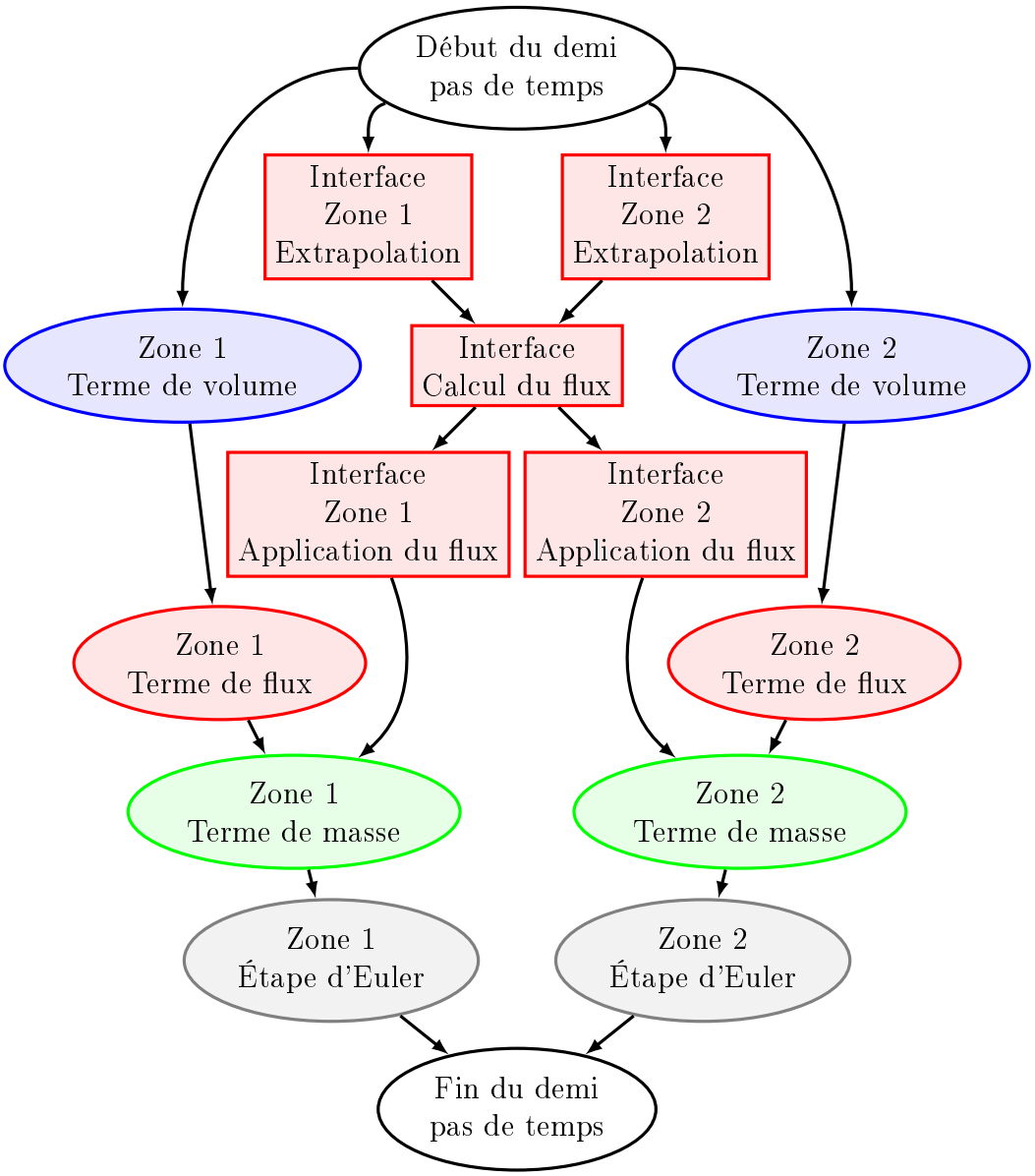
Ce kernel (figure 4.5) n'utilise aucune donnée géométrique ou d'interpolation, nous pouvons donc associer un work-item à chaque composante du champ sans tenir compte de la répartition en zones. Ainsi, chacun effectue le calcul pour la composante qui lui est associée.

Enfin, les lectures et écritures en mémoire sont coalescentes dans ce cas et l'utilisation de la mémoire locale s'avère inutile puisque aucun calcul intermédiaire n'est effectué.

4.2.3 Kernel du terme de masse

L'intégrale (3.28) qui représente le terme de masse est calculée par un kernel parallélisé sur un nombre de work-items égal au nombre de points d'approximation présents dans le maillage. Les points d'approximation d'une même maille sont regroupés dans un work-group. Nous notons $i_L \in [\![0\,;\,N-1]\!]$

FIGURE 4.4 – Graphe des tâches d'un demi pas de temps du schéma couplé GD-RK2 pour deux zones homogènes.



le numéro du work-group courant et $i \in [\![0\,;\,(d+1)^3-1]\!]$ le numéro du work-item courant dans son work-group.

Dans un premier temps, chaque work-item lit en mémoire globale les coordonnées des nœuds de la maille à laquelle il est associé. Ensuite, chaque work-item calcule le poids d'intégration ω_i de la fonction de base $\hat{\psi}_i$ associée au point d'approximation $\hat{g}_i$ et le jacobien de la transformation géométrique

FIGURE 4.5 – Kernel de calcul de l'étape Euler.

```
__kernel void euler(double dt,
                    __global const double* Wn,
                    __global const double* dtWn,
                    __global double* Wnp1) {
    // Numero du work-item
    const int i = get_global_id(0);

    // Mise a jour
    Wnp1[i] = Wn[i] + dt * dtWn[i];
}
```

en ce même point. Enfin, la dérivée en temps des champs au point d'approximation $\hat{g}_i$ est mise à jour en mémoire globale en la divisant par le produit $\omega_i \mathrm{jac}\,(\tau_L)\,(\hat{g}_i)$.

Dans ce kernel, le regroupement des work-items par maille est uniquement utilisé pour récupérer les nœuds de la maille.

4.2.4 Kernel du terme de volume

L'intégrale (3.31) qui représente le terme de volume est calculée par un kernel parallélisé sur un nombre de work-items égal au nombre de fonctions de base définies sur la zone considérée. Les fonctions de base à support sur une même maille sont regroupées dans un work-group. Cette répartition est équivalente à celle utilisée pour le terme de masse. Nous notons $i_L \in [\![0\,;\, N-1]\!]$ le numéro du work-group courant et $i \in [\![0\,;\, (d+1)^3 - 1]\!]$ le numéro du work-item courant dans son work-group.

Dans un premier temps, chaque work-item lit en mémoire globale les coordonnées des nœuds de la maille à laquelle il est associé. Cette étape est nécessaire et requiert plusieurs lectures en mémoire non coalescentes car le maillage est généralement non structuré. Ensuite, les champs de la maille i_L sont lus et stockés en mémoire locale. Cette lecture est faite de manière parallèle : chaque work-item lit les champs du point d'approximation qui lui est associé. La coalescence des données lues est assurée par l'ordre de stockage des champs illustré par la figure 4.2. Après le chargement des champs en mémoire locale, une synchronisation locale est nécessaire.

Nous calculons alors l'intégrale (3.31) sur les mailles hexaédriques en ti-

rant parti de la propriété vérifiée par le gradient des fonctions de base exposée dans la section 3.2.2. Pour un work-item d'indice i, nous évaluons cette intégrale en ne considérant que les points alignés avec le point d'indice i. Observons alors les termes de cette somme :

$$\sum_j \omega_j F\left(\boldsymbol{w}_{L,j}, \boldsymbol{w}_{L,j}, \operatorname{com}(\tau'_L)\,\hat{\nabla}\hat{\psi}_i(\hat{g}_j)\right). \tag{4.1}$$

Les termes demandant le plus de calcul sont le flux F et la comatrice $\operatorname{com}(\tau'_L)$. Le flux dépend des deux indices i et j, alors que la comatrice ne dépend que de l'indice j. Nous pouvons aussi rappeler que le gradient $\hat{\nabla}\hat{\psi}_i$ s'annule en tout point d'approximation non aligné avec $\hat{g}_j$. Nous parallélisons ce calcul en attribuant à un work-item le calcul des termes pour un indice i fixé (fonctions de base) afin d'éviter les écritures concurrentes qu'il faudrait gérer en parallélisant par rapport à l'indice j (points d'approximation). Pour cela, nous considérons le multi-indice (p_0, p_1, p_2) tel que $i = p_0 + (d+1)(p_1 + (d+1)p_2)$. Les indices des points alignés avec $\hat{g}_i$ dans une direction sont obtenus en faisant varier la composante du multi-indice de cette direction.

Soit R_d l'application qui à un entier associe le reste de la division euclidienne de cet entier par $d+1$. Nous pouvons alors définir les 3 fonctions d'indices :

$$\begin{aligned} i_0(\alpha) &= R_d(p_0 + \alpha) + (d+1)(p_1 + (d+1)p_2), \\ i_1(\alpha) &= p_0 + (d+1)(R_d(p_1 + \alpha) + (d+1)p_2), \\ i_2(\alpha) &= p_0 + (d+1)(p_1 + (d+1)R_d(p_2 + \alpha)). \end{aligned} \tag{4.2}$$

pour α variant dans l'intervalle $[\![0\,;\,d]\!]$.

Chaque work-item (fonction de base) ajoute les contributions qu'il calcule à un tableau privé contenant la dérivée en temps au point d'approximation associé. Enfin, la dérivée en temps est mise à jour en mémoire globale par des accès coalescents.

Le calcul du terme de volume est très bien adapté à la parallélisation et l'algorithme décrit ci-dessus est donné en pseudo-code dans la figure 4.6.

Remarque 15. *Les champs en mémoire globale sont stockés de façon à ce que les accès soient coalescents. Ce rangement des données est nécessaire afin d'assurer l'efficacité des lectures et écritures.*

Remarque 16. *L'utilisation de la mémoire locale est impérative. Dans le cas contraire, un grand nombre d'accès non coalescents à la mémoire globale seraient effectués, dégradant fortement les performances.*

Remarque 17. *Le chargement en mémoire locale s'effectue en parallèle : chaque work-item charge les champs qui lui sont associés. Après le chargement en mémoire locale, les work-items sont synchronisés, ce qui assure que toutes les données utiles pour la suite du calcul sont disponibles.*

FIGURE 4.6 – Algorithme parallèle du calcul du terme de volume.

```
iel // indice du work-group (de l'element)
i // indice du work-item (de la fonction de base)
S // sommets de l'element

// Copie des champs de l'element en memoire locale
local_Wn[i] = global_Wn[iel][i]

Synchronisation_locale

// Initialisation de la derivee des champs
dtWn = 0

for (dir = 0; dir < 3; ++dir) { // directions
  for (a = 0; a < d + 1; ++a) { // alignement
    // Calcul de l'indice j
    j = i_dir(a)

    // Donnees d'interpolation (utilise S)
    g_j // point d'integration j
    w_j // poids d'integration j

    // Calcul de la comatrice
    // de la transformation (utilise S et g_j)
    codtau = comatrice au point j

    // Calcul du gradient
    gradpsi = gradient i au point j

    // Ajout des contributions en memoire locale
    dtWn += w_j *
        F(local_Wn[j], local_Wn[j], codtau * gradpsi)
  }
}

// Mise a jour en memoire globale
global_dtWn[iel][i] += dtWn
```

Remarque 18. *Au cours de la boucle sur les points j, nous ne calculons pas toutes les composantes du gradient de la fonction de base. En effet, puisque i est différent de j, seule la composante correspondant à la direction d'alignement de i et j est non nulle.*

Remarque 19. *Dans le cas des mailles non hexaédriques, la propriété de nullité du gradient des fonctions de base ne s'applique pas et tous les termes de la somme approximant le terme de volume doivent être calculés dans les 3 directions de l'élément de référence.*

4.2.5 Kernels du terme de flux

Le terme de flux représente les échanges aux interfaces entre les mailles d'une même zone homogène. Le raccord entre ces mailles est conforme et leurs ordres d'interpolation sont identiques.

Rappelons la formule d'approximation du terme de flux entre deux mailles homogènes (3.40) :

$$\omega_{f,\pi(f,i)}\hat{\psi}_i(\hat{g}_{f,\pi(f,i)})F\left(\boldsymbol{w}_{L,f,\pi(f,i)}, \boldsymbol{w}_{R,f',\sigma_{LR}(\pi(f,i))}, \mathrm{com}\left(\tau'_L\right)\hat{\boldsymbol{n}}\right). \tag{4.3}$$

Cette intégrale est calculée en trois étapes. La première étape consiste en l'extrapolation des champs du volume sur les faces de chaque maille. Ensuite, les flux sont calculés à partir des champs extraits dans la seconde étape, puis appliqués à la dérivée en temps dans la dernière étape. Chacune de ces étapes est effectuée par un kernel distinct.

Extrapolation des champs

L'extrapolation est parallélisée sur un nombre de work-groups égal au nombre de mailles présentes dans la zone considérée. Chaque work-group est composé d'autant de work-items qu'il y a de points d'approximation sur les faces d'une maille. Nous notons $i_L \in [\![0\,;\, N-1]\!]$ le numéro du work-group courant et i_b le numéro du work-item courant dans son work-group. L'indice de la face $f \in [\![0\,;\, 5]\!]$ associé à chaque work-item ainsi que l'indice du point d'approximation $i_f \in [\![0\,;\, (d+1)^2-1]\!]$ sur cette face sont calculés à partir de l'indice i_b du work-item tels que :

$$i_b = i_f + f(d+1)^2. \tag{4.4}$$

Dans un premier temps, les work-items se partagent la lecture en mémoire globale des champs de la maille i_L et les stockent en mémoire locale. Cette étape est effectuée comme dans le calcul des termes de volume, par les $(d+1)^3$ premiers work-items.

Remarque 20. *Jusqu'à l'ordre d'interpolation* 5 *y compris, le nombre de points de faces d'une maille est supérieur (ou égal) au nombre de points volumiques. En effet, lorsque* $d = 5$, $6(d+1)^2 = (d+1)^3$. *Au-dessous, certains work-items restent inactifs au cours de cette étape. Au-delà, certaines données ne sont pas chargées en mémoire locale, mais en pratique nous n'utilisons jamais d'interpolation d'ordre supérieur à* 5.

Chaque work-item calcule alors la valeur des champs provenant de la maille sur le point de face qui lui est associé. Pour cela nous commençons par calculer les coordonnées du point de face i_b dans l'élément de référence. Ensuite, nous ajoutons les contributions des fonctions de base en ce point. Dans notre cas de mailles hexaédriques, seules les fonctions de base associées aux points d'approximation alignés avec le point de face considéré sont non nulles en ce point (proposition 9). Ainsi, seules $d + 1$ contributions sont à prendre en compte, contre celles de toutes les fonctions de base de la maille dans le cas non hexaédrique.

Ces champs extrapolés sont ensuite stockés de façon coalescente en mémoire globale à destination du kernel chargé de calculer le flux. L'algorithme décrit ci-dessus est donné en pseudo-code dans la figure 4.7.

Calcul des flux

Le calcul des flux est parallélisé de la même façon que l'extrapolation des champs : un nombre de work-groups égal au nombre de mailles présentes dans la zone considérée et autant de work-items qu'il y a de points d'approximation sur les faces d'une maille. Nous notons i_L le numéro du work-group courant et i_b le numéro du work-item courant dans son work-group. L'indice de la face f ainsi que l'indice du point d'approximation i_f sont calculés comme précédemment (4.4).

Dans un premier temps, chaque work-item lit en mémoire globale les coordonnées des nœuds de la maille à laquelle il est associé. Comme pour le kernel de volume, cette étape est nécessaire et requiert plusieurs lectures en mémoire non coalescentes car le maillage est généralement non structuré. Les work-items chargent aussi en mémoire locale les champs extraits dans le kernel d'extrapolation au point de face associé. Cette lecture est faite en accès coalescent.

Ensuite, lorsque le flux numérique classique s'applique (1.41), en l'absence de modèle (ou condition) de surface spécifique (chapitre 2), nous chargeons les champs de face extraits de la maille voisine R. Pour cela nous calculons l'indice du point de face de la maille R correspondant au point courant de la maille L grâce aux données de connectivité des mailles et en tenant compte

FIGURE 4.7 – Algorithme parallèle de l'extrapolation des champs dans le calcul du terme de flux.

```
iel // indice du work-group (de l'element)
ib // indice du work-item (du point de bord)

f = ib / (d + 1)^2 // indice de face
i = ib % (d + 1)^2 // indice du point sur la face

// Copie des champs de l'element en memoire locale
if (ib < (d + 1)^3)
  local_Wn[ib] = global_Wn[iel][ib]

Synchronisation_locale

// Calcul des coordonnees du point de reference
g^_{f,i} // point i sur face f

// Initialisation de la valeur des champs
face_Wn = 0

dir // direction normale a la face f
for (a = 0; a < d + 1; ++a) { // alignement
  // Calcul de l'indice de la fonction de base
  j = i_dir(a)

  // Extrapolation
  face_Wn += psi_j(g^_{f,i}) * local_Wn[j]
}

// Stockage en memoire globale
global_face_Wn[iel][ib] = face_Wn
```

des orientations des faces. Cette lecture est généralement faite en accès non-coalescent, compte tenu de la structure du maillage.

Pour terminer cette étape, chaque work-item calcule le flux numérique au point qui lui est associé. Nous devons pour cela calculer les coordonnées physiques du point d'intégration ainsi que la normale en ce point. Les coordonnées physiques ne servent que pour certains flux spécifiques, mais elles

sont nécessaires pour le calcul de la normale.

Ces flux sont ensuite stockés en mémoire globale de façon coalescente à destination du kernel chargé d'ajouter leur contribution au tableau de la dérivée temporelle de la solution. L'algorithme décrit ci-dessus est donné en pseudo-code dans la figure 4.8.

Application à la dérivée

Ce kernel est parallélisé de la même façon que le kernel de volume : un nombre de work-groups égal au nombre de mailles présentes dans la zone considérée et autant de work-items qu'il y a de fonctions de base définies sur la maille considérée. Nous notons i_L le numéro du work-group courant et i le numéro du work-item courant dans son work-group.

Dans un premier temps, les work-items se partagent la lecture en mémoire globale des flux calculés sur les faces de la maille i_L et les stockent en mémoire locale. Comme précisé dans la remarque 20, le nombre de points de volume (ou de fonctions de base) est généralement inférieur au nombre de points de faces. Une boucle optimisée a donc été implémentée afin de charger ces flux en réduisant l'inactivité de certains work-items : à chaque itération, $(d+1)^3$ valeurs sont copiées en accès coalescent. À l'ordre 0, 6 itérations sont nécessaires ; à l'ordre 1, 3 itérations sont nécessaires ; aux ordres 2 à 4, 2 itérations sont nécessaires ; et aux ordres supérieurs à 5, une seule itération est nécessaire. Après le chargement des flux en mémoire locale, une synchronisation locale doit être effectuée.

Ensuite, chaque work-item i calcule l'intégrale (3.40) pour chaque face de la maille. En appliquant la propriété 9, ce calcul ne nécessite que la contribution d'un flux par face : pour une face f donnée, le seul flux utilisé est celui associé au point de face sur lequel le point de volume est projeté. Chaque work-item (fonction de base) ajoute les contributions qu'il calcule à un tableau privé contenant la dérivée en temps au point d'approximation associé.

Enfin, la dérivée en temps est mise à jour en mémoire globale par des accès coalescents. L'algorithme décrit ci-dessus est donné en pseudo-code dans la figure 4.9.

4.2.6 Kernels du terme d'interface

Le terme d'interface représente les échanges entre les mailles de 2 zones homogènes distinctes. Le raccord entre ces mailles n'est pas nécessairement conforme ou leurs ordres d'interpolation sont distincts. Dans ce cas, les points d'interpolation des faces ne coïncident pas.

FIGURE 4.8 – Algorithme parallèle du calcul du flux.

```
iel // indice du work-group (de l'element)
ib // indice du work-item (du point de bord)
S // sommets de l'element
t // temps physique courant

f = ib / (d + 1)^2 // indice de face
i = ib % (d + 1)^2 // indice du point sur la face

// Copie des champs du point de face
WL = global_face_Wn[iel][ib]

// Copie des champs du point de face voisin
jel // indice de la maille voisine
if (jel) {
  jb // indice du point voisin (utilise jel, i et f)
  WR = global_face_Wn[jel][jb]
}

// Donnees d'interpolation (utilise S, i et f)
g_i // point d'integration i
w_i // poids d'integration i
n_i // normale au point i

// Initialisation de la valeur du flux
face_flux = 0

// Calcul du flux (utilise aussi g_i et t en fonction
// du type de flux)
face_flux = F(WL, WR, n_i)

// Stockage en memoire globale
// Application directe du poids d'integration
global_face_flux[iel][ib] = face_flux * w_i
```

Nous identifions alors le côté le plus finement discrétisé spatialement (remarque 21), supposons qu'il s'agit du côté gauche. Les points d'interpolation de face des mailles de gauche situés sur l'interface servent de points d'extra-

FIGURE 4.9 – Algorithme parallèle de l'application des flux à la dérivée temporelle.

```
iel // indice du work-group (de l'element)
i // indice du work-item (de la fonction de base)

// Copie des flux ponderes en memoire locale
// Boucle optimisee
j_max = ceil(6 * (d + 1)^2 / (d + 1)^3)
for (j = 0; j < j_max; ++j) {
  ib = i + j * (d + 1)^3 // point de face
  if (ib < 6 * (d + 1)^2)
    face_flux[ib] = global_face_flux[iel][ib]
}

Synchronisation_locale

// Initialisation de la derivee des champs
dtWn = 0

for (f = 0; f < 6; ++f) { // faces
  j // indice du point de face aligne
  jb = j + f * (d + 1)^2 // indice du point de bord

  // Calcul des coordonnees du point de reference
  g^_{f,j} // point j sur face f

  // Application du flux pondere
  dtWn -= psi_i(g^_{f,j}) * face_flux[jb]
}

// Mise a jour en memoire globale
global_dtWn[iel][i] += dtWn
```

polation des champs. Ainsi, sur une maille L du côté gauche, le comportement des kernels est identique à un simple terme de flux homogène (section 4.2.5).

Du côté droit, sur une maille R, puisque les points d'extrapolation (ou d'interface) ne sont pas les projetés des points de volume, l'extrapolation et l'application des flux ne bénéficient pas des simplifications apportées par

les propriétés des fonctions de base. Ainsi, l'extrapolation des champs sur les points d'interface nécessite de calculer la somme complète (3.41) pour les $(d+1)^3$ fonctions de base (contre $d+1$ dans le cas conforme). Les coordonnées des points d'interface dans l'élément de référence associé à la maille R doivent être calculées en approchant la réciproque de la transformation τ_R par la méthode de Newton, appliquée au point d'interface physique. De plus, le flux calculé en chaque point d'interface de la maille R doit ajouter sa contribution à la dérivée temporelle pour les $(d+1)^3$ fonctions de base (là encore contre $d+1$ dans le cas conforme).

Remarque 21. *En pratique, nous utilisons des maillages à connectivité conforme et n'avons que* 2 *cas de figure à traiter dans le solveur :*

— *raccord de face conforme et ordre d'interpolation distinct ;*

— *raccord* 1/6 *en bord de domaine (figure 4.10).*

Dans le premier cas, nous extrapolons les champs sur les points de face du côté d'ordre le plus élevé. Ce choix est stratégiquement le meilleur dans le but de diminuer la quantité de calculs et d'améliorer la précision. En effet, nous bénéficions ainsi des simplifications liées aux propriétés des fonctions de base du côté le plus coûteux.

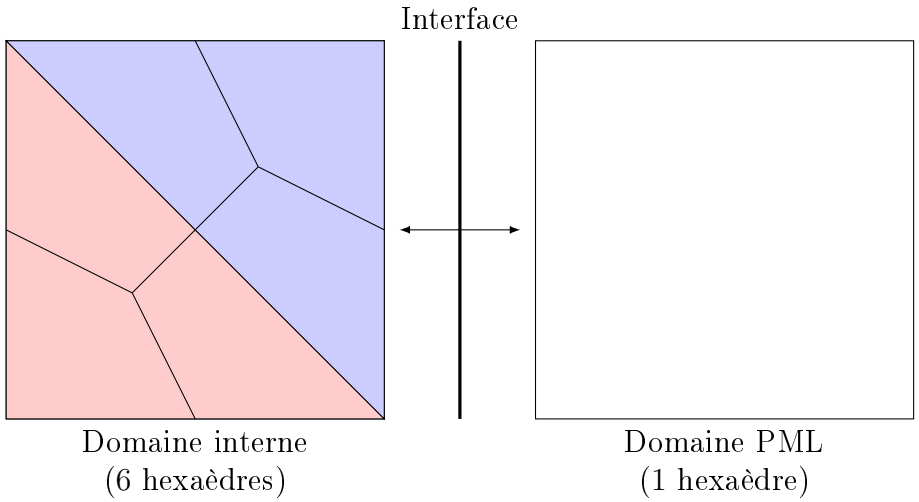
Le second cas se présente lors de l'ajout de couches PML (section 2.1.5) autour d'un maillage en tétraèdres découpés (figure 4.3). Nous obtenons alors un raccord d'une face contre 6 *à cette interface. Nous extrapolons alors les champs sur les points de face du côté le plus raffiné : le côté interne, non PML.*

4.2.7 Parallélisation MPI

L'architecture classique des (super)calculateurs possédant des GPU est d'associer un GPU à chaque nœud de calcul. Dans le cas des CPU (section 4.3), chaque périphérique est par nature un nœud de calcul. Chaque nœud de calcul possède son propre espace mémoire. Ainsi, afin de paralléliser les calculs sur plusieurs périphériques OpenCL, nous utilisons une bibliothèque MPI [14]. Celle-ci permet de réaliser des calculs sur des machines à mémoire distribuée. Plusieurs processus sont lancés en parallèle. Chaque processus possède son propre espace mémoire. Les différents processus communiquent alors entre eux par un système de messages.

Cette bibliothèque permet aussi d'effectuer des calculs sur une machine à mémoire partagée. Par exemple, les machines de calcul possédées par Axes-Sim qui associent plusieurs GPU à un unique nœud de calcul. Dans ce cas, la mémoire de la machine doit être suffisamment grande pour accueillir tous les

FIGURE 4.10 – Schéma représentant le raccord 1/6 entre les couches de PML adjointes et le maillage initial en tétraèdres découpés (figure 4.3). Les faces des tétraèdres découpés sont représentées en bleu et en rouge.



processus, car, même si ces derniers sont hébergés dans le même espace mémoire, chacun gère sa propre copie de la mémoire allouée par le programme au cours du pré-traitement.

Pour paralléliser les calculs sur plusieurs périphériques, le maillage global est réparti de manière équilibrée entre les différents processus MPI. Cette subdivision du maillage est effectuée à l'aide de la bibliothèque de partitionnement `metis` [43]. Chaque processus utilise un périphérique OpenCL différent pour effectuer les calculs sur le sous-domaine qui lui a été associé.

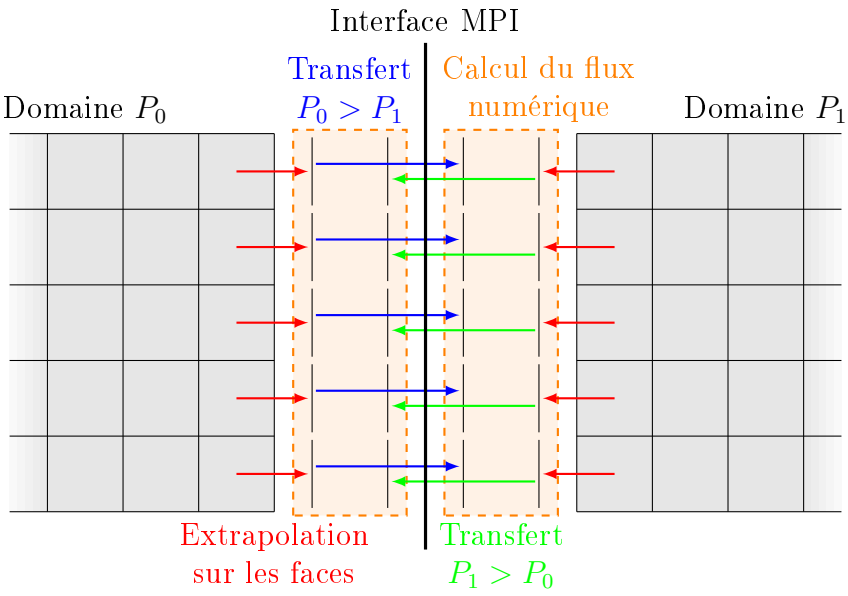
Les communications sont effectuées aux interfaces entre les sous-domaines, lors du calcul du terme de flux. Une telle interface est appelée « interface MPI ». Le calcul du flux nécessite la connaissance des champs provenant des mailles voisines. Si la maille voisine est traitée par un processus différent, l'information doit être récupérée au préalable.

Pour réduire le volume des données communiquées, chaque processus effectue l'extrapolation des champs de son côté de l'interface MPI. Ceci permet de diviser le nombre de données échangées par $d+1$ contrairement à une méthode qui consisterait à échanger les données volumiques des mailles voisines de l'interface. De plus, nous sommes assurés que les données à échanger sont stockées de manière contiguë dans un unique tableau.

Une fois les champs extraits des deux côtés, les processus partageant une

interface MPI s'échangent ces données afin de calculer le flux. Cet échange de données est effectué de manière asynchrone à l'aide de threads dédiés (figure 4.11). Pour chaque interface MPI et à chaque calcul du terme de flux, deux threads sont générés (un par processus) : le processus noté P_0, respectivement P_1, transfère les données extraites du périphérique de calcul à l'hôte et les échange (envoi, réception) avec le processus P_1, respectivement P_0, puis transfère les données reçues de l'hôte au périphérique de calcul. Ensuite, chaque domaine effectue le calcul et l'application du flux.

FIGURE 4.11 – Schéma représentant une interface MPI.



Comme nous l'avons précisé dans la section 4.1.1, la soumission et l'ordonnancement des tâches sont gérés par un graphe des tâches. Ce graphe des tâches utilise les événements de l'API OpenCL pour contrôler l'ordre des tâches. Afin d'intégrer un thread de communication MPI dans le graphe des tâches, son exécution est conditionnée par la complétion d'une liste d'événements OpenCL (ceux associés aux kernels d'extrapolation) et un nouvel événement OpenCL est généré à sa soumission. Ce nouvel événement est marqué comme étant terminé à la fin du thread de communication MPI. C'est aussi cet événement qui conditionnera l'exécution du kernel de calcul du flux d'interface MPI.

Le partitionnement du maillage effectué dans le but de diviser le problème simulé sur plusieurs périphériques de calcul génère des interfaces supplémen-

taires qui augmentent le coût de calcul de la simulation. Ce surcoût diminue l'efficacité du solveur qui doit être étudiée dans le but de valider l'implémentation MPI. Nous étudions l'efficacité MPI (ou scalabilité) dans la section 7.4.

4.3 Adaptations CPU

Les kernels optimisés pour GPU présentés dans la section 4.2 fonctionnent aussi sur CPU mais ne sont pas optimisés pour ces derniers. La différence d'architecture — SIMD pour les GPU *versus* MIMD pour les CPU — implique une différence de performances pour un même kernel.

Les axes d'optimisation sont documentés par les constructeurs des CPU (Intel [40], AMD [1]). Afin d'améliorer les performances d'un kernel sur CPU, les points suivants sont à respecter :

— n'utiliser que la mémoire globale ;
— augmenter la sérialisation des traitements ;
— préférer stocker les résultats récurrents ;
— solliciter les unités de calcul vectoriel.

Nous présentons dans la suite les gains de performance constatés suite à l'application de ces différentes étapes [74]. Les accélérations présentées ont été obtenues à l'ordre d'interpolation spatiale $d = 2$.

Le cas d'application utilisé est la propagation d'une onde plane dans une grille régulière (section 7.1). Nous procédons à quelques itérations sur un maillage cube de 90 mailles par côté. Les périphériques utilisés pour ces résultats sont un CPU Intel Core i7-920 et un GPU NVidia GeForce GTX 460. Initialement, ce CPU est 15.97 fois plus lent que ce GPU lorsque nous utilisons les mêmes kernels (optimisés pour GPU). La répartition en temps de calcul des différents kernels GPU est présentée dans le tableau 4.12.

FIGURE 4.12 – Comparaison de la répartition en temps de calcul sur CPU Intel Core i7-920 et GPU NVidia GeForce GTX 460 des principaux kernels de calcul optimisés pour GPU (avant spécialisation pour CPU).

	GPU	CPU
Temps total (s)	17.49	279.38
Volume (%)	23.47	47.57
Flux (%)	68.75	48.71
Masse (%)	5.17	2.58
Euler (%)	1.88	0.81

Étant déjà en possession de kernels optimisés GPU, l'objectif de ces travaux a été de les optimiser au mieux à destination des CPU, tout en conservant leur essence. Cette approche nous a permis d'obtenir de bonnes performances, tout en limitant le coût de développement.

4.3.1 Suppression de l'utilisation de la mémoire locale

Pour les CPU, la mémoire locale est généralement de nature identique à la mémoire globale, dans les deux cas il s'agit de la mémoire vive du nœud. Cette information est donnée par l'identifiant `CL_DEVICE_LOCAL_MEM_TYPE` dans l'API OpenCL. Contrairement aux GPU pour lesquels le temps d'accès à la mémoire locale est bien plus rapide que le temps d'accès à la mémoire globale, dans le cas du CPU ce temps est identique.

Nous avons modifié les kernels du terme de volume 4.2.4 et de flux 4.2.5 afin qu'ils ne chargent plus les champs de la maille considérée en mémoire locale, les kernels du terme de masse et l'étape d'Euler n'utilisant pas la mémoire locale. Les champs sont alors directement lus en mémoire globale dès que utilisés dans les calculs.

Puisque la mémoire locale n'est plus utilisée, nous ne renseignons plus la taille des work-groups à l'exécution de ces kernels. Ce choix est laissé à l'API OpenCL qui regroupe les work-groups au mieux en fonction du périphérique cible (comportement par défaut). Nous retrouvons l'indice de la maille et de la fonction de base à traiter à l'aide d'une division euclidienne de l'indice global du work-item.

Suite à ces modifications (et d'autres simplifications qui en découlent), nous avons constaté un facteur d'accélération de 2.39 pour le kernel du terme de volume.

4.3.2 Sérialisation

L'objectif d'un CPU est la polyvalence. Chaque cœur de calcul fonctionne indépendamment avec ses propres unités d'instruction dans le but d'exécuter des traitements distincts en parallèle (c'est le principe du « *multitasking* »). Il n'est donc pas nécessaire de paralléliser le code à l'échelle de l'instruction, au contraire, l'augmentation de la sérialisation au sein d'un kernel accorde plus de liberté aux cœurs de calcul et produit de meilleures performances. Cette approche permet aussi de bénéficier de l'effet de cache dans l'accès aux données.

Nous avons modifié le kernel du terme de volume 4.2.4 afin que chaque work-item traite, non plus une fonction de base, mais l'ensemble des fonctions de base d'une maille. Pour cela il a suffi d'ajouter une boucle sur les fonctions de base englobant le traitement déjà codé dans le kernel. Le nombre de work-items passe donc de $N(d+1)^3$ à N.

Nous avons aussi modifié les kernels du terme de flux 4.2.5 afin que chaque work-item traite, non plus un point de face, mais l'ensemble des points d'une face dans le cas des étapes d'extrapolation et de calcul du flux. L'application des flux a été adaptée comme le kernel de volume. De plus, les deux premières étapes (extraction des champs et calcul du flux numérique) ont été regroupées en un seul kernel et les flux sont calculés une seule fois par interface entre mailles, contre un calcul de flux de chaque côté de l'interface dans le cas du kernel optimisé pour GPU. Le nombre de work-items passe donc de $6N(d+1)^2$ à $3N$, dans le cas de ce kernel fusionné. L'exécution d'un seul kernel donne de meilleures performances. Ce choix revient à supprimer une synchronisation globale des work-items.

Suite à ces modifications, nous avons constaté un facteur d'accélération de 1.40 pour le kernel du terme de volume. Dans le cas des kernels du terme de flux, la combinaison de la suppression de l'utilisation de la mémoire locale et de la sérialisation des traitements a mené à une accélération de 2.24.

4.3.3 Stockage des données récurrentes

Dans l'implémentation GPU, compte tenu de la lenteur des accès mémoire par rapport à la vitesse de calcul et de l'espace mémoire limité, nous avons fait le choix de calculer à chaque itération un certain nombre de données constantes, propres à chaque maille, dont notamment : le gradient des fonctions de base qui intervient dans le flux numérique du terme de volume et la valeur de la normale aux faces qui intervient dans le flux numérique du terme de flux.

En comparaison, la taille mémoire associée aux CPU d'une machine de calcul est colossale : généralement de 64 à 256 Go contre 8 à 16 Go de mémoire sur un GPU conçu pour le calcul. Ces données constantes, récurrentes et propres à chaque maille sont donc pré-calculées puis passées en argument aux kernels, comme pour les tableaux des champs, des nœuds, de connectivité, *etc.*

Suite à ces modifications, nous avons constaté des facteurs d'accélération de 4.45 pour le kernel du terme de volume et 1.3 pour les kernels du terme de flux avec une augmentation de la taille mémoire de 57 %.

4.3.4 Unités de calcul vectoriel

Une action supplémentaire pour améliorer les performances sur CPU est l'utilisation des unités de calcul vectoriel SSE (pour « *Streaming SIMD Extensions*) ou AVX (pour « *Advanced Vector Extensions* ») lorsqu'ils en sont pourvus. Ce sont des architectures pour lesquelles les registres sont de plus grande taille, pouvant ainsi stocker plusieurs valeurs à traiter en parallèle. Dans le cas AVX-512 (dernière génération, avec registres de 512 bits), il est possible de traiter simultanément jusqu'à 16 opérations de nombres réels en simple précision et 8 en double précision. Ces valeurs correspondent aussi au facteur d'accélération théorique lié à l'utilisation de ces capacités vectorielles du CPU.

Cependant, la complexité algorithmique de la méthode GD rend cette adaptation très difficile. Pour être pleinement efficace, les données devraient être ordonnées sous forme de vecteurs en mémoire vive afin que les kernels puissent directement appliquer le masque du type vectoriel (par exemple `float4` pour un vecteur de 4 réels simple précision). Ensuite, tous les kernels doivent traiter les données sous forme vectorielle (ou être adaptés pour extraire les données des vecteurs), ce qui implique de devoir adapter tout le code de calcul avant de pouvoir valider les développements et constater une accélération.

4.3.5 Bilan du gain de performances

Finalement, nous avons constaté une accélération d'un facteur 5.52 sur le cas d'application décrit plus haut. Le GPU NVidia GeForce GTX 460 que nous avons utilisé, avec les kernels optimisés pour GPU, n'est donc plus que 2.90 fois plus rapide que le CPU Intel Core i7-920 utilisé avec les kernels optimisés pour CPU. La répartition en temps de calcul des différents kernels après optimisations est présentée dans le tableau 4.13.

Ces résultats sont dépendants des périphériques utilisés. En effet, dans le cas d'une comparaison entre un CPU Intel Core i7-5820K et un GPU NVidia GeForce GTX 1070 sur 100 itérations du cas d'application 7.2, le facteur d'accélération constaté sur ce CPU avec l'utilisation des kernels optimisés CPU n'est que de 2.76. De plus, ce GPU reste 10 fois plus rapide que ce CPU, chacun utilisé avec les kernels optimisés pour son architecture. La répartition en temps de calcul des différents kernels pour ces périphériques est présentée dans le tableau 4.14.

FIGURE 4.13 – Comparaison de la répartition en temps de calcul sur CPU Intel Core i7-920 et GPU NVidia GeForce GTX 460 des principaux kernels de calcul optimisés pour l'architecture cible.

	GPU	CPU
Temps total (s)	17.49	50.63
Volume (%)	23.47	12.90
Flux (%)	68.75	70.94
Masse (%)	5.17	9.69
Euler (%)	1.88	4.47

FIGURE 4.14 – Comparaison de la répartition en temps de calcul sur CPU Intel Core i7-5820K et GPU NVidia GeForce GTX 1070 des principaux kernels de calcul optimisés pour l'architecture cible.

	GPU	CPU
Temps total (s)	2.02	20.4
Volume (%)	10.34	10.37
Flux (%)	65.42	63.88
Masse (%)	7.26	6.96
Euler (%)	7.63	11.67

4.4 Ordre d'interpolation adaptatif

Une autre amélioration apportée au solveur est ce que nous appelons l'ordre d'interpolation adaptatif.

Lorsque nous procédons au maillage du domaine de calcul, les contraintes géométriques peuvent générer de petites mailles. Ces petites mailles, sous contrainte d'une condition CFL de stabilité, induisent une réduction significative du pas de temps de la simulation, rallongeant ainsi le temps de simulation. Afin de limiter l'impact des petites mailles sur le temps de simulation, nous adaptons leur ordre d'interpolation. En réduisant l'ordre, nous augmentons la distance entre les points d'interpolation et par conséquent le pas de temps.

Au cours de la phase de maillage, nous utilisons la règle empirique du « $\lambda_{\min}$ sur 10 » pour déterminer la distance entre deux points d'interpolation, et par extension la longueur des arêtes des mailles. Le paramètre $\lambda_{\min}$ représente la longueur d'onde de la fréquence maximale $f_{\max}$ simulée. Nous

avons :

$$\lambda_{\min} = \frac{v}{f_{\max}} = \frac{1}{f_{\max}\sqrt{\varepsilon\mu}}, \tag{4.5}$$

avec v la vitesse de propagation dans le milieu considéré. Dans le cas des hexaèdres, la longueur approximative des arêtes l_H passée au mailleur est donc donnée par :

$$l_H = \frac{\lambda_{\min}}{10} d = \frac{vd}{10 f_{\max}}. \tag{4.6}$$

Dans le cas de maillages en tétraèdres, puisque chaque arête de tétraèdre est coupée en son milieu dans le but de former les hexaèdres (figure 4.3), nous retiendrons la formule suivante :

$$l_T = 2 l_H = \frac{2vd}{10 f_{\max}}. \tag{4.7}$$

Remarque 22. *En général, du point de vue de la stabilité, le choix de taille d'arête* (4.7) *dans la phase de maillage en tétraèdres est plus contraignant que la règle du « $\lambda_{\min}$ sur 10 ». En effet, les arêtes des hexaèdres issues du découpage (figure 4.3) et intérieures aux tétraèdres sont de taille inférieure à celle donnée par la formule* (4.6).

Ce sont aussi ces formules que nous utilisons pour appliquer la règle de l'ordre adaptatif suivante : si la plus petite arête d'une maille d'ordre d vérifie la règle du « $\lambda_{\min}$ sur 10 » donnée par la formule (4.6) avec un ordre d'interpolation d' inférieur à d, alors nous appliquons l'ordre d' à la maille.

Cette adaptation de l'ordre peut aussi être utilisée dans le but d'améliorer la précision dans d'éventuelles zones sous-maillées. Lorsqu'une maille est trop grande, il peut en résulter une perte d'information dans les hautes fréquences, proches de $f_{\max}$. Nous appliquons alors la règle suivante : tant que la plus grande arête d'une maille d'ordre d ne vérifie pas la règle du « $\lambda_{\min}$ sur 2 » (théorème de Shannon), nous incrémentons son ordre.

Toutefois, en présence d'une antenne dans le domaine simulé, il est fréquent que celle-ci soit modélisée par les plus petites mailles du maillage. Pour ne pas dénaturer la source du rayonnement, nous choisissons de ne pas réduire l'ordre d'interpolation à 0. La méthode GD d'ordre 0 discrétise la solution de façon constante par maille et induirait une perte de précision trop importante pour être utilisée à l'origine du rayonnement.

Des résultats de l'application de cette méthode sont présentés dans la section 7.2.3. Une accélération d'un facteur 3.8 y est constatée.

Conclusion

En conclusion, dans ce chapitre, nous avons présenté l'implémentation GPU que nous avons faite de la méthode GD sur hexaèdres décrite dans le chapitre 3. Le solveur `teta-clac` qui en est issu utilise la bibliothèque OpenCL couplée avec une implémentation du standard MPI pour proposer un outil de simulation efficace permettant de traiter des problèmes de grande taille. Ces qualités seront étudiées dans le chapitre 7.

Nous avons ensuite décrit les adaptations qui ont été menées dans le but d'optimiser les kernels de calcul OpenCL à destination des CPU. Les performances ont été nettement améliorées sur ce type de périphériques mais restent dépendantes des caractéristiques du périphérique en lui-même.

Enfin, nous avons présenté la méthode de l'ordre d'interpolation spatiale adaptatif qui permet de corriger automatiquement la discrétisation dans les zones sur- ou sous-maillées. Cette méthode facilite l'étape de maillage et donne plus de liberté à l'utilisateur. Elle permet aussi d'effectuer des simulations sur différentes bandes de fréquences tout en conservant le même maillage en évaluant l'ordre d'interpolation à appliquer à chaque maille.

Dans le chapitre suivant, nous allons introduire un schéma temporel permettant de découpler l'avancée en temps des mailles. Ce schéma va nous permettre d'appliquer un pas de temps local à chaque maille.

Chapitre 5

Schéma à pas de temps local

Dans le chapitre précédent, nous avons décrit l'implémentation de la version parallèle du solveur, pouvant utiliser simultanément un grand nombre d'accélérateurs. Cette capacité de gérer plusieurs accélérateurs nous a déjà permis d'accélérer considérablement la vitesse de calcul.

Afin d'améliorer encore la vitesse de calcul du solveur, sans avoir recours à de plus puissants et plus nombreux accélérateurs, nous allons présenter une adaptation algorithmique très efficace. Après avoir détaillé les schémas temporels de type Runge-Kutta dans la section 3.3, nous allons à présent décrire un schéma inspiré des méthodes de type ADER initialement proposée par Toro (voir par exemple [64, 65]).

Le schéma LRK2 (pour « *Local* RK2 ») que nous présentons dans cette section est une adaptation du schéma RK2, modifié à l'aide d'une phase de prédiction locale. Le schéma RK2 tel que présenté dans la partie 3.3.1 se décompose en 2 étapes faisant chacune intervenir l'opérateur $\mathcal{G}$ qui permet d'évaluer la dérivée à un instant t donné par la méthode Galerkin Discontinue (GD). Le schéma LRK2 reprend les mêmes étapes que le schéma RK2 à la différence près que l'opérateur $\mathcal{G}$ de la première étape est remplacé par une version locale de celui-ci.

Nous utilisons ensuite les propriétés locales du schéma LRK2 afin de découpler l'avancée en temps de l'approximation de la solution sur les différentes mailles. Ce second schéma, que nous appellerons LTS2 (pour « *Local Time Step* 2 »), permettra alors de réduire considérablement le temps de simulation en divisant la quantité de calculs lorsque la géométrie est composée de mailles de tailles très variées.

5.1 Approximation locale en dimension 1

Remarque 23. *Nous décrivons les modifications apportées à la méthode GD* (1.59) *en dimension* 1 *d'espace. Il s'agit de l'implémentation développée afin de tester la méthode sur un cas simple d'équation du transport. Ces modifications se généralisent facilement en dimension* 3.

Nous considérons le problème d'évolution (1.14) pour un système de Friedrichs (1.5). Afin de simplifier les écritures, nous considérons le problème sans source ($\boldsymbol{C} = 0$ et $\boldsymbol{S} = 0$) et nous supposons que la matrice $\boldsymbol{A}_t$ est égale à l'identité. Dans ces conditions, le problème d'évolution s'écrit :

$$\partial_t \boldsymbol{w} + \boldsymbol{A}\partial_x \boldsymbol{w} = 0. \tag{5.1}$$

5.1.1 Formulation GD en dimension 1

Nous considérons le maillage $\mathcal{M}$ du domaine $\Omega =]a\,;b[$ donné par l'ensemble des points $(x_i)_{i\in[\![0\,;N]\!]}$ tels que :

$$a = x_0 < \ldots < x_N = b. \tag{5.2}$$

Les mailles $(L_i)_{i\in[\![1\,;N]\!]}$ sont donc les intervalles :

$$L_i =]x_{i-1}\,;x_i[, \text{ pour } i \in [\![1\,;\,N]\!]. \tag{5.3}$$

Remarque 24. *Puisque nous décrivons cette méthode en dimension* 1*, dans la suite, nous notons* L *l'indice de la maille* L_i *considérée. Les indices des mailles voisines,* L_{i+j} *pour* $j \in \mathbb{Z}^\star$*, sont alors notés* $L + j$*.*

Nous utilisons alors l'élément de référence $\hat{L} =]0\,;1[$. Dans chaque maille, nous considérons une approximation spatiale d'ordre d, avec $d+1$ points d'approximation par maille. Les points $(\hat{g}_i)_{i\in[\![0\,;d]\!]}$ et poids $(\omega_i)_{i\in[\![0\,;d]\!]}$ d'interpolation choisis sont ceux de Gauss-Lobatto.

Remarque 25. *Nous aurions pu utiliser une interpolation à l'aide des points de Gauss-Legendre d'ordre* d*. L'utilisation des points de Gauss-Lobatto, par la présence des points de bord, simplifie l'implémentation en supprimant l'extrapolation des champs sur les interfaces entre les mailles.*

Les fonctions de base sur l'élément de référence $\hat{L}$ sont données par les polynômes de Lagrange d'ordre d définis par :

$$\hat{\psi}_i(\hat{x}) = \prod_{\substack{j=0\\ j\neq i}}^{d} \frac{\hat{x} - \hat{g}_j}{\hat{g}_i - \hat{g}_j}, \text{ pour } i \in [\![0\,;\,d]\!], \tag{5.4}$$

ainsi, par définition :

$$\hat{\psi}_i(\hat{g}_j) = \delta_{i,j}. \tag{5.5}$$

Nous notons τ_L la transformation permettant de passer de l'élément de référence $\hat{L}$ à la maille physique $L =]x_{L-1}\,; x_L[$. L'expression de celle-ci est donnée par :

$$\tau_L(\hat{x}) = x_{L-1} + \hat{x}(x_L - x_{L-1}). \tag{5.6}$$

Pour une maille L donnée (figure 5.1), les points d'approximation physiques sont notés :

$$g_{L,i} = \tau_L(\hat{g}_i), \text{ pour } i \in [\![0\,; d]\!], \tag{5.7}$$

et les fonctions de base physiques sont données par :

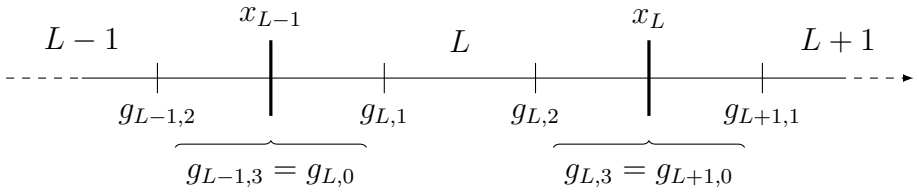
$$\psi_{L,i} \circ \tau_L(\hat{x}) = \hat{\psi}_i(\hat{x}) \Leftrightarrow \psi_{L,i}(x) = \hat{\psi}_i \circ \tau_L^{-1}(x). \tag{5.8}$$

Ainsi, dans chaque maille L, nous approchons $\boldsymbol{w}(x,t)$ par :

$$\boldsymbol{w}(x,t) \approx \sum_{i=0}^{d} \boldsymbol{w}_{L,i}(t)\psi_{L,i}(x) = \boldsymbol{w}_{L,i}(t)\psi_{L,i}(x), \tag{5.9}$$

en utilisant la convention de sommation des indices répétés.

FIGURE 5.1 – Mailles et points d'approximations du maillage considéré pour l'ordre d'approximation 3.



Avec ces notations, nous avons la formulation GD suivante : trouver une solution $\boldsymbol{w}$ telle que :

$$\begin{aligned}
&\forall L =]x_{L-1}\,; x_L[\in \mathcal{M}, \forall \psi_{L,i} \in \mathcal{C}^1(\overline{L}),\\
&\int_L \partial_t \boldsymbol{w}\psi_{L,i}dx - \int_L \boldsymbol{A}\boldsymbol{w}\partial_x\psi_{L,i}dx\\
&\qquad + (\boldsymbol{A}^+\boldsymbol{w}(x_L^-,t) + \boldsymbol{A}^-\boldsymbol{w}(x_L^+,t))\psi_{L,i}(x_L^-)\\
&\qquad - (\boldsymbol{A}^-\boldsymbol{w}(x_{L-1}^+,t) + \boldsymbol{A}^+\boldsymbol{w}(x_{L-1}^-,t))\psi_{L,i}(x_{L-1}^+) = 0,
\end{aligned} \tag{5.10}$$

où $\boldsymbol{A}^+$, respectivement $\boldsymbol{A}^-$, représente la partie positive, respectivement négative, de $\boldsymbol{A}$ (définition 6) et $\boldsymbol{w}(x_L^+, t)$, respectivement $\boldsymbol{w}(x_L^-, t)$, représente la valeur de $\boldsymbol{w}$ à l'instant t en x_L à droite, respectivement à gauche. En d'autres termes :

$$\boldsymbol{w}(x_L^+, t) = \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \boldsymbol{w}(x_L + \varepsilon, t) = \boldsymbol{w}_{L+1,0}(t), \tag{5.11a}$$

$$\boldsymbol{w}(x_L^-, t) = \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \boldsymbol{w}(x_L - \varepsilon, t) = \boldsymbol{w}_{L,d}(t). \tag{5.11b}$$

Et de même pour l'évaluation de la fonction $\psi_{L,i}$:

$$\psi_{L,i}(x_{L-1}^+) = \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \psi_{L,i}(x_{L-1} + \varepsilon) = \psi_{L,i}(g_{L,0}) = \delta_{i,0}, \tag{5.12a}$$

$$\psi_{L,i}(x_L^-) = \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \psi_{L,i}(x_L - \varepsilon) = \psi_{L,i}(g_{L,d}) = \delta_{i,d}. \tag{5.12b}$$

En utilisant l'approximation de $\boldsymbol{w}$ dans la base d'approximation (5.9), nous obtenons :

$$\begin{aligned} \int_L \partial_t \boldsymbol{w}_{L,j} \psi_{L,j} \psi_{L,i} dx - \int_L \boldsymbol{A} \boldsymbol{w}_{L,j} \psi_{L,j} \partial_x \psi_{L,i} dx \\ + (\boldsymbol{A}^+ \boldsymbol{w}_{L,d} + \boldsymbol{A}^- \boldsymbol{w}_{L+1,0}) \delta_{i,d} \\ - (\boldsymbol{A}^- \boldsymbol{w}_{L,0} + \boldsymbol{A}^+ \boldsymbol{w}_{L-1,d}) \delta_{i,0} = 0. \end{aligned} \tag{5.13}$$

Ainsi, en supposant connue la solution à l'instant $t = \alpha$, nous pouvons évaluer la solution à l'instant $t = \beta > \alpha$ en intégrant (5.13) sur $[\alpha\,;\beta]$:

$$\begin{aligned} \left(\int_L \psi_{L,j} \psi_{L,i} dx \right) (\boldsymbol{w}_{L,j}(\beta) - \boldsymbol{w}_{L,j}(\alpha)) \\ - \int_{t=\alpha}^{\beta} \int_L \boldsymbol{A} \boldsymbol{w}_{L,j} \psi_{L,j} \partial_x \psi_{L,i} dx dt \\ + \int_{t=\alpha}^{\beta} (\boldsymbol{A}^+ \boldsymbol{w}_{L,d} + \boldsymbol{A}^- \boldsymbol{w}_{L+1,0}) \delta_{i,d} dt \\ - \int_{t=\alpha}^{\beta} (\boldsymbol{A}^- \boldsymbol{w}_{L,0} + \boldsymbol{A}^+ \boldsymbol{w}_{L-1,d}) \delta_{i,0} dt = 0. \end{aligned} \tag{5.14}$$

En appliquant les simplifications vues dans la section 3.2.1, le terme de masse devient :

$$\int_L \psi_{L,j} \psi_{L,i} dx = \omega_i (x_L - x_{L-1}). \tag{5.15}$$

Nous allons nous placer dans un cadre un peu plus général. Nous pouvons aussi considérer un système d'équations différentielles ordinaires défini par un graphe $G = (\mathcal{M}, E)$, où l'ensemble des nœuds du graphe est noté $\mathcal{M}$ et $E \subset \mathcal{M}^2$ est l'ensemble des arêtes orientées du graphe. Dans notre cas $\mathcal{M}$ est l'ensemble des mailles L du maillage et E s'identifie à l'ensemble des interfaces entre deux mailles voisines. Avec ces notations, l'équation différentielle (5.13) s'écrit pour chaque nœud L du graphe :

$$\partial_t \boldsymbol{w}_L = \sum_{(L,R)\in E} \boldsymbol{\mu}_{L,R}(\boldsymbol{w}_L, \boldsymbol{w}_R), \tag{5.16}$$

où les $\boldsymbol{\mu}_{L,R}$ sont des matrices de couplage entre les inconnues des mailles L et R. L'expression de la solution au temps $t = \beta$ en fonction de la solution connue au temps $t = \alpha$ devient alors :

$$\boldsymbol{w}_L(\beta) - \boldsymbol{w}_L(\alpha) = \int_{t=\alpha}^{\beta} \sum_{(L,R)\in E} \boldsymbol{\mu}_{L,R}(\boldsymbol{w}_L, \boldsymbol{w}_R). \tag{5.17}$$

Remarquons que le système composé des équations (5.16) permet de déterminer la dérivée en temps de la solution à un instant donné pour lequel la solution est connue. Cette opération correspond à l'application de l'opérateur $\mathcal{G}$ (3.42) en dimension 1.

5.1.2 Formulation GD locale en dimension 1

Afin de supprimer le couplage entre les mailles, nous approchons l'équation différentielle (5.13) par un « prédicteur » local à la maille :

$$\begin{aligned}\int_L \partial_t \boldsymbol{v}_{L,j}\psi_{L,j}\psi_{L,i}dx - \int_L \boldsymbol{A}\boldsymbol{v}_{L,j}\psi_{L,j}\partial_x\psi_{L,i}dx \\ + (\boldsymbol{A}^+\boldsymbol{v}_{L,d} + \boldsymbol{A}^-\boldsymbol{v}_{L,d})\delta_{i,d} \\ - (\boldsymbol{A}^-\boldsymbol{v}_{L,0} + \boldsymbol{A}^+\boldsymbol{v}_{L,0})\delta_{i,0} = 0.\end{aligned} \tag{5.18}$$

Puis, en simplifiant les termes de bord :

$$\begin{aligned}\int_L \partial_t \boldsymbol{v}_{L,j}\psi_{L,j}\psi_{L,i}dx - \int_L \boldsymbol{A}\boldsymbol{v}_{L,j}\psi_{L,j}\partial_x\psi_{L,i}dx \\ + \boldsymbol{A}(\boldsymbol{v}_{L,d}\delta_{i,d} - \boldsymbol{v}_{L,0}\delta_{i,0}) = 0.\end{aligned} \tag{5.19}$$

Nous obtenons ainsi un autre système d'équations différentielles dont la solution est une approximation locale du problème (5.13) ou (5.16), $\boldsymbol{v}_L \approx \boldsymbol{w}_L$.

Dans notre formalisme d'équations différentielles posées sur les nœuds d'un graphe, (5.16) est donc approché par un système diagonal, sans interaction entre les nœuds voisins. Ce système diagonal est de la forme :

$$\partial_t \boldsymbol{v}_L = \boldsymbol{\nu}_L(\boldsymbol{v}_L), \tag{5.20}$$

où les termes de couplage $\boldsymbol{\mu}_{L,R}$ sont remplacés par une approximation diagonale $\boldsymbol{\nu}_L$. Nous notons $\mathcal{G}_l$ l'opérateur associé à l'évaluation de la dérivée en temps à l'aide du système composé des équations (5.20).

Nous sommes alors en mesure de calculer une prédiction locale de la dérivée en temps à partir de la solution de bonne qualité $\boldsymbol{w}$, à l'aide du système (5.20) :

$$\partial_t \boldsymbol{v} = \mathcal{G}_l(\boldsymbol{w}, t). \tag{5.21}$$

Remarque 26. *L'application de l'opérateur $\mathcal{G}_l$ correspond à l'application de l'opérateur $\mathcal{G}$ (3.42) en remplaçant le flux d'interface $F(\boldsymbol{w}_L, \boldsymbol{w}_R, \boldsymbol{n})$ par le flux local $F(\boldsymbol{w}_L, \boldsymbol{w}_L, \boldsymbol{n})$.*

Remarque 27. *Ce prédicteur ne prend pas en compte la valeur de la solution des mailles voisines. L'évaluation de la dérivée en temps ne doit donc pas être calculée uniquement à l'aide de cet opérateur. Nous l'utilisons, comme son nom l'indique, dans la phase de prédiction, une mise à jour étant toujours calculée à l'aide du système d'approximation GD (5.16).*
En appliquant la prédiction locale (5.20), nous admettons que les variables conservatives à gauche et à droite des interfaces entre les mailles sont identiques. Nous négligeons cet écart afin de réduire les coûts de calcul et, dans la suite, découpler l'avancée en temps des mailles.

Supposons calculée la solution $\boldsymbol{w}$ au temps $t = \alpha$. Pour calculer la solution au temps $t = \beta$, nous utilisons alors la mise à jour :

$$\boldsymbol{w}_L(\beta) - \boldsymbol{w}_L(\alpha) = \int_{t=\alpha}^{\beta} \sum_{(L,R)\in E} \boldsymbol{\mu}_{L,R}(\boldsymbol{v}_L, \boldsymbol{v}_R), \tag{5.22}$$

où $\boldsymbol{v}$ est calculé grâce au équations différentielles locales (5.20). Cette intégrale en temps peut être approchée par n'importe quelle formule de quadrature, par exemple la méthode du point milieu :

$$\begin{aligned} &\boldsymbol{w}_L(\beta) - \boldsymbol{w}_L(\alpha) \approx \\ &\qquad (\beta - \alpha) \sum_{(L,R)\in E} \boldsymbol{\mu}_{L,R} \left(\boldsymbol{v}_L\left(\frac{\alpha+\beta}{2}\right), \boldsymbol{v}_R\left(\frac{\alpha+\beta}{2}\right)\right). \end{aligned} \tag{5.23}$$

Il s'agit là du schéma LRK2 que nous présentons ci-après (section 5.2).

De manière plus générale, nous pouvons choisir une intégration temporelle propre à chaque maille, à l'aide de points de Gauss temporels. Afin d'avancer la solution en temps sur une maille donnée, une prédiction doit être calculée en chaque point temporel, sur la maille considérée ainsi que sur ses voisins. La mise à jour s'effectue ensuite en appliquant la formule de quadrature temporelle (5.22) avec ces prédictions.

5.2 Schéma LRK2

L'avancée en temps dans l'évaluation de la solution peut se faire sur le modèle du schéma RK2 (3.43), en substituant la première étape par une prédiction locale (5.21) :

$$\begin{aligned} \boldsymbol{v}^{n+\frac{1}{2}} &= \boldsymbol{w}^n + \frac{\Delta t}{2}\mathcal{G}_l(\boldsymbol{w}^n, t), \\ \boldsymbol{w}^{n+1} &= \boldsymbol{w}^n + \Delta t \mathcal{G}\left(\boldsymbol{v}^{n+\frac{1}{2}}, t + \frac{\Delta t}{2}\right). \end{aligned} \tag{5.24}$$

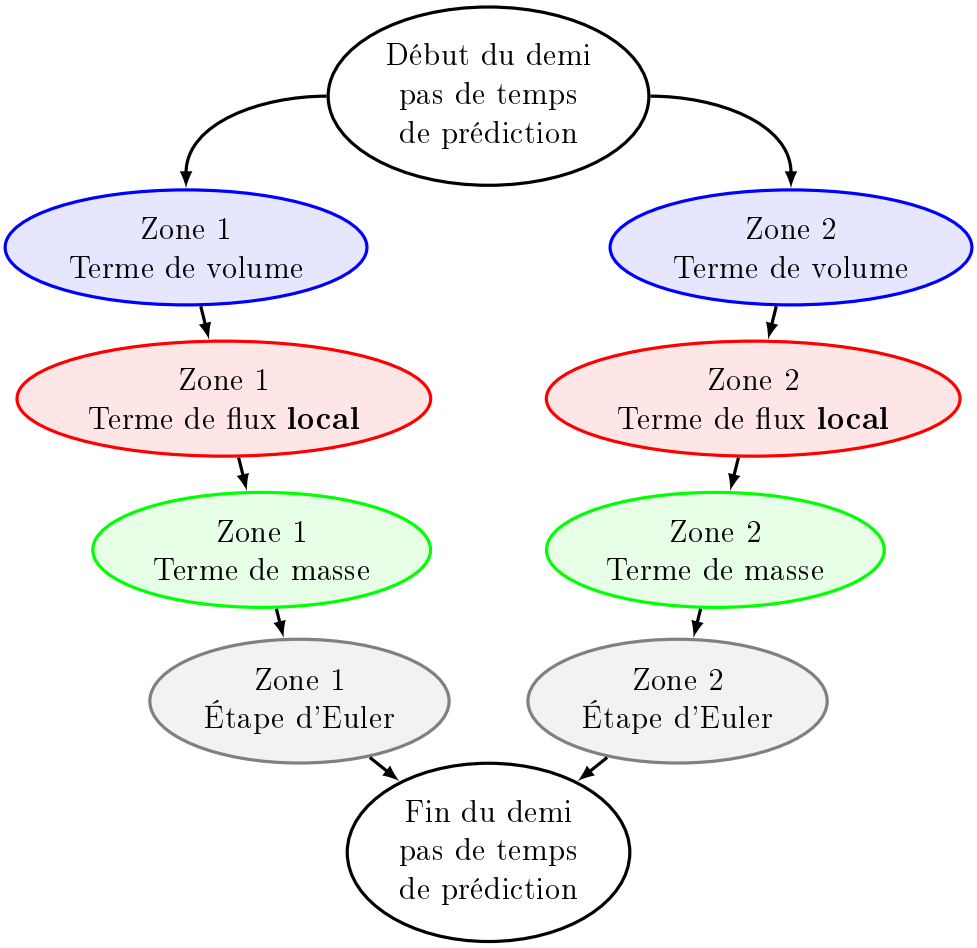
Il s'agit de la formule d'intégration temporelle (5.23) appliquée de manière uniforme sur tout le maillage. Là aussi, la première étape est appelée prédiction et la seconde étape est appelée mise à jour. Le graphe des tâches de l'étape de prédiction locale du schéma couplé GD-LRK2 est donné dans la figure 5.2. L'étape de mise à jour est identique à celle du schéma couplé GD-RK2 dont le graphe des tâches est présenté dans la figure 4.4.

La stabilité et la convergence de cette méthode sont expérimentalement démontrées équivalentes au schéma RK2 dans les sections 5.4 (équation du transport 1-d) et 7.2.4 (équations de Maxwell 3-d). De plus, un facteur d'accélération non négligeable a été constaté suite à la simplification des flux d'interfaces entre les mailles. Nous présentons ces résultats dans la section 7.2.4.

5.3 Schéma LTS2

Les critères de stabilité de type CFL donnent des pas de temps proportionnels à la taille des mailles. Or, dans le cas des schémas que nous avons présenté précédemment, l'avancée en temps de la solution s'effectue au rythme du plus petit pas de temps. Ainsi, dans le cas de maillages contenant des mailles de tailles très variées, une méthode permettant de faire avancer la

FIGURE 5.2 – Graphe des tâches du demi pas de temps de prédiction locale du schéma couplé GD-LRK2 pour deux zones homogènes. L'étape de mise à jour est donnée dans la figure 4.4.



solution sur chaque maille au rythme du pas de temps propre à cette maille nous donnerait un important facteur d'accélération en réduisant la quantité de calculs nécessaires.

C'est le cas du schéma que nous présentons ici. Nous utilisons les propriétés locales du prédicteur $\mathcal{G}_l$ afin de découpler l'avancée en temps de la solution sur les mailles. Rappelons que cet opérateur permet d'estimer la valeur de la solution à un instant donné, indépendamment des mailles voisines.

5.3.1 Description

Remarque 28. *Nous nous replaçons dans le cas en dimension* 1 *pour les explications schématiques de cette méthode.*

Considérons les mailles L, $L+1$, $L+2$ et Δt_L, Δt_{L+1}, Δt_{L+2} leurs pas de temps respectifs donnés par l'application d'un critère CFL de stabilité. Supposons ces mailles définies telles que :

$$\Delta t_{L+2} \geq \Delta t_{L+1} = 2\Delta t_L. \tag{5.25}$$

Avec un schéma temporel à pas de temps uniforme tel que les schémas de type RK, la solution sur toutes ces mailles avance d'un pas de temps Δt_L à chaque itération. En effet, ce pas de temps est imposé par la maille L qui nécessite l'avancée synchronisée de la maille $L+1$ afin de procéder de manière consistante et conservative au calcul du flux entre les mailles (figure 5.3). Ainsi, de proche en proche, la solution évolue sur le maillage complet avec le plus petit pas de temps à chaque itération.

Dans les figures 5.3, 5.4 et 5.5, les lignes pointillées rouges représentent le domaine de définition de la solution aux différentes étapes des schémas d'intégration temporelle présentés. La solution peut être définie à chaque étape sur le maillage complet (figures 5.3 et 5.4) ou partiellement sur quelques mailles (figure 5.5). Afin d'avancer en temps (passer d'une ligne pointillée rouge à la suivante), nous appliquons l'opérateur $\mathcal{G}$ ou l'opérateur $\mathcal{G}_l$ suivi d'une étape de type Euler (section 4.2.2). Seul le terme de flux (local ou non) des opérateurs est représenté dans ces figures à l'aide de flèches.

Remarque 29. *Dans la suite, nous utiliserons l'abus de notation qui consiste à identifier une maille à sa solution. Ainsi, lorsque nous dirons qu'une maille avance d'un pas de temps* Δt*, cela signifie que la solution définie sur cette maille est avancée d'un pas de temps* Δt *par une méthode d'intégration, ou de manière plus générale, par un schéma temporel.*

Dans le cas du schéma LRK2 (5.24), l'avancée en temps doit aussi s'effectuer de manière synchronisée (figure 5.4) au rythme du plus petit pas de temps.

Ces deux schémas implémentent une quadrature temporelle par la méthode du point milieu, de manière uniforme sur le maillage. À partir de la solution au temps t, une prédiction au temps $t + \frac{1}{2}\Delta t_L$ — par l'opérateur $\mathcal{G}$ dans le cas RK2 et par l'opérateur $\mathcal{G}_l$ dans le cas LRK2 — permet ensuite d'intégrer les champs sur l'intervalle $[t\,;t+\Delta t_L]$ afin d'obtenir la solution au temps $t + \Delta t_L$.

FIGURE 5.3 – Calcul d'un pas de temps RK2 dans une configuration de mailles hétérogènes. Les étapes dans le calcul de la solution sont en pointillés rouges. Le calcul des flux dans l'opérateur $\mathcal{G}$ est représenté par les cercles bleus et leur application via l'avancée en temps par les flèches bleues.

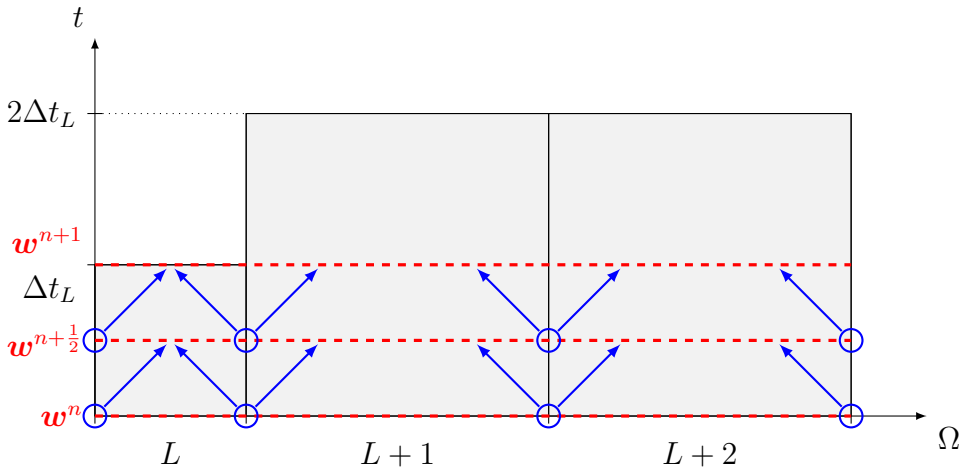
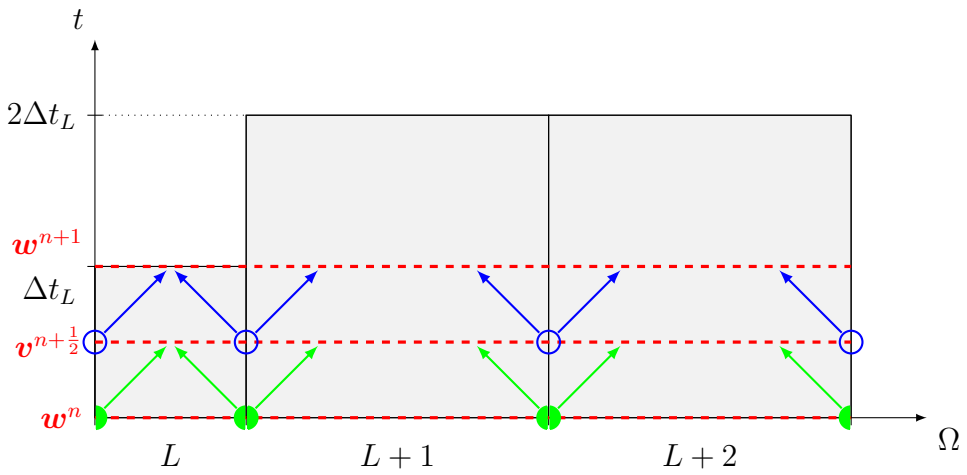


FIGURE 5.4 – Calcul d'un pas de temps LRK2 dans une configuration de mailles hétérogènes. Les étapes dans le calcul de la solution sont en pointillés rouges. Le calcul des flux dans l'opérateur $\mathcal{G}$ est représenté par les cercles bleus et leur application via l'avancée en temps par les flèches bleues. Le calcul des flux locaux dans l'opérateur $\mathcal{G}_l$ est représenté par les demi-disques verts et leur application via l'avancée en temps par les flèches vertes.



Nous adaptons alors ce dernier schéma en utilisant les propriétés locales du prédicteur $\mathcal{G}_l$ afin d'avancer en temps indépendamment sur chaque maille.

Toujours dans la configuration des 3 mailles L, $L+1$ et $L+2$, nous appliquons le pas de temps Δt_L à la maille L et le pas de temps $2\Delta t_L$ à la maille $L+2$. Ainsi, sur un pas de temps $2\Delta t_L$, nous avons 2 points d'intégration temporels à gauche de la maille $L+1$ et un seul à droite. La maille $L+1$ est alors appelée maille « tampon ». Celle-ci est stable pour le pas de temps $2\Delta t_L$, mais nous la faisons avancer en temps de manière hybride afin de satisfaire la consistance et la conservation du flux avec ses voisins.

Tout d'abord, sur les mailles L et $L+1$, nous commençons par effectuer une première prédiction au temps $t+\frac{1}{2}\Delta t_L$ afin de nous permettre d'avancer la maille L au temps $t+\Delta t_L$ par une mise à jour (figure 5.5a). Il s'agit là d'une itération de type LRK2 de pas de temps Δt_L sur la maille L.

Ensuite, toujours sur les mailles L et $L+1$, nous effectuons une seconde prédiction au temps $t+\frac{3}{2}\Delta t_L$ afin de nous permettre d'avancer la maille L au temps $t+2\Delta t_L$ par une seconde mise à jour (figure 5.5b). Il s'agit là d'une seconde itération de type LRK2 de pas de temps Δt_L sur la maille L.

Enfin, nous effectuons une itération de type LRK2 de pas de temps $2\Delta t_L$ sur les mailles $L+1$ et $L+2$ afin de les avancer au temps $t+2\Delta t_L$ (figure 5.5c). La subtilité réside au niveau de l'interface entre les mailles L et $L+1$. En effet, pour assurer la propriété de conservation du flux, nous devons intégrer les 2 flux issus de la maille L lorsque nous intégrons — en une seule étape — la solution sur la maille $L+1$. Informatiquement, nous stockons les flux calculés à droite de cette interface au cours des mises-à-jour de la maille L (étapes 5.5a et 5.5b). Ces flux sont ensuite utilisés dans la mise à jour de la maille $L+1$ (étape 5.5c).

Cette méthode permet de découpler de manière conservative l'avancée en temps des mailles associées à un pas de temps distinct.

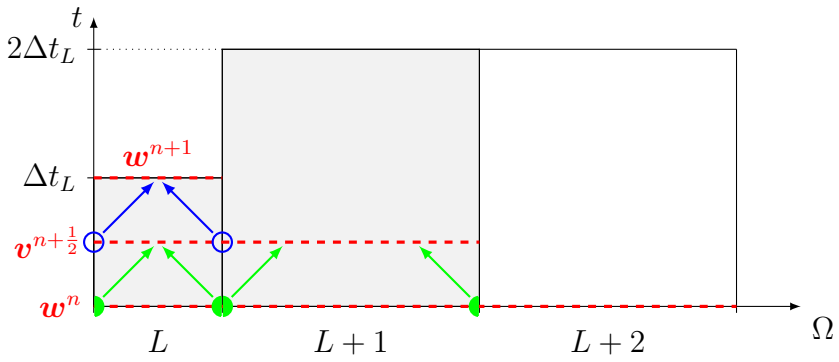
5.3.2 Classes de pas de temps

Remarque 30. *Nous nous replaçons dans le cas général en dimension quelconque pour la description des classes de pas de temps et de l'algorithme.*

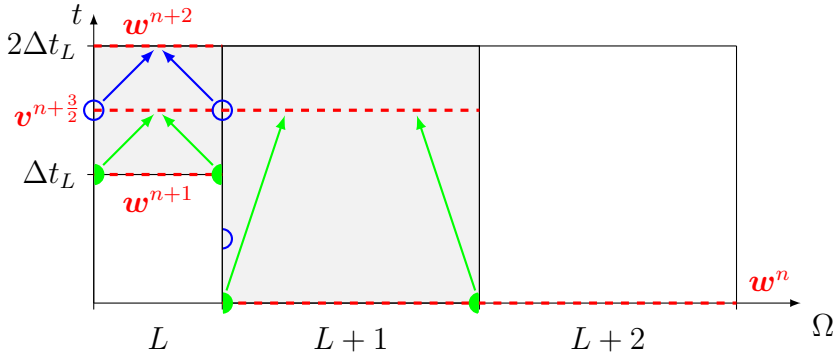
Afin de faciliter l'implémentation de cette méthode, nous utilisons des pas de temps de la forme $2^i\Delta t_{\min}$, $i \in \mathbb{N}$, avec $\Delta t_{\min}$ le pas de temps minimal du maillage. Nous associons à chaque maille L, de pas de temps exact Δt_L,

FIGURE 5.5 – Calcul d'un pas de temps complet dans une configuration de mailles hétérogènes avec application d'un pas de temps local par maille. Nous reprenons la légende de la figure 5.4.

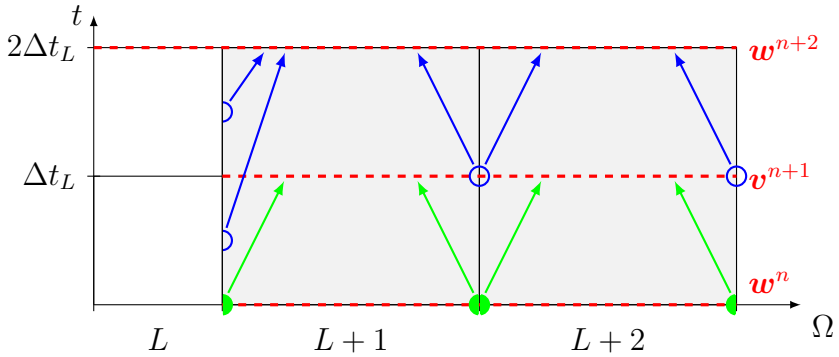
(a) L avance au temps $t + \Delta t_L$.



(b) L avance au temps $t + 2\Delta t_L$.



(c) $L + 1$ et $L + 2$ avancent au temps $t + 2\Delta t_L$.



le pas de temps local :

$$\Delta t_L^l = 2^{\mathrm{E}\left(\log_2 \frac{\Delta t_L}{\Delta t_{\min}}\right)} \Delta t_{\min} \leq \Delta t_L, \tag{5.26}$$

où E est la fonction partie entière. Le pas de temps local maximal est donné par :

$$\Delta t_{\max}^l = 2^{\mathrm{E}\left(\log_2 \frac{\Delta t_{\max}}{\Delta t_{\min}}\right)} \Delta t_{\min} \leq \Delta t_{\max}, \tag{5.27}$$

où $\Delta t_{\max}$ est le pas de temps maximal du maillage. Nous notons N_t le nombre entier défini tel que :

$$\Delta t_{\max}^l = 2^{N_t} \Delta t_{\min} \tag{5.28}$$

et qui correspond au nombre de pas de temps locaux distincts, différents de $\Delta t_{\min}$, définis sur le maillage. Si $N_t = 0$, alors le pas de temps est uniforme sur le maillage et vaut $\Delta t_{\min}$.

Ce choix de pas de temps nous assure, sous contraintes, la possibilité d'appliquer la méthode décrite par la figure 5.5 à tout le maillage. Ces contraintes s'appliquent à la régularité des pas de temps locaux et nous assurent la présence des mailles tampon :

$$\begin{aligned} \forall L, L', L'' \in \mathcal{M} : \overline{L} \cap \overline{L'} \neq \emptyset, \overline{L'} \cap \overline{L''} \neq \emptyset, \\ \frac{\max(\Delta t_L^l, \Delta t_{L'}^l, \Delta t_{L''}^l)}{\min(\Delta t_L^l, \Delta t_{L'}^l, \Delta t_{L''}^l)} \leq 2. \end{aligned} \tag{5.29}$$

Afin de les appliquer, il nous suffit d'abaisser le pas de temps local de certaines mailles à l'aide de divisions par 2.

L'application qui associe à une maille son pas de temps local est continue par morceaux. Nous pouvons alors regrouper les mailles en classes de pas de temps :

$$\forall i \in [\![0\,;\, N_t]\!], \mathcal{C}_i = \left\{ L \in \mathcal{M} : \Delta t_L^l = 2^i \Delta t_{\min} \right\}. \tag{5.30}$$

Nous avons :

$$\mathcal{M} = \bigcup_{i \in [\![0\,;\, N_t]\!]} \mathcal{C}_i, \text{ avec } \forall i \neq j \in [\![0\,;\, N_t]\!],\ \mathcal{C}_i \cap \mathcal{C}_j = \emptyset \tag{5.31}$$

et, par l'application des contraintes (5.29), les mailles voisines sont classées telles que :

$$\forall L \in \mathcal{C}_i, \forall R \in \mathcal{C}_j : \overline{L} \cap \overline{R} \neq \emptyset,\ i = j \text{ ou } |i - j| = 1. \tag{5.32}$$

Ces contraintes sont même plus fortes puisqu'elles imposent localement dans chaque classe de pas de temps une épaisseur minimale de 2 mailles.

Nous pouvons aussi définir les ensembles, pour $i \in [\![0\,;\,N_t]\!]$:

$$\mathcal{K}_i = \begin{cases} \mathcal{C}_i & \text{si } i = N_t, \\ \mathcal{C}_i \cup \left\{ L \in \mathcal{C}_{i+1} : \exists R \in \mathcal{C}_i : \overline{L} \cap \overline{R} \neq \emptyset \right\} & \text{sinon,} \end{cases} \tag{5.33}$$

$$\mathcal{T}_i = \mathcal{K}_i \setminus \mathcal{C}_i \tag{5.34}$$

et

$$\mathcal{I}_i = \begin{cases} \emptyset & \text{si } i = N_t, \\ \left\{ \overline{L} \cap \overline{R} \neq \emptyset : L \in \mathcal{C}_i, R \in \mathcal{C}_{i+1} \right\} & \text{sinon.} \end{cases} \tag{5.35}$$

Les $(\mathcal{K}_i)$ sont les classes de pas de temps, chacune enrichie des mailles tampon de la classe supérieure. Les $(\mathcal{T}_i)$ sont les mailles tampon de la classe supérieure. Les $(\mathcal{I}_i)$ sont les interfaces entre les classes $(\mathcal{C}_i)$ et les mailles tampon de la classe supérieure. Ces ensembles sont représentés en dimension 2 dans la figure 5.6.

5.3.3 Algorithme

Avec ces notations, nous pouvons écrire le schéma en temps correspondant à la figure 5.5. Ce schéma est composé d'une boucle principale permettant d'intégrer toutes les mailles sur un pas de temps complet $\Delta t^l_{\max}$. Cette boucle est composée de 2^{N_t} étapes. Les mailles de la classe $\mathcal{C}_0$ avancent d'un pas de temps $\Delta t_{\min}$ à chaque étape. Les mailles de la classe $\mathcal{C}_{N_t}$ n'avancent qu'une seule fois d'un pas de temps $\Delta t^l_{\max}$ à la dernière étape. Plus généralement, les mailles de la classe $\mathcal{C}_j$ avancent $2^{N_t - j}$ fois d'un pas de temps $2^j \Delta t_{\min}$.

Contrairement au schéma de la figure 5.5, nous indiçons les itérations de la solution par rapport au pas de temps maximal afin que les termes de la suite résultante soient totalement définis sur le maillage complet.

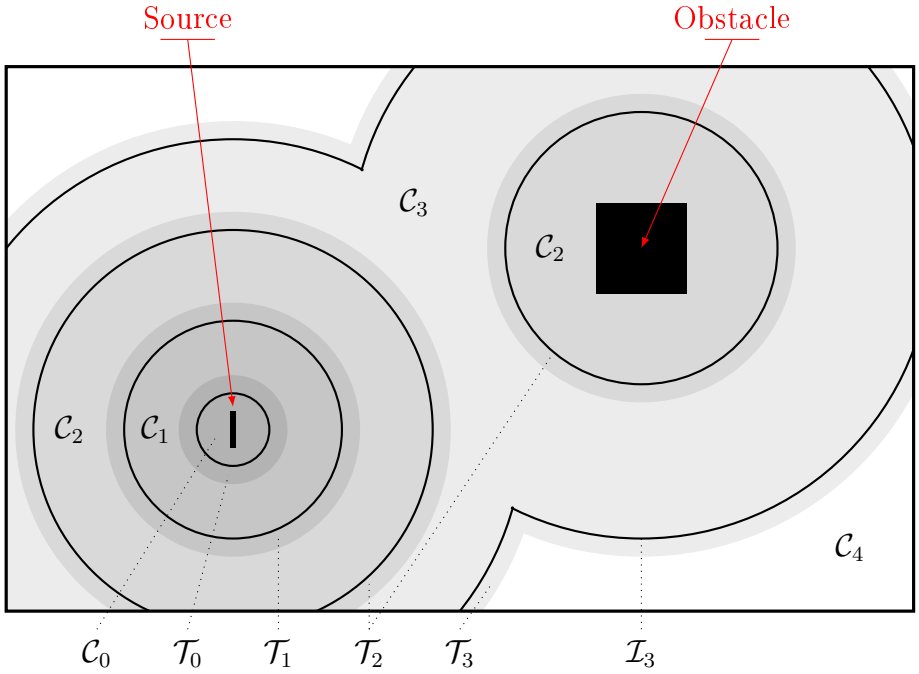
Ainsi, considérons connue la solution du problème à résoudre au temps $t = n\Delta t^l_{\max}$. À chaque étape $i \in [\![1\,;\,2^{N_t}]\!]$, pour tous les $j \in [\![0\,;\,N_t]\!]$ tels que 2^j divise i, avec k l'indice de l'étape courante de la classe $\mathcal{C}_j$:

$$k \in \mathbb{N} : i = 2^j(k+1), \tag{5.36}$$

r l'indicateur de décalage temporel de la dernière solution calculée entre la classe $\mathcal{C}_j$ et les mailles tampon $\mathcal{T}_j$:

$$r \in \{0, 1\} : k \equiv r\ [2], \tag{5.37}$$

FIGURE 5.6 – Schéma représentant un agencement classique des classes de pas de temps dans le domaine d'étude. Plan de coupe d'un domaine en dimension 3 contenant une source rayonnante et un obstacle éclairé. La classe $\mathcal{C}_0$ est généralement définie au niveau de la source du rayonnement électromagnétique où la miniaturisation et les détails de la structure imposent la finesse du maillage. La classe $\mathcal{C}_{N_t}$ est généralement définie sur les bords du domaine où aucune contrainte autre que la permittivité du milieu ne s'applique sur les mailles.



Δt_j^l le pas de temps local associé à la classe $\mathcal{C}_j$:

$$\Delta t_j^l = 2^j \Delta t_{\min} = 2^{j-N_t} \Delta t_{\max}^l = \alpha \Delta t_{\max}^l, \tag{5.38}$$

s l'indice temporel de la dernière solution intermédiaire calculée :

$$s = \begin{cases} n + \alpha k & \text{sur } \mathcal{C}_j, \\ n + \alpha(k - r) & \text{sur } \mathcal{T}_j, \end{cases} \tag{5.39}$$

et p l'indice temporel de la prédiction servant à calculer la solution intermédiaire suivante :

$$p = n + \alpha \left(k + \frac{1}{2} \right) = \begin{cases} s + \frac{\alpha}{2} & \text{sur } \mathcal{C}_j, \\ s + \alpha \left(\frac{1}{2} + r \right) & \text{sur } \mathcal{T}_j, \end{cases} \tag{5.40}$$

nous avons :

$$\begin{aligned} \boldsymbol{v}^p &= \boldsymbol{w}^s + (p-s)\Delta t^l_{\max}\mathcal{G}_l(\boldsymbol{w}^s, s\Delta t^l_{\max}), \text{ sur } \mathcal{C}_j \cup \mathcal{T}_j, \\ \boldsymbol{w}^{s+\alpha} &= \boldsymbol{w}^s + \alpha\Delta t^l_{\max}\mathcal{G}(\boldsymbol{v}^p, p\Delta t^l_{\max}), \text{ sur } \mathcal{C}_j, \end{aligned} \tag{5.41}$$

avec l'opérateur $\mathcal{G}$ intégrant les flux suivants aux interfaces de la classe $\mathcal{C}_j$:

$$\begin{cases} F\left(\boldsymbol{v}_L^p, \boldsymbol{v}_R^p, \boldsymbol{n}\right) & \text{sur } \mathcal{I}_j^-, \\ \frac{1}{2}\left(F\left(\boldsymbol{v}_L^{p-\frac{\alpha}{4}}, \boldsymbol{v}_R^{p-\frac{\alpha}{4}}, \boldsymbol{n}\right) + F\left(\boldsymbol{v}_L^{p+\frac{\alpha}{4}}, \boldsymbol{v}_R^{p+\frac{\alpha}{4}}, \boldsymbol{n}\right)\right) & \text{sur } \mathcal{I}_{j-1}^+, j > 0, \end{cases} \tag{5.42}$$

où $\mathcal{I}_j^+$, respectivement $\mathcal{I}_j^-$, représente le côté $\mathcal{T}_j$, respectivement $\mathcal{C}_j$, de l'interface $\mathcal{I}_j$ dans le cadre du stockage et de l'application des flux entre deux classes de pas de temps.

Après ces 2^{N_t} étapes, toutes les mailles du maillage sont avancées au temps $n\Delta t^l_{\max} + 2^{N_t}\Delta t_{\min} = (n+1)\Delta t^l_{\max}$. Le graphe des tâches du schéma couplé GD-LTS2 est donné dans la figure 5.7. Ce graphe représente le calcul d'un pas de temps $\Delta t^l_{\max}$ sur 2 classes de pas de temps.

5.4 Application à l'équation du transport

5.4.1 Implémentation en dimension 1

Nous appliquons les schémas d'intégration LRK2 (5.24) et LTS2 (5.41) dans le cas de l'équation de transport en dimension 1. Il s'agit du premier cas d'application qui a permis de vérifier la validité et la stabilité numérique de ces méthodes. Ainsi, $m = 1$, $\Omega = [a\,;b] \subset \mathbb{R}$, $t \in [0\,;T] \subset \mathbb{R}$, $\boldsymbol{w} = w(x,t)$ et $\boldsymbol{A} = u \in \mathbb{R}$ la vitesse de propagation. Dans ces conditions, le problème d'évolution s'écrit :

$$\partial_t w + u\partial_x w = 0. \tag{5.43}$$

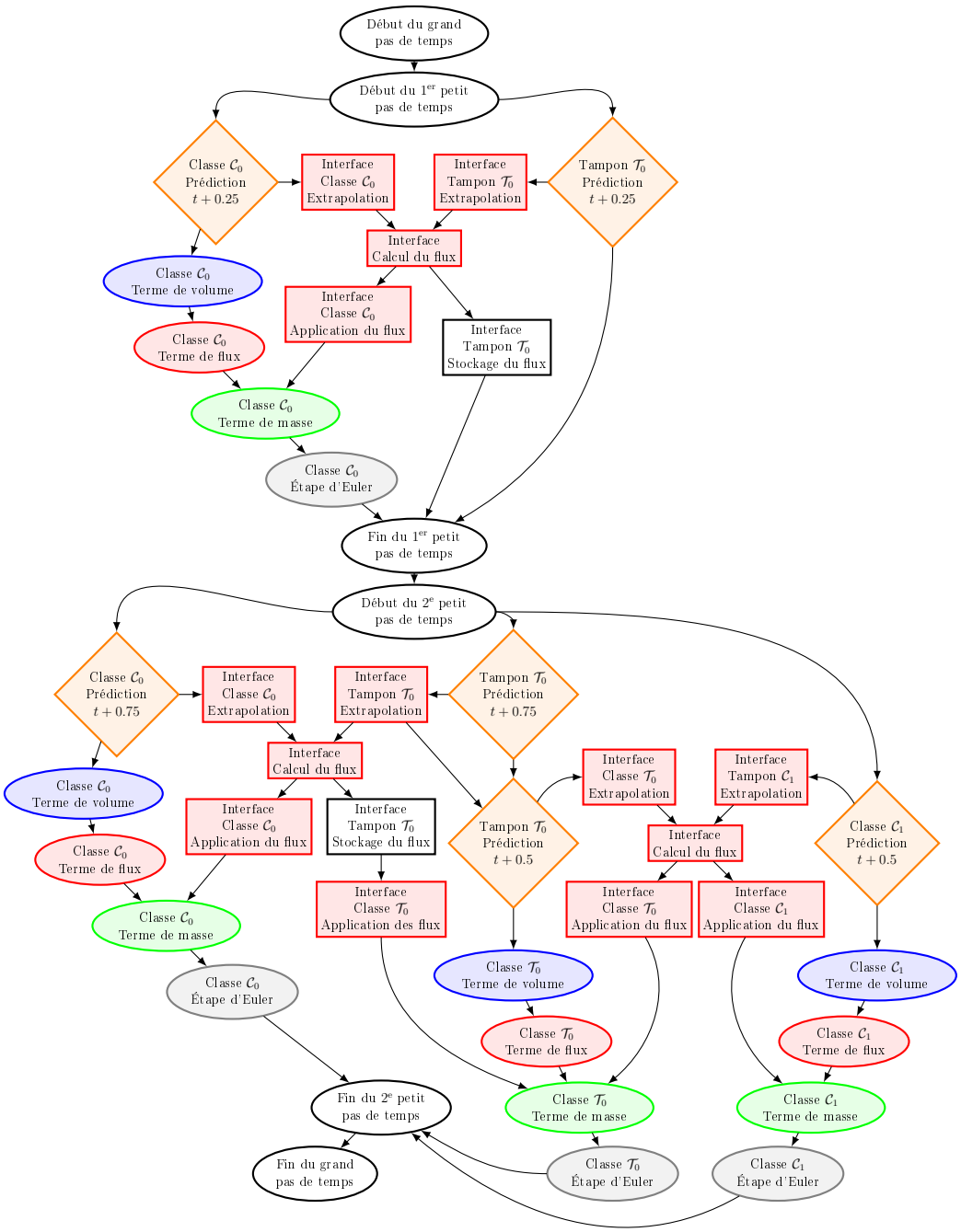
Les nœuds du maillage sont placés à l'aide d'une bijection continue σ :

$$\begin{aligned} \sigma : [0\,;1] &\to [0\,;1] \\ x &\mapsto \sigma(x) \end{aligned} \tag{5.44}$$

définie pour contrôler la taille des mailles. Nous plaçons $N+1$ nœuds $(x_i)_{i\in[\![0\,;N]\!]}$ délimitant N mailles :

$$x_i = a + (b-a)\sigma\left(\frac{i}{N}\right), \tag{5.45}$$

FIGURE 5.7 – Graphe des tâches d'un pas de temps $\Delta t^l_{\max}$ du schéma couplé GD-LTS2 pour 2 classes de pas de temps.

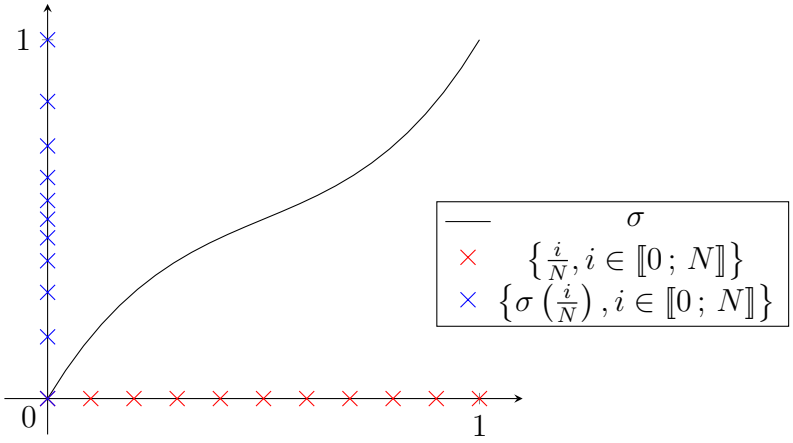


d'où :

$$h_i = x_i - x_{i-1} = (b-a)\left(\sigma\left(\frac{i}{N}\right) - \sigma\left(\frac{i-1}{N}\right)\right). \tag{5.46}$$

Ainsi, lorsque $\sigma(x) = x$, les mailles sont homogènes et le pas de temps est uniforme sur le maillage. De manière plus générale, si nous prenons pour σ une fonction polynomiale strictement croissante et d'ordre supérieur à 1, nous pouvons obtenir un maillage contenant des mailles dilatées sur les bords, aux voisinages de a et b, et contractées au centre au voisinage de 0 (figure 5.8). Pour la réalisation des tests de la méthode, nous nous plaçons sur l'intervalle $[-1\,;1]$.

FIGURE 5.8 – Application de la fonction σ sur l'intervalle $[0\,;1]$ pour $\sigma(x) = 2x^3 - 3x^2 + 2x$. L'intervalle de départ est représenté par l'axe horizontal. L'intervalle image est représenté par l'axe vertical.



Une solution exacte du problème (5.43) est donnée par :

$$w_{\text{exact}}(x,t) = \cos(2\pi(x-ut)). \tag{5.47}$$

Nous simulons donc le déplacement de la solution initiale :

$$w_0(x) = \cos(2\pi x). \tag{5.48}$$

Nous effectuons ces simulations dans les cas $u = -1$ et $u = 1$.

5.4.2 Prédicteur exact

Afin de simplifier l'implémentation de cette application en dimension 1, nous choisissons un prédicteur local exact. Soit w_L une solution approchée sur la maille L. Nous pouvons exprimer la valeur de sa dérivée temporelle à partir de sa décomposition dans la base d'approximation :

$$\partial_x w_L(x,t) \approx \partial_x \left(\sum_{i=0}^{d} w_{L,i}(t)\psi_{L,i}(x) \right) = \sum_{i=0}^{d} w_{L,i}(t)\partial_x \psi_{L,i}(x) \tag{5.49}$$

et de la formulation de l'équation du transport (5.43) :

$$\partial_t w_L = -u\partial_x w_L \approx -u \sum_{i=0}^{d} w_{L,i}(t)\partial_x \psi_{L,i}(x). \tag{5.50}$$

Ainsi, en décomposant aussi les dérivées des fonctions de base dans la base d'approximation, et par orthogonalité des fonctions de base, nous obtenons le système matriciel suivant :

$$\partial_t \begin{pmatrix} w_{L,0} \\ \vdots \\ w_{L,d} \end{pmatrix} + \boldsymbol{A} \begin{pmatrix} w_{L,0} \\ \vdots \\ w_{L,d} \end{pmatrix} = 0, \tag{5.51}$$

avec

$$\boldsymbol{A} = u(\partial_x \psi_{L,j}(g_{L,i}))_{i,j\in[\![0\,;\,d]\!]} = (\tau_L')^{-1} u(\partial_{\hat{x}} \hat{\psi}_j(\hat{g}_i))_{i,j\in[\![0\,;\,d]\!]} \tag{5.52}$$

et

$$\tau_L' = \mathrm{jac}\,(\tau_L) = h_L. \tag{5.53}$$

Le prédicteur local est donné par une exponentielle de matrice, qui est calculable exactement (par exemple avec un logiciel de calcul formel) :

$$\begin{pmatrix} v_{L,0} \\ \vdots \\ v_{L,d} \end{pmatrix}_p = \exp(\Delta t \boldsymbol{A}) \begin{pmatrix} w_{L,0} \\ \vdots \\ w_{L,d} \end{pmatrix}_s. \tag{5.54}$$

L'expression de ce prédicteur est donnée pour les ordres d'interpolation 0, 1 et 2 dans le tableau 5.9.

FIGURE 5.9 – Expression du prédicteur exact utilisé dans l'implémentation du schéma LTS2 appliqué à l'équation du transport en dimension 1 pour les ordres d'interpolation 0, 1 et 2. Nous notons $\lambda = (\tau'_L)^{-1} u \Delta t$.

d	Solution en t	Prédiction exacte en $t + \Delta t$
0	$w_{L,0}$	$w_{L,0}$
1	$w_{L,0}$	$(1+\lambda)w_{L,0} - \lambda w_{L,1}$
	$w_{L,1}$	$\lambda w_{L,0} + (1-\lambda)w_{L,1}$
2	$w_{L,0}$	$(1+\lambda)(1+2\lambda)w_{L,0} - 4\lambda(1+\lambda)w_{L,1} + \lambda(1+2\lambda)w_{L,2}$
	$w_{L,1}$	$\lambda(1+2\lambda)w_{L,0} + (1-2\lambda)(1+2\lambda)w_{L,1} - \lambda(1-2\lambda)w_{L,2}$
	$w_{L,2}$	$-\lambda(1-2\lambda)w_{L,0} + 4\lambda(1-\lambda)w_{L,1} + (1-\lambda)(1-2\lambda)w_{L,2}$

5.4.3 Condition de stabilité et convergence

Ce cas d'application a expérimentalement été vérifiée stable en temps long avec un pas de temps uniforme sur tout le maillage (schéma LRK2 (5.24)) pour une condition CFL de type :

$$\Delta t \leq \frac{C(d)}{|u|} \Delta x \tag{5.55}$$

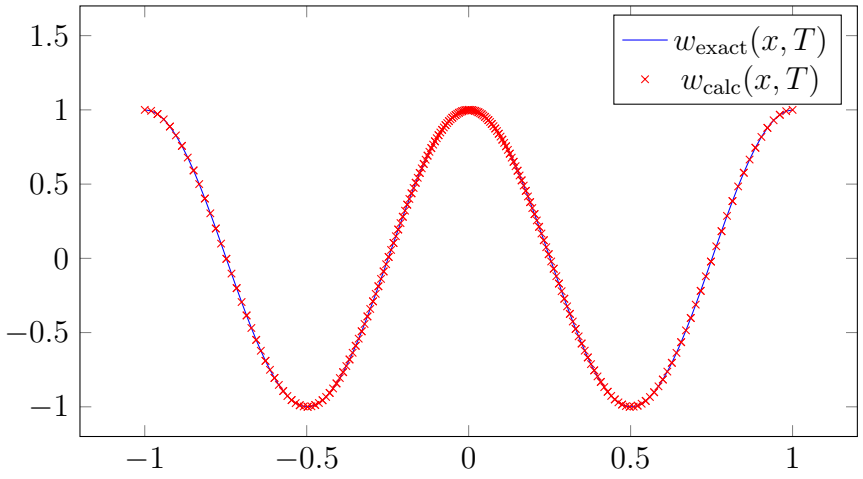
avec C une constante dépendante de l'ordre d dont les valeurs sont données dans le tableau 5.10. Ces valeurs ont été utilisées dans le cas du schéma LTS2 (5.41) et n'ont généré aucune instabilité.

FIGURE 5.10 – Valeur de la constante CFL de stabilité C (5.55) utilisée dans l'implémentation du schéma LRK2 (et LTS2) appliqué à l'équation du transport en dimension 1 en fonction de l'ordre d'interpolation d.

d	$C(d)$
0	0.56
1	0.50
2	0.17
3	0.08
4	0.05

La solution calculée converge vers la solution exacte pour ces conditions de stabilité. La figure 5.11 présente un résultat dans le cas LRK2 avec $u = 1$, sur 100 mailles et pour l'ordre d'interpolation $d = 2$ jusqu'au temps $T = 1$. Nous

FIGURE 5.11 – Solution calculée w_{calc} et solution exacte w_{exact} de l'équation du transport en dimension 1 dans le cas du schéma LRK2 (pas de temps homogène) avec $u = 1$, $N = 100$, $d = 2$ et $T = 1$.



avons choisi la bijection $\sigma(x) = 2x^3 - 3x^2 + 2x$ représentée dans la figure 5.8 pour ce résultat, avec un pas de temps homogène sur tout le maillage.

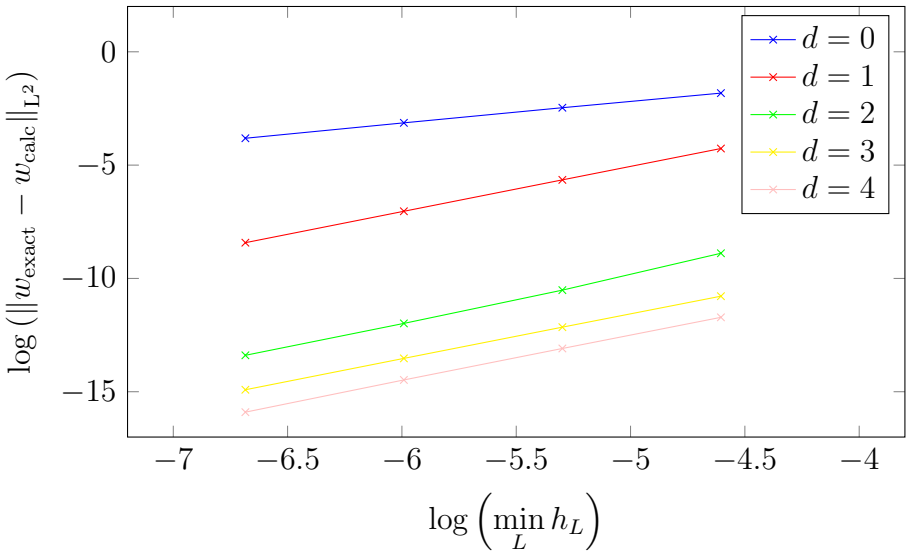
Les courbes de convergence du schéma LRK2 sont données dans la figure 5.12. Nous pouvons constater que cette méthode est généralement d'ordre 2 en temps, tout comme le schéma RK2. Elle est d'ordre 1 en temps uniquement lorsque l'ordre d'interpolation spatiale choisi est 0.

Remarque 31. *Les résultats de convergence présentés dans les figures 5.12 et 5.14 sont ceux du schéma en temps théoriquement d'ordre* 2. *Pour constater l'ordre de convergence de la méthode d'intégration spatiale, le coefficient CFL devrait être choisi très petit afin de minimiser l'erreur induite par l'intégration temporelle.*

Dans le cas de l'application du schéma LTS2, la solution calculée converge aussi vers la solution exacte, dans les mêmes conditions que pour le résultat LRK2 de la figure 5.11, à savoir $u = 1$, $N = 100$, $d = 2$ et $T = 1$. Le critère de stabilité $C(2) = 0.17$ donné dans le tableau 5.10 est encore vérifié dans ce cas. Nous avons là aussi choisi la bijection $\sigma(x) = 2x^3 - 3x^2 + 2x$ représentée dans la figure 5.8, mais cette fois sans imposer de pas de temps homogène. Le calcul a donc été effectué avec 2 classes de pas de temps. Ce résultat est présenté dans la figure 5.13. Les classes de pas de temps des mailles y sont aussi représentées par rapport à l'axe de droite.

Les courbes de convergence du schéma LTS2 sont données dans la figure 5.14. Nous pouvons constater que cette méthode est aussi d'ordre 2

FIGURE 5.12 – Convergence du schéma LRK2 (pas de temps homogène) appliqué à l'équation du transport en dimension 1 avec $u = 1$ et $T = 1$. Les points correspondent à l'application de la méthode pour les premiers termes ($i \in [\![0\,;\,3]\!]$) de la suite $100 \cdot 2^i$ en nombre de mailles.

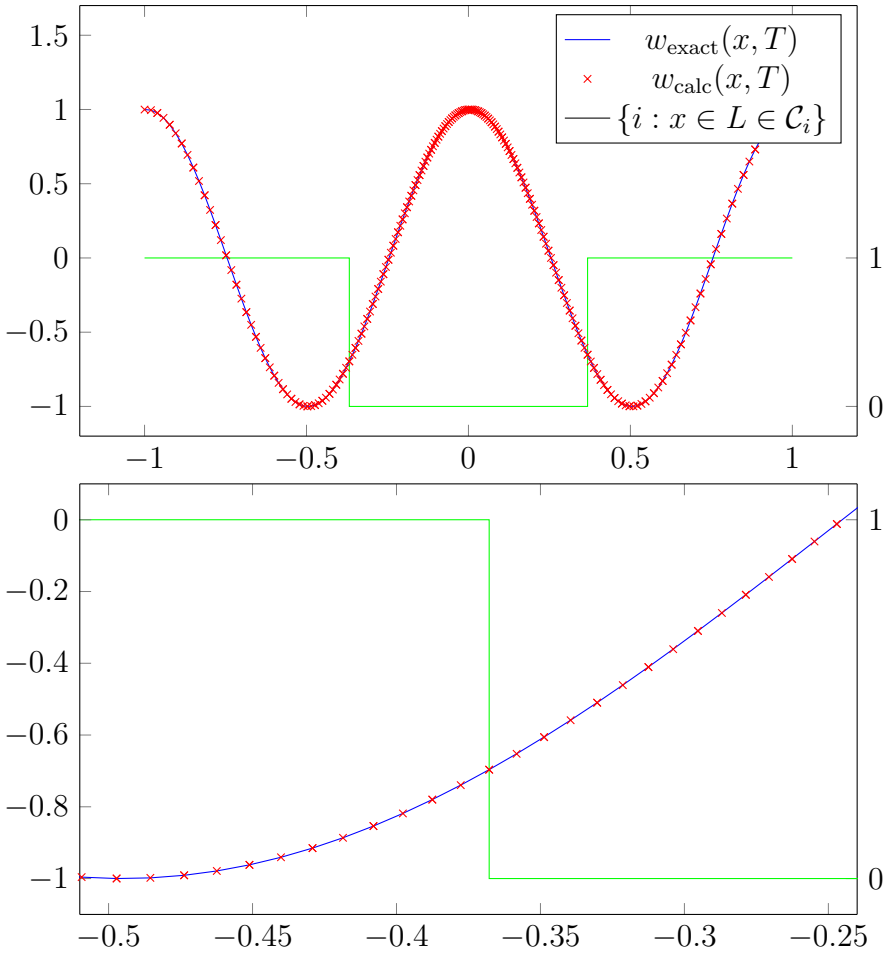


en temps, tout comme les schémas RK2 et LRK2. Là aussi, ce schéma est d'ordre 1 en temps uniquement lorsque l'ordre d'interpolation spatiale choisi est 0.

Ces résultats démontrent la capacité de cette méthode à produire des solutions de bonne qualité. Nous avons aussi la preuve que notre méthode d'application des flux sur les mailles tampon est conservative. Dans notre cas d'étude, la solution se déplace dans un sens, mais les interfaces entre les classes de pas de temps sont définies dans les deux sens de la direction étudiée. Nous avons présenté des résultats dans le cas $u = 1$. Les résultats dans le cas $u = -1$ sont identiques, hormis le sens de déplacement opposé.

Remarque 32. *Dans notre implémentation, nous avons choisi d'utiliser un prédicteur exact. La propriété de convergence de la méthode dépend de la qualité du prédicteur. Dans la section suivante, pour l'application aux équations de Maxwell en dimension 3, nous utiliserons l'opérateur $\mathcal{G}_l$ dans les phases de prédiction.*

FIGURE 5.13 – Solution calculée w_{calc} et solution exacte w_{exact} de l'équation du transport en dimension 1 dans le cas du schéma LTS2 (pas de temps local) pour $u = 1$, $N = 100$, $d = 2$, $T = 1$ et 2 classes de pas de temps. La valeur de la solution est donnée par l'axe de gauche. La classe de pas de temps est donnée par l'axe de droite. Agrandissement de la solution au voisinage d'une interface située entre les classes de pas de temps.

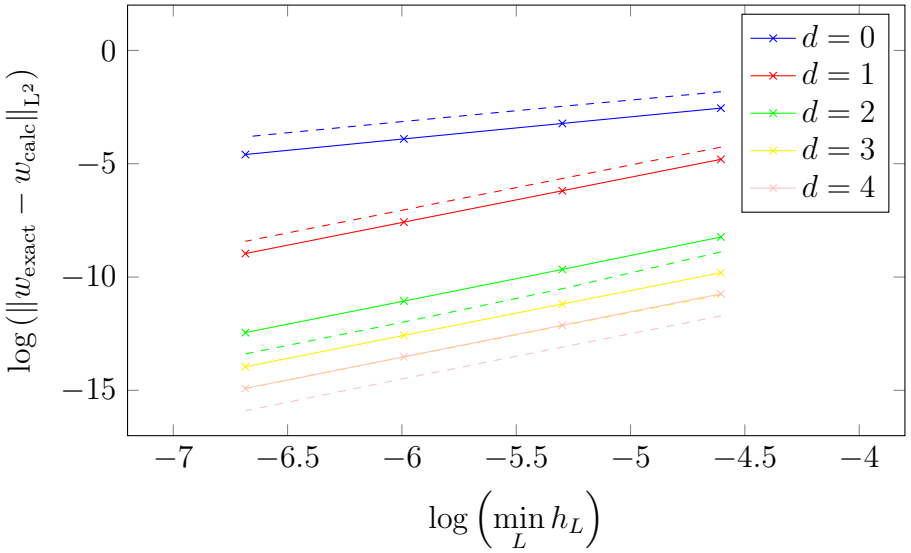


5.4.4 Performances

Remarque 33. *Le programme qui nous a servi à évaluer les performances de cette méthode appliquée à l'équation du transport en dimension* 1 *a été codé en C, sans optimisation particulière.*

Maintenant que la convergence et la stabilité de la méthode sont assurées, nous pouvons évaluer le gain de performance apporté par celle-ci. Intuitive-

FIGURE 5.14 – Convergence du schéma LTS2 (pas de temps local) appliqué à l'équation du transport en dimension 1, pour $u = 1$ et $T = 1$. Les points correspondent à l'application de la méthode pour les premiers termes ($i \in [\![0\,;\,3]\!]$) de la suite $100 \cdot 2^i$ en nombre de mailles. Les courbes en pointillés sont, à titre de comparaison, celles obtenues avec un pas de temps homogène (figure 5.12). Les résultats de convergence pour LRK2 avec $d = 3$ sont masqués par les résultats pour LTS2 avec $d = 4$.



ment, une telle méthode devrait produire un temps de calcul inférieur de la forme :

$$T_{\text{s}}(\text{LTS2}, \mathcal{M}) \geq T_{\text{i}}(\text{LTS2}, \mathcal{M}) = \frac{T_{\text{s}}(\text{LRK2}, \mathcal{M})}{\#\mathcal{M}} \sum_{i=0}^{N_t} \frac{1}{2^i} \#\mathcal{C}_i, \tag{5.56}$$

où T_{s} représente le temps de calcul d'une simulation pour une méthode donnée sur un ensemble de mailles, T_{i} représente le temps de calcul idéal (théorique) d'une méthode et $\#[.]$ représente le cardinal d'un ensemble.

En effet, au cours d'une étape du schéma LTS2, chaque classe de pas de temps $\mathcal{C}_i$ n'effectue que $2^{N_t - i}$ étapes de prédiction et mise à jour dans le calcul d'un pas de temps Δt^l_{max} qui se décompose en 2^{N_t} pas de temps Δt_{min}. Le temps de calcul pour une classe de pas de temps est donc donné par la proportion d'étapes effectuées :

$$\frac{2^{N_t - i}}{2^{N_t}} = \frac{1}{2^i} \tag{5.57}$$

multiplié par le temps de calcul engendré par les mailles de cette classe :

$$T_{\mathrm{s}}(\mathrm{LTS2}, \mathcal{C}_i) \approx \frac{1}{2^i} \frac{T_{\mathrm{s}}(\mathrm{LRK2}, \mathcal{M})}{\#\mathcal{M}} \#\mathcal{C}_i. \tag{5.58}$$

La formule (5.56) nous donne une expression du temps de simulation idéal en appliquant le schéma LTS2. Cependant, cette formule ne tient pas compte des mailles tampon qui nécessitent plus de calculs que leurs mailles voisines.

Nous donnons des résultats de performance en fonction de la taille des classes de pas de temps dans le tableau 5.15. Nous nous contentons de 2 classes de pas de temps et choisissons des bijections de la forme $\sigma(x) = \alpha x^3 + (1-\alpha)x$. Ainsi, nous imposons les classes de pas de temps :

$$\mathcal{C'}_0 = \mathcal{C}_0, \tag{5.59a}$$

$$\mathcal{C'}_1 = \bigcup_{i \in [\![1\,;\,N_t]\!]} \mathcal{C}_i. \tag{5.59b}$$

FIGURE 5.15 – Facteurs de gain de temps mesurés (et idéal) du schéma LTS2 appliqué à l'équation du transport en dimension 1 par rapport au schéma LRK2 pour différentes tailles de 2 classes de pas de temps et différents ordres d'interpolation d. Le paramètre α intervient dans l'expression de la bijection $\sigma(x) = \alpha x^3 + (1-\alpha)x$.

α	$\#\mathcal{C'}_0$	$\#\mathcal{C'}_1$	$d=1$	$d=2$	$d=3$	Idéal
0.5	579	421	0.99	0.90	0.80	0.7895
0.7	380	620	0.87	0.78	0.73	0.69
0.9	194	806	0.73	0.67	0.64	0.597
0.99	60	940	0.66	0.61	0.56	0.53
0.999	20	980	0.63	0.59	0.54	0.51
1	3	997	0.63	0.58	0.53	0.5015

Nous constatons que le gain de temps résultant de l'utilisation du schéma LTS2 est principalement lié au nombre de mailles présentes dans la classe de pas de temps $\mathcal{C}_0$. Moins il y a de mailles dans cette classe, plus le gain de temps est conséquent. Ce résultat est logique puisque la majeure partie des calculs est effectuée dans cette classe. Idéalement, le temps LTS2 est la moitié du temps LRK2 lorsque le nombre de mailles dans la classe $\mathcal{C}_0$ passe à 0. Toutes les mailles sont alors de classe $\mathcal{C'}_1$ et la résolution s'effectue globalement avec le pas de temps $2\Delta t_{\min}$.

Cette méthode est aussi plus efficace en ordre d'interpolation spatiale élevé. Ce phénomène s'explique par le fait que le surcoût de calculs engendré par la structure du programme et les mailles tampon sont compensés par la complexité globale des calculs lorsque l'ordre augmente. Bien qu'en dimension 1 et faible ordre d'interpolation spatiale, ce surcoût ne représente que peu de calculs contrairement à une méthode en dimension 3 avec un ordre d'interpolation élevé.

Enfin, pour les cas en ordre spatial 3 présentés dans le tableau 5.15, les performances sont proches de celles données théoriquement par la formule du temps idéal (5.56).

FIGURE 5.16 – Facteurs de gain de temps mesurés (et idéal) du schéma LTS2 appliqué à l'équation du transport en dimension 1 par rapport au schéma LRK2 pour différentes tailles de classes de pas de temps et différents ordres d'interpolation d. Le paramètre α intervient dans l'expression de la bijection $\sigma(x) = \alpha x^3 + (1-\alpha)x$.

α	$(\#\mathcal{C}_0, \#\mathcal{C}_1, \ldots)$	$d=1$	$d=2$	$d=3$	Idéal
0.5	$(579, 421)$	0.99	0.90	0.80	≈ 0.79
0.7	$(380, 277, 343)$	0.75	0.68	0.64	≈ 0.60
0.9	$(194, 141, 176, 236, 253)$	0.46	0.41	0.39	≈ 0.35
0.99	$(60, 43, 53, 71, 98, 138, 193, 273, 71)$	0.17	0.14	0.13	≈ 0.12

Dans le tableau 5.16, les résultats de performances sont présentés pour les mêmes simulations que celles présentées dans le tableau 5.15. Pour cette seconde salve de résultats, nous n'avons pas forcé le nombre de classes de pas de temps à 2. Celles-ci sont en nombre croissant en fonction du paramètre α de la bijection σ utilisée.

Nous constatons que plus le nombre de classes augmente, plus le temps idéal diminue et plus le gain de temps augmente. Ainsi, à l'ordre spatial 1, avec les 9 classes de pas de temps ($N_t = 8$) du cas $\alpha = 0.99$, le facteur d'accélération est d'approximativement 6. Ce facteur d'accélération passe à 7.6 pour l'ordre spatial 3.

D'importantes accélérations sont réalisables avec cette méthode, dans les cas où la géométrie est bien adaptée. Son efficacité est directement liée à l'hétérogénéité des mailles. Lorsque les tailles des mailles sont homogènes, aucun gain ne sera constaté. Au contraire, plus le facteur d'échelle entre les mailles est important et moins il y a de petites mailles, meilleure sera l'accélération.

Nous présentons des résultats de l'application de cette méthode aux équa-

tions de Maxwell en dimension 3 dans la section 7.2.5.

Conclusion

Dans ce chapitre, nous avons présenté une variante locale de la méthode GD décrite au chapitre 1. Cet opérateur d'intégration local permet d'alléger les calculs de flux et de découpler l'avancée en temps des mailles. Notons toutefois que cet opérateur local n'est à utiliser qu'en phase de prédiction (calcul des points d'intégration temporels), une mise à jour globale est nécessaire à chaque étape du calcul dans le but d'assurer la stabilité du schéma.

Cet opérateur local nous a permis de décrire deux schémas temporels. Le schéma LRK2 qui est une implémentation à prédiction locale du schéma RK2 (section 3.3.1). Ce schéma utilise un pas de temps homogène sur tout le maillage. Il a expérimentalement été démontré convergeant et stable. Nous verrons cependant dans la section 7.2.4 que ce schéma est moins intéressant en temps de calcul que le schéma RK2 car ce dernier admet un coefficient CFL de stabilité plus grand.

Nous avons aussi présenté le schéma LTS2 pour lequel le pas de temps est local à chaque maille. Pour ce schéma, les mailles sont réparties en classes de pas de temps : sous-ensembles de mailles sur lesquels le pas de temps est homogène. Le schéma LRK2 est alors localement appliqué sur ces classes avec une intégration temporelle spécifique aux interfaces entre les classes. Ce second schéma a lui aussi été expérimentalement démontré convergeant et stable. Les accélérations obtenues à l'aide de cette méthode sont très proches de l'accélération théorique en dimension 1. Nous constaterons aussi une accélération des calculs en dimension 3 dans la section 7.2.5. Ce schéma LTS2 permet de réduire considérablement le temps de simulation en divisant la quantité de calculs lorsque la géométrie est composée de mailles de tailles très variées.

Dans le chapitre suivant, nous allons présenter les résultats de l'exécution du solveur GD `schnaps` sur architecture hybride (composition de CPU et/ou GPU hétérogènes). Pour cela nous avons utilisé (entre autres) la bibliothèque StarPU développée à l'INRIA de Bordeaux [2,3]. Des expérimentations d'équilibrage automatique de la charge de calcul entre les différents supports d'exécution ont aussi été effectuées à l'aide de la bibliothèque de partitionnement `metis` [43].

Chapitre 6

Exécution sur ordinateur hybride

Dans les chapitres précédents, nous avons décrit la méthode Galerkin Discontinue (GD) de résolution des équations de Maxwell, puis nous avons développé l'implémentation parallèle de cette méthode que nous avons programmé, pouvant utiliser simultanément un grand nombre d'accélérateurs OpenCL identiques.

Après avoir présenté une adaptation algorithmique permettant d'appliquer un pas de temps local à chaque maille et allégeant considérablement la quantité de calculs, nous allons montrer dans quelle mesure un solveur GD s'adapte à l'exécution sur ordinateur hybride.

Les (super)calculateurs actuels sont généralement équipés de périphériques (de même nature) identiques. L'équilibrage de la charge de calcul est donc aisé entre périphériques de même nature. Cependant, bien que tous les CPU soient identiques et de même pour les GPU, les CPU sont très différents des GPU, tant au niveau de l'architecture que de leur puissance de calcul respective. Or, chaque nœud de calcul d'un (super)calculateur est généralement composé d'un CPU et d'un GPU. Ainsi, bien que le fait de savoir diviser efficacement un problème entre 2 GPU hétérogènes ne soit pas une priorité, il est très intéressant de savoir équilibrer la charge ce calcul entre un CPU et un GPU. Nous pourrions alors exploiter toute la puissance de calcul disponible.

À ces fins, nous avons utilisé la bibliothèque StarPU développée à l'INRIA de Bordeaux [2, 3]. Cette bibliothèque permet de se défaire de l'équilibrage manuel des tâches via un graphe des tâches. Ce dernier est automatiquement généré par StarPU selon une politique d'ordonnancement choisie par l'utilisateur (nous) et est basé sur la dépendance des données passées en entrée et en sortie des tâches. Les tâches sont codées sous forme de « *codelets* » écrits en langage C et pouvant contenir des soumissions de kernels OpenCL. Chaque

tâche peut être implémentée par plusieurs codelets (par exemple une version C pour CPU et une version C-OpenCL pour GPU) et StarPU choisit à l'exécution de chaque tâche quel codelet exécuter en fonction de ses performances passées et du périphérique de destination. Dans le code du solveur, toutes les tâches sont alors soumises dans un ordre séquentiel (dictant la dépendance des données) à StarPU. Les transferts de données sont eux aussi délégués à StarPU.

Les expérimentations de l'utilisation de cette bibliothèque ont été effectuées sur le solveur universitaire `schnaps` [36] et sont présentées en anglais dans l'article qui suit (publication à venir). Les notations et numérotations sont donc globalement différentes du reste de ce mémoire. Le solveur `schnaps` implémente la méthode GD avec une interpolation spatiale utilisant les points de Gauss-Lobatto. L'autre grande différence avec le solveur `teta-clac` est que les mailles sont des subdivisions régulières de macro-mailles équivalentes à des zones homogènes structurées globalement déformées.

OPTIMIZATION OF A DISCONTINUOUS FINITE ELEMENT SOLVER WITH OPENCL AND STARPU

PHILIPPE HELLUY, MICHEL MASSARO, LAURA MENDOZA, BRUNO WEBER

Abstract. `schnaps` is a finite element solver designed to simulate various physical phenomena. It is designed to run on hybrid computers made of several CPUs and GPUs. In order to address the hybrid architectures we rely on the StarPU runtime. StarPU allows to optimize in an incremental way a sequential algorithm in order to migrate to multicore parallelism and then to hybrid computing with OpenCL accelerators. StarPU proposes several task scheduling strategies in order to efficiently exploit the available computing power. We present the design of `schnaps` and the performance gain that we have obtained in electromagnetic simulations thanks to OpenCL codelets.

1. Introduction

The development of engineering simulation software is difficult. On one hand, the user is interested in handling more and more complex physical phenomena, in devices with arbitrary geometries. These constraints require to adopt a generic and abstract software engineering approach in order to ensure the generality of the code. On the other hand, the user also wants to harness the full computational power of his hybrid computer. This requirement generally imposes to use low level hardware optimization hacks that are not always compatible with a generic and elegant approach. In addition, as the hardware evolves, optimizations of yesterday are not necessarily relevant for the devices of tomorrow...

OpenCL is a nice software environment for optimization purposes. It presents the good balance between an abstract view of the computing devices and some important hardware aspects, such as local memory for accelerating data transfers or subgroups for minimizing synchronization barriers. However depending on architecture, optimizations written for one accelerator may be completely irrelevant for another one. This is especially true with local memory optimizations. For instance, cache prefetching is generally efficient for discrete GPUs while inefficient for IGPs.

In this paper, we present our practical approach to this issue, with the help of the StarPU runtime system. StarPU is developed at Inria Bordeaux since 2006[1]. StarPU is a runtime C library based on the dataflow paradigm. The programmer has to split the computation workload into abstract computational tasks. Each task processes data buffers that can be in `read` (R), `write` (W) or `read/write` (RW) mode.

The tasks are then conveniently implemented into C *codelets*. It is possible (and recommended) to write several implementations of the same task into several codelets. For instance, one can imagine to write an unoptimized codelet for validation purposes and one or several optimized OpenCL codelets. The user then submits his tasks in a sequential way to StarPU. At runtime, StarPU is able to construct a task graph based on buffer dependencies. The tasks are submitted to the available

[1]We are not involved in the creation and development of StarPU, but only users.

accelerators, in parallel if the dependencies allow it. In addition, StarPU automatically handles data transfers between the accelerators. It is also able to measure the efficiency of the different codelets' implementations in order to choose the best, according to the scheduling strategy.

With StarPU, implementing a complex dataflow of OpenCL kernels becomes easier. Indeed, the programmer does not have to handle the kernel dependencies with OpenCL events. In addition, it is possible to first write a well validated pure C version of the software, with only C codelets. Then, one can enrich the tasks with OpenCL codelets, which allows an incremental optimization of the code. At each stage, StarPU should be able to use the available codelets in an efficient way.

In this paper, we describe how we applied the StarPU philosophy in order to optimize a discontinuous finite element solver for conservation laws. The outlines are as follows: first we will present in its main lines the Discontinuous Galerkin (DG) scheme that is used in the solver. Then, after a short presentation of StarPU, we will explain how we have integrated the OpenCL optimizations into the DG solver. Finally, we will present some numerical experiments.

2. Discontinuous Galerkin method

The Discontinuous Galerkin (DG) method is a general finite element method for approximating systems of conservation laws of the form

$$\partial_t \mathbf{w} + \sum_{k=1}^{D} \partial_k \mathbf{f}^k(\mathbf{w}) = 0.$$

The unknown is the vector of conservative variables $\mathbf{w}(\mathbf{x}, t) \in \mathbb{R}^m$ depending on the space variable $\mathbf{x} = (x^1, \ldots, x^D) \in \mathbb{R}^D$ and on time t. In this paper, the space dimension is $D = 3$. We adopt the notations ∂_t for the partial derivative with respect to t and ∂_k for the the partial derivative with respect to x^k. Let $\mathbf{n} = (n_1, \ldots, n_D) \in \mathbb{R}^D$ be a spatial direction, the flux in direction $\mathbf{n}$ is defined by

$$\mathbf{f}(\mathbf{w}, \mathbf{n}) = \sum_{k=1}^{D} n_k \mathbf{f}^k(\mathbf{w}).$$

For instance in this work we consider the numerical simulation of an electromagnetic wave. In this particular case, the conservative variables are

$$\mathbf{w} = (\mathbf{E}^{\mathrm{T}}, \mathbf{H}^{\mathrm{T}}, \lambda, \mu)^{\mathrm{T}} \in \mathbb{R}^m, \quad m = 8.$$

where $\mathbf{E} \in \mathbb{R}^3$ is the electric field, $\mathbf{H} \in \mathbb{R}^3$ is the magnetic field and λ, μ are divergence cleaning potentials (Munz et al. [2000]). The flux is given by

$$\mathbf{f}(\mathbf{w}, \mathbf{n}) = \begin{pmatrix} -\mathbf{n} \times \mathbf{H} + \lambda \mathbf{n} \\ \mathbf{n} \times \mathbf{E} + \mu \mathbf{n} \\ c\mathbf{n} \cdot \mathbf{E} \\ c\mathbf{n} \cdot \mathbf{H} \end{pmatrix}$$

and $c > 1$ is the divergence cleaning parameter.

We consider a mesh $\mathcal{M}$ of Ω made of open sets, called *cells*, $\mathcal{M} = \{L_i,\, i = 1 \ldots N_c\}$. In the most general setting, the cells satisfy

(1) $L_i \cap L_j = \emptyset$, if $i \neq j$;
(2) $\overline{\cup_i L_i} = \overline{\Omega}$.

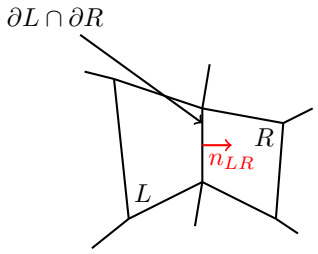


FIGURE 2.1. Convention for the L and R cells orientation.

In each cell $L \in \mathcal{M}$, we consider a basis of functions $(\varphi_{L,i}(\mathbf{x}))_{i=0\dots N_d-1}$ constructed from polynomials of order d. We denote by h the maximal diameter of the cells. With an abuse of notation we still denote by $\mathbf{w}$ the approximation of $\mathbf{w}$, defined by

$$\mathbf{w}(\mathbf{x},t) = \sum_{j=0}^{N_d-1} \mathbf{w}_{L,j}(t)\varphi_{L,j}(\mathbf{x}), \quad \mathbf{x} \in L.$$

The DG formulation then reads: find $\mathbf{w}_{L,j}$ such that for all cell L and all test function $\varphi_{L,i}$

$$\int_L \partial_t \mathbf{w}_L \varphi_{L,i} - \int_L \mathbf{f}(\mathbf{w}_L, \boldsymbol{\nabla}\varphi_{L,i}) + \int_{\partial L} \mathbf{f}(\mathbf{w}_L, \mathbf{w}_R, \mathbf{n})\varphi_{L,i} = 0. \tag{2.1}$$

In this formula (see Figure 2.1):

- R denotes the neighboring cell to L along its boundary $\partial L \cap \partial R$, or the exterior of Ω on $\partial L \cap \partial\Omega$.
- $\mathbf{n} = \mathbf{n}_{LR}$ is the unit normal vector on ∂L oriented from L to R.
- $\mathbf{w}_R$ denotes the value of $\mathbf{w}$ in the neighboring cell R on $\partial L \cap \partial R$.
- If L is a boundary cell, one may have to use the boundary values instead: $\mathbf{w}_R = \mathbf{w}_b$ on $\partial L \cap \partial\Omega$.
- $\mathbf{f}(\mathbf{w}_L, \mathbf{w}_R, \mathbf{n})$ is the standard upwind numerical flux encountered in many finite volume or DG methods.

In our application, we consider hexahedral cells. We have a reference cell

$$\hat{L} =]-1,1[^D$$

and a smooth transformation $\mathbf{x} = \boldsymbol{\tau}_L(\hat{\mathbf{x}})$, $\hat{\mathbf{x}} \in \hat{L}$, that maps $\hat{L}$ on L

$$\boldsymbol{\tau}_L(\hat{L}) = L.$$

We assume that $\boldsymbol{\tau}_L$ is invertible and we denote by $\boldsymbol{\tau}'_L$ its (invertible) Jacobian matrix. We also assume that $\boldsymbol{\tau}_L$ is a direct transformation

$$\det \boldsymbol{\tau}'_L > 0.$$

In our implementation, $\boldsymbol{\tau}_L$ is a quadratic map based on hexahedral curved "H20" finite elements with 20 nodes. The mesh of H20 finite elements is generated by `gmsh` (Geuzaine and Remacle [2009]).

On the reference cell, we consider the Gauss-Lobatto points $(\hat{\mathbf{x}}_i)_{i=0\dots N_d-1}$, $N_d = (d+1)^D$ and associated weights $(\omega_i)_{i=0\dots N_d-1}$. They are obtained by tensor products of the $(d+1)$ one-dimensional

Gauss-Lobatto (GL) points on $]-1,1[$. The reference GL points and weights are then mapped to the physical GL points of cell L by

$$\mathbf{x}_{L,i} = \boldsymbol{\tau}_L(\hat{\mathbf{x}}_i), \quad \omega_{L,i} = \omega_i \det \boldsymbol{\tau}'_L(\hat{\mathbf{x}}_i) > 0. \tag{2.2}$$

In addition, the six faces of the reference hexahedral cell are denoted by F_ϵ, $\epsilon = 1 \dots 6$ and the corresponding outward normal vectors are denoted by $\hat{\mathbf{n}}_\epsilon$. One advantage of choosing the GL points is that the cells and the faces share the same quadrature points. We use the following notations to define the face quadrature weights:

- if a GL point $\hat{\mathbf{x}}_i \in F_\epsilon$, we denote by μ_i^ϵ the corresponding quadrature weight on face $\mathbf{w}_\epsilon$;
- we also use the convention that $\mu_i^\epsilon = 0$ if $\hat{\mathbf{x}}_i$ does not belong to face F_ϵ.

Let us remark that a given GL point $\hat{\mathbf{x}}_i$ can belong to several faces when it is on an edge or in a corner of $\hat{L}$. Because of symmetry, we observe that if $\mu_i^\epsilon \neq 0$, then the weight μ_i^ϵ does not depend on ϵ.

We then consider basis functions $\hat{\varphi}_i$ on the reference cell: they are the Lagrange polynomials associated to the Gauss-Lobatto points and thus satisfy the interpolation property

$$\hat{\varphi}_i(\hat{\mathbf{x}}_j) = \delta_{ij}.$$

The basis functions on cell L are then defined according to the formula

$$\varphi_{L,i}(\mathbf{x}) = \hat{\varphi}_i(\boldsymbol{\tau}_L^{-1}(\mathbf{x})).$$

In this way, they also satisfy the interpolation property

$$\varphi_{L,i}(\mathbf{x}_{L,j}) = \delta_{ij}. \tag{2.3}$$

In this paper, we only consider conformal meshes: the GL points on cell L are supposed to match the GL points of cell R on their common face. Dealing with non-matching cells is the object of a forthcoming work. Let L and R be two neighboring cells. Let $\mathbf{x}_{L,j}$ be a GL point in cell L that is also on the common face between L and R. In the case of conformal meshes, it is possible to define the index j' such that

$$\mathbf{x}_{L,j} = \mathbf{x}_{R,j'}.$$

Applying a numerical integration to (2.1), using (2.2) and the interpolation property (2.3), we finally obtain

$$\partial_t \mathbf{w}_{L,i} \omega_{L,i} - \sum_{j=0}^{N_d-1} \mathbf{f}(\mathbf{w}_{L,j}, \boldsymbol{\nabla}\varphi_{L,i}(\mathbf{x}_{L,j}))\omega_{L,j} + \sum_{\epsilon=1}^{6} \mu_i^\epsilon \mathbf{f}(\mathbf{w}_{L,i}, \mathbf{w}_{R,i'}, \mathbf{n}_\epsilon(\mathbf{x}_{L,i})) = 0. \tag{2.4}$$

We have to detail how the gradients and normal vectors are computed in the above formula. Let $\mathbf{A}$ be a square matrix. We recall that the cofactor matrix of $\mathbf{A}$ is defined by

$$\mathrm{co}(\mathbf{A}) = \det(\mathbf{A}) \left(\mathbf{A}^{-1}\right)^{\mathrm{T}}. \tag{2.5}$$

The gradient of the basis function is computed from the gradients on the reference cell using (2.5)

$$\boldsymbol{\nabla}\varphi_{L,i}(\mathbf{x}_{L,j}) = \frac{1}{\det \boldsymbol{\tau}'_L(\hat{\mathbf{x}}_j)} \mathrm{co}(\boldsymbol{\tau}'_L(\hat{\mathbf{x}}_j)) \hat{\nabla}\hat{\varphi}_i(\hat{\mathbf{x}}_j).$$

In the same way, the scaled normal vectors $\mathbf{n}_\epsilon$ on the faces are computed by the formula

$$\mathbf{n}_\epsilon(\mathbf{x}_{L,i}) = \mathrm{co}(\boldsymbol{\tau}'_L(\hat{\mathbf{x}}_i))\hat{\mathbf{n}}_\epsilon.$$

We introduce the following notation for the cofactor matrix

$$\mathbf{c}_{L,i} = \mathrm{co}(\boldsymbol{\tau}'_L(\hat{\mathbf{x}}_i)).$$

The DG scheme then reads

$$\partial_t \mathbf{w}_{L,i} - \frac{1}{\omega_{L,i}} \sum_{j=0}^{N_d-1} \mathbf{f}(\mathbf{w}_{L,j}, \mathbf{c}_{L,j}\hat{\nabla}\hat{\varphi}_i(\hat{\mathbf{x}}_j))\omega_j + \frac{1}{\omega_{L,i}} \sum_{\epsilon=1}^{6} \mu_i^\epsilon \mathbf{f}\left(\mathbf{w}_{L,i}, \mathbf{w}_{R,i'}, \mathbf{c}_{L,i}\hat{\mathbf{n}}_\epsilon\right) = 0. \tag{2.6}$$

In formula (2.6) we identify volume terms that are associated to Gauss-Lobatto points in the volume

$$V_{i,j} = \mathbf{f}(\mathbf{w}_{L,j}, \mathbf{c}_{L,j}\hat{\nabla}\hat{\varphi}_i(\hat{\mathbf{x}}_j))\omega_j \tag{2.7}$$

and surface terms that are associated to cell boundaries

$$S_{i,i'} = \mu_i^\epsilon \mathbf{f}\left(\mathbf{w}_{L,i}, \mathbf{w}_{R,i'}, \mathbf{c}_{L,i}\hat{\mathbf{n}}_\epsilon\right). \tag{2.8}$$

Then, once these terms have been accumulated, one has to apply the inverse of the mass matrix. Here, this matrix is diagonal and it corresponds to a simple multiplication of $\partial_t \mathbf{w}_{L,i}$ by

$$\frac{1}{\omega_{L,i}}. \tag{2.9}$$

Generally, i and i' are associated to unknowns of internal cells, *i.e.* cells that do not touch the boundaries. On boundary GL points, the value of $\mathbf{w}_{R,i'}$ is given by the boundary condition

$$\mathbf{w}_{R,i'} = \mathbf{w}_b(\mathbf{x}_{L,i}, t), \quad \mathbf{x}_{L,i} = \mathbf{x}_{R,i'}.$$

For practical reasons, it can be interesting to also consider $\mathbf{w}_{R,i'}$ as an artificial unknown in a fictitious cell. The fictitious unknown is then a solution of the differential equation

$$\partial_t \mathbf{w}_{R,i'} = \partial_t \mathbf{w}_b(\mathbf{x}_{L,i}, \cdot). \tag{2.10}$$

In the end, if we put all the unknowns in a large vector $\mathbf{W}(t)$, (2.6) and (2.10) read as a large system of coupled differential equations

$$\partial_t \mathbf{W} = \mathbf{F}(\mathbf{W}). \tag{2.11}$$

This set of differential equations is then numerically solved by a Runge-Kutta numerical method. In practice, we use a second order Runge-Kutta method (RK2).

3. Data-based parallelism and StarPU

StarPU is a runtime system library developed at Inria Bordeaux (Augonnet et al. [2011, 2012]). It relies on the data-based parallelism paradigm. The user has first to split its whole problem into elementary computational tasks. The elementary tasks are then implemented into *codelets*, which are simple C functions. The same task can be implemented differently into several codelets. This allows the user to harness special accelerators, such as vectorial CPU cores or OpenCL devices, for example. In the StarPU terminology these devices are called *workers*. If a codelet contains OpenCL kernel submissions, special utilities are available in order to map the StarPU buffers to OpenCL buffers.

For each task, the user also has to describe precisely what are the input data, in `read` mode, and the output data, in `write` or `read/write` mode. The user then submits the task in a sequential way to the StarPU system. StarPU is able to construct at runtime a task graph from the data dependencies. The task graph is analyzed and the tasks are scheduled automatically to the available workers (CPU cores, GPUs, *etc.*). If possible, they are executed in parallel into concurrent threads.

The data transfer tasks between the threads are automatically generated and managed by StarPU. OpenCL data transfers are also managed by StarPU.

When a StarPU program is executed, it is possible to choose among several schedulers. The simplest `eager` scheduler adopts a very simple strategy: the tasks are executed in the order of submission by the free workers, without optimization. More sophisticated schedulers, such as the `dmda` or `dmdar` scheduler, are able to measure the efficiency of the different codelets and the data transfer times, in order to apply a more efficient allocation of tasks.

Recently a new data access mode has been added to StarPU: the `commute` mode. In a task, a buffer of data can now be accessed in `commute` mode, in addition to the `write` or `read/write` modes. A `commute` access tells to StarPU that the execution of the corresponding task may be executed before or after other tasks containing commutative access. This allows StarPU to perform additional optimizations.

There also exists a MPI version of StarPU. In the MPI version, the user has to decide an initial distribution of data among the MPI nodes. Then the tasks are submitted as usual (using the function starpu_mpi_insert_task instead of starpu_insert_task). Required MPI communications are automatically generated by StarPU. For the moment, this approach does not guarantee a good load balancing. It is the responsibility of the user to migrate data from one MPI node to another to improve the load balancing, if necessary. The MPI version of StarPU is not tested in this work.

3.1. **Macrocell approach.** StarPU is quite efficient, but there is an unavoidable overhead due to the task submissions and to the on-the-fly construction and analysis of the task graph. Therefore it is important to ensure that the computational tasks are not too small, in which case the overhead would not be amortized, or not too big, in which case some workers would be idle. To achieve the right balance, we decided not to apply the DG algorithm at the cell level but to groups of cells instead.

The implementation of the DG scheme has been made into the `schnaps` software[2] which is a C99 software dedicated to the numerical simulation of conservation laws. In `schnaps`, we first construct a *macromesh* of the computational domain. Then each *macrocell* of the macromesh is split into *subcells*. See Figure 3.1. We also arrange the subcells into a regular sub-mesh of the macrocells. In this way, it is possible to apply additional optimizations. For instance, the subcells L of a same macrocell $\mathcal{L}$ share the same geometrical transformation $\boldsymbol{\tau}_L$, which saves memory access.

In `schnaps` we also defined an *interface* structure in order to manage data communications between the macrocells. An interface contains a copy of the data at the Gauss-Lobatto points that are common to the two neighboring macrocells.

To solve one time step of the DG method, here are the most important tasks:

(1) Interface extraction: this task simply extracts the values of $\mathbf{w}$ coming from the neighboring macrocells to the interface. In this task, the macrocell buffers are accessed in `read` mode and the interface data in `write` mode.
(2) Interface fluxes: this task only computes the numerical fluxes (2.8) that are located at the Gauss-Lobatto points on the interface. The fluxes are then applied to the neighboring macrocells. In this task, the interface data are accessed in `read` mode and the macrocell data in `read/write` mode. For a better efficiency, we also assume a `commute` access to the

[2] `http://schnaps.gforge.inria.fr/`

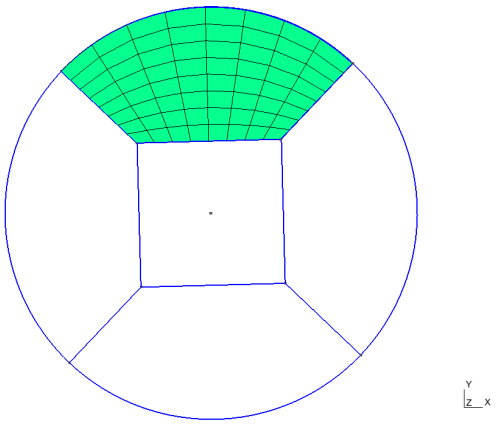


FIGURE 3.1. Macrocell approach: an example of a mesh made of five macrocells. Each macrocell is then split into several subcells. Only the subcells of the top macrocell are represented here (in green).

Algorithm 1 DG algorithm

for all interface:
Extract interface (copy the data from the neighboring macrocells to the interface)
If the interface is a boundary interface, compute the boundary fluxes (2.8) and apply them to the neighboring macrocell
Else the interface is an internal one. Compute the numerical fluxes (2.8) and apply them to the two neighboring macrocells
End for
Then, for each macrocell:
Compute and apply the numerical fluxes (2.8) between the subcells
Compute the volume terms (2.7) inside the subcells
Apply the inverse of the mass matrix (2.9) inside the subcells
End for

macrocell data. In this way the interface fluxes can be assembled in any order, which can help the StarPU scheduling.

(3) Interface boundary fluxes: this task computes the boundary data and only numerical fluxes (2.8) that are located at the boundary interface. The fluxes are applied to the neighboring macrocell. In this task, the interface data are accessed in `read` mode and the macrocell data in `read/write` and `commute` mode.
(4) Volume terms: this task applies the volumic terms (2.7) in a given macrocell. The macrocell data are accessed in `read/write/commute` mode.
(5) Subcell fluxes: this task applies the numerical subcell fluxes (2.8) that are internal to a given macrocell. The macrocell data are accessed in `read/write/commute` mode.

Additional simple tasks are needed to apply the Runge-Kutta algorithm. We do not describe them.

The general sequential DG algorithm is then given in Algorithm 1. Thanks to StarPU, this algorithm can be submitted in a sequential way. StarPU then uses the data dependency in order to distribute the tasks in parallel on the available workers.

In the next section we give more details about the implementation of the OpenCL codelets.

4. Hybrid C/OpenCL codelets

In order to attain better performance, we programmed an OpenCL version of the previously described codelets. As it is often the case, special care has to be given in order to improve the coalescence of memory access. The values of $\mathbf{w}$ at the Gauss-Lobatto points are stored into what we call a *field* structure. A field is attached to a macrocell. A given component of $\mathbf{w}$ in a field is located thanks to its subcell index `ic`, a Gauss-Lobatto index `ig` in the cell, and a component index `iw`. If the macrocell is cut into n_r subcells in each direction, if the polynomial order is d and for a system of m conservation laws, these indices have the following bounds

$$0 \leq \texttt{ic} < n_r^3, \quad 0 \leq \texttt{ig} < (d+1)^3, \quad 0 \leq \texttt{iw} < m.$$

The values of $\mathbf{w}$ in a given field are stored into a memory buffer `wbuf`. In order to test several memory arrangements, the index in the buffer is computed by a function `varindex(ic,ig,iw)` that we can easily change. For instance, we can consider the following formula, in which the electromagnetic components at a given Gauss-Lobatto point are grouped in memory

$$\texttt{varindex}(\texttt{ic},\texttt{ig},\texttt{iw}) = \texttt{iw}+m*(\texttt{ig}+(d+1)^3*\texttt{ic}), \tag{4.1}$$

or this formula

$$\texttt{varindex}(\texttt{ic},\texttt{ig},\texttt{iw}) = \texttt{ig}+(d+1)^3*(\texttt{iw}+m*\texttt{ic}), \tag{4.2}$$

in which the electromagnetic components are separated in memory. We have programmed an OpenCL codelet for each task described in Section 3.1. The most time consuming kernels are: (i) the kernel associated to the computations of the volume terms (2.7) inside the macrocells, *volume* kernel, and (ii) the kernel that computes the subcell fluxes, the *surface* kernel. Boundary and interface terms generally involve less computations.

A natural distribution of the workload is to associate one subcell to each OpenCL work-group and the computations at a Gauss point to a work-item. With this natural distribution, formula (4.2) ensures optimal coalescent memory access in the volume kernel, but the access is no more optimal in the surface kernel, because neighboring Gauss points on the subcell interfaces are not necessarily close in memory. Therefore, we prefer considering formula (4.1).

5. Numerical results

In this section, we present some practical experiments that we realized with `schnaps`. The first experiment deals with the efficiency of the StarPU C99 codelets on a multicore CPU. The second experiment deals with the efficiency of the OpenCL codelets. Then we test the code in a CPU/GPU configuration.

We approximate by the DG algorithm an exact plane wave solution of the Maxwell equations

$$\mathbf{E} = \frac{c}{r}\begin{pmatrix} -v \\ u \\ 0 \end{pmatrix}, \quad \mathbf{H} = \frac{c}{r}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \lambda = 0, \quad \mu = 0.$$

config.	CPU	# cores	mem. CPU	GPU	mem. GPU
"PC"	AMD FX-8320E 3.2 GHz	8	8 GB	NVIDIA GTX 1050 Ti	4 GB
"WS"	Intel Xeon E5-2609 1.7 GHz	2×8	64 GB	NVIDIA QUADRO P6000	24 GB

TABLE 1. Configurations of the computers used for the tests.

n_r	n_{cpu}	Time (s)	Ideal time	Efficiency
4	2	704	704	1
4	4	365	352	0.96
4	7	209	201	0.96
6	7	689	679	0.99
8	4	2902	2816	0.97
8	7	1650	1609	0.98

TABLE 2. Efficiency of the CPU codelets for the "PC" configuration with the `dmdar` scheduler.

with $c = -\cos(\pi/2(ux^1 + vx^2 - t))$ and $(u, v) = (\cos(\pi/4), \sin(\pi/4))$.

We consider the propagation of the previous electromagnetic plane wave into a torus-shaped domain Ω represented on Figure 5.1. In the DG solver, we consider only third order polynomials: $d = 3$. The domain is split into $M = 400$ macrocells. The macrocells are split into n_r subcells in each direction. The number of degrees of freedom is thus

$$n_{\text{d.o.f.}} = Mm(d+1)^3 n_r^3, \quad M = 400, \quad m = 8, \quad d = 3.$$

StarPU will have to take decisions on how to distribute efficiently the tasks on the available accelerators. We have performed our experiments on two different computers. The first configuration "PC" correspond to a standard desktop personal computer. The second configuration "WS" is made of a more powerful two-CPU workstation The technical details are listed in Table 1.

5.1. **Multicore CPU tests.** In the first test, we only activate the C99 codelets, and thus the GPU is not activated. We also vary the number of computing CPU cores from 2 to 7 for the "PC" configuration and from 2 to 15 for the "WS" configuration. One CPU core is anyway reserved for the main StarPU thread.

We perform a fixed number of n_t iterations of the RK2 algorithm. We assume that the number of elementary computing operations increases as $O(n_r^3)$. Ideally, with infinitely fast operations at interfaces, and with n_a identical CPU cores, the computations time would behave like

$$T \sim C n_t \frac{n_r^3}{n_a} \tag{5.1}$$

where C is fixed constant, depending on the hardware. In our benchmark the *efficiency* is the ratio of the measured execution time over this ideal time. Normally it should be smaller than one (because of communications). If it is close to one, the algorithm is very efficient. In Table 2 and 3, we observe an excellent efficiency of the StarPU task distribution on the CPU cores even on a machine with two NUMA nodes.

n_r	n_{cpu}	Time (s)	Ideal time	Efficiency
4	2	763	763	1
4	4	382	381	0.99
4	8	196	191	0.97
4	15	110	102	0.92
8	8	1586	1526	0.96
8	15	828	814	0.98

TABLE 3. Efficiency of the CPU codelets for the "WS" configuration with the `dmdar` scheduler.

n_r	config.	t_{CPU} (s)	t_{GPU}(s)	$t_{\text{CPU}}/t_{\text{GPU}}$	$t_{\text{CPU+GPU}}$ (s)	$t_{\text{CPU}}/t_{\text{CPU+GPU}}$
4	PC	209	73	2.86	32	6.53
6	PC	689	86	8.01	64	10.77
8	PC	1650	171	9.65	130	12.69
4	WS	110	86	1.28	36	3.05
8	WS	828	88	9.41	77	10.75

TABLE 4. Efficiency of the CPU/GPU implementation.

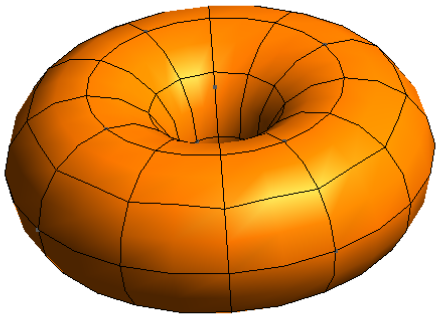


FIGURE 5.1. Computational domain Ω. Only the macrocells of the mesh are visible. The represented mesh contains 240 macrocells (poloidal refinement set to $n_{pol} = 3$ and toroidal refinement set to $n_{tor} = 3$. In the numerical experiments, we take $n_{pol} = 3$ and $n_{tor} = 5$, which leads to a mesh of 400 macrocells). Each macrocell is then cut regularly into the three directions.

5.2. **OpenCL codelets.** In this section, we compare the efficiency of the CPU/C99 and GPU/OpenCL codelets on the two configurations "PC" and "WS". We first perform a CPU-only computation with all the cores activated (7 for the "PC" platform and 15 for the "WS" platform) and a GPU-only computation. The results are given in Table 4. We observe that the OpenCL codelets are faster than the CPU codelets. The highest efficiency is achieved for the finer meshes ($n_r = 8$).

5.3. **Hybrid CPU/GPU computations.** Now that we are equipped with verified OpenCL codelets we can try to run the code with StarPU in order to see if it is able to distribute the tasks efficiently on the available accelerators.

One difficulty is to manage efficiently the data transfers between the accelerators. Several scheduling strategies are available in StarPU. With the `eager` scheduler, the task are distributed in the order of submission to the inactive device, without taking into account the cost of memory transfers. This strategy is obviously not optimal.

It is better to choose another scheduler, such as the `dmdar` scheduler. With the `dmdar` scheduler, the computational and memory transfer costs are evaluated in a preliminary benchmarking phase. Then an optimized scheduling is activated in order to better overlap computations and communications.

We perform computations by letting StarPU decide how to distribute the computations on the available CPU cores and GPU.

The results are given in the two last columns of Table 4. We observe that in all the situations, StarPU is able to get an additional gain from the CPU cores, even if the GPU codelets are faster.

6. Conclusion

We have proposed an optimized implementation of the Discontinuous Galerkin method on hybrid computer made of several GPUs and multicore CPUs. In order to manage the heterogeneous architecture easily and efficiently, we rely on OpenCL for the GPU computations and on the StarPU runtime for distributing the computational tasks on the available devices.

OpenCL programming becomes much easier because the task dependency is computed by StarPU. We only had to concentrate on the optimization of the individual OpenCL kernels and not on data distribution or memory transfers. We first tested the efficiency of our OpenCL kernels. We verified that the macrocell approach and cache prefetching strategy, while not optimal, give good results.

In addition, with a good choice of scheduler, and for heavy computations, we have shown that StarPU is able to overlap computations and memory transfer in a quite efficient way. It is also able to use the available CPU codelets to achieve even higher acceleration.

References

Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. Starpu: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, 23(2):187–198, 2011.

Cédric Augonnet, Olivier Aumage, Nathalie Furmento, Raymond Namyst, and Samuel Thibault. Starpu-mpi: Task programming over clusters of machines enhanced with accelerators. In *European MPI Users' Group Meeting*, pages 298–299. Springer, 2012.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

C-D Munz, Pascal Omnes, Rudolf Schneider, Eric Sonnendrücker, and Ursula Voss. Divergence correction techniques for maxwell solvers based on a hyperbolic model. *Journal of Computational Physics*, 161(2):484–511, 2000.

Université de Strasbourg, Inria Tonus, AxesSim. philippe.helluy@unistra.fr

Conclusion

Dans ce chapitre (cet article), nous avons présenté les résultats de l'utilisation de la bibliothèque StarPU pour l'exécution et l'ordonnancement des tâches de calcul du solveur GD `schnaps` sur ordinateur hybride. Les tâches ainsi exécutées sont chacune codées en 2 versions : sous forme de codelets écrits en langage C à destination des cœurs CPU (notons que StarPU se réserve un cœur pour l'effort d'ordonnancement) et sous forme de codelets C-OpenCL à destination des GPU.

Les résultats sur CPU multi-cœurs se sont avérés excellents : une efficacité presque optimale quelque soit le nombre de cœurs sollicités et pour diverses tailles (divers raffinements) du problème étudié. Ces résultats ont été constatés pour 2 CPU distincts, l'un produit par AMD, l'autre par Intel. Le second CPU est de plus composé de 2 nœuds NUMA, ce qui ne semble pas impacter les performances et témoigne d'une bonne gestion des affinités entre cœurs.

Les résultats en configuration hybride sont eux aussi de bonne qualité. Dans un premier temps, nous avons facilement pu comparer la différence de performance entre les codelets CPU et les codelets GPU testés. Nous avons aussi constaté que l'ajout des cœurs CPU au GPU a permis d'améliorer les performances du GPU seul. Dans le cas des plus petites tailles de problème, l'accélération constatée est au-delà de l'accélération théorique donnée par les temps sur CPU seul et GPU seul. Ce phénomène s'explique par le fait que le CPU soulage le GPU des petites tâches qui génèrent beaucoup de temps de latence OpenCL et le GPU peut ainsi se concentrer sur les grandes tâches de calcul.

Remarquons que dans les expérimentations présentées nous utilisons un seul GPU NVidia. Rappelons alors qu'il est possible de proposer à StarPU plusieurs versions de codelet C ou C-OpenCL pour une même tâche. Cela permettrait de proposer une version optimisée AMD du codelet C-OpenCL au cas où les performances ne sont pas au rendez-vous avec les kernels optimisés sur NVidia.

Cependant, l'utilisation de la bibliothèque StarPU n'a pas été implémentée dans le solveur `teta-clac`. Le nombre de kernels (GD, modèles) impliquerait un effort de développement très important alors que ce solveur permet déjà d'effectuer des simulations équilibrées sur un grand nombre de GPU. La stratégie de découpage en zones et les communications MPI telles qu'implémentées sont éprouvées et fonctionnent parfaitement dans le cadre d'une exploitation industrielle.

Dans le dernier chapitre, nous allons revenir sur le solveur `teta-clac`

et présenter un certain nombre de résultats de validation. Nous commencerons par un cas académique de propagation d'onde plane dans un cube afin de démontrer la convergence du schéma. Nous présenterons aussi un cas d'application représentatif d'une scène de télécommunication sur lequel nous confrontons notre solveur à une implémentation de la méthode des différences finies. Enfin, nous terminerons par le cas d'application du corps humain complet à proximité duquel nous faisons rayonner une antenne Bluetooth.

Chapitre 7

Validations

Dans les chapitres précédents, nous avons présenté la méthode Galerkin Discontinue (GD) et son implémentation parallèle optimisée pour traiter des mailles hexaédriques. Nous avons aussi décrit plusieurs modèles mathématiques de simulation et différents schémas d'intégration en temps.

Dans ce dernier chapitre, nous présentons des résultats permettant de valider notre implémentation de solveur GD, autant par la précision des calculs que par leur vitesse d'exécution.

Ainsi, nous commençons par présenter des résultats de convergence sur un cas académique de propagation d'une onde plane dans un cube. Nous en profitons pour effectuer des comparaisons entre maillages structurés (hexaèdres droits) et non structurés (issus du découpage de tétraèdres).

Nous comparons ensuite notre solveur GD à un solveur implémentant la méthode des différences finies sur un cas d'application mettant en scène une antenne dipolaire à proximité de l'oreille d'une tête humaine simplifiée. Nous donnons aussi les résultats, sur ce même cas d'application, des différents schémas d'intégration en temps que nous avons présenté plus tôt. Ce cas d'application s'apparente à une communication téléphonique et a servi de fil conducteur au cours de cette thèse.

Le second fil conducteur a été l'objectif d'effectuer des simulations sur un modèle de corps humain complet (avec organes) dans le cadre du projet HOROCH. Nous présentons les résultats de la simulation effectuée sur ce modèle de corps humain, à côté duquel nous avons placé une antenne de type Bluetooth. Il s'agit de la plus grande simulation réalisée par le solveur à ce jour.

Enfin, nous terminons par présenter les résultats d'efficacité du solveur exécuté en parallèle sur un grand nombre de périphériques de calcul.

7.1 Simulation d'onde plane

7.1.1 Convergence du schéma couplé GD-RK2

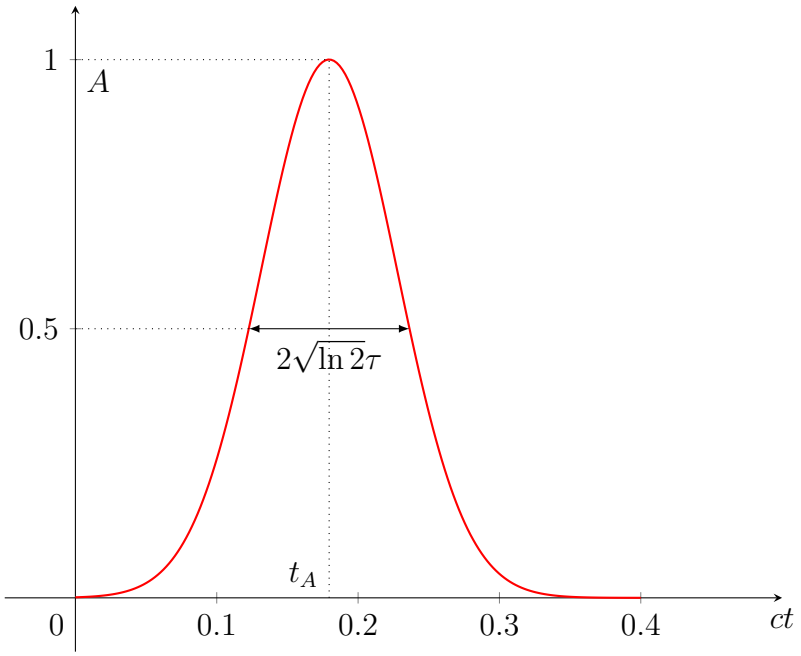
Pour déterminer l'ordre de convergence du schéma couplé GD-RK2, nous simulons la propagation d'une onde plane dans le vide, à l'aide d'une surface de Huygens (section 2.3). La condition aux limites utilisée est celle de Silver-Müller (section 2.1.4).

Une onde plane est définie par une fonction donnant l'amplitude du signal au cours du temps (proposition 5). Le signal que nous utilisons est une impulsion gaussienne définie sur l'intervalle $[0\,;T]$ par :

$$t \mapsto A \exp\left(-\left(\frac{t-t_A}{\tau}\right)^2\right), \tag{7.1}$$

avec A l'amplitude de l'onde, τ donnant l'intervalle de temps entre les deux points de demie-amplitude (de largeur $2\sqrt{\ln 2}\tau$) et t_A l'antécédent de la valeur d'amplitude maximale (figure 7.1).

FIGURE 7.1 – Signal adimensionné définissant l'onde plane utilisée au cours des tests de convergence du schéma couplé GD-RK2.



Afin de définir le signal temporel, nous utilisons la définition fréquentielle

de l'onde plane donnée par :

$$f \mapsto A\tau\sqrt{\pi}\exp\left(-(\pi f\tau)^2 - 2j\pi f t_A\right), \tag{7.2}$$

avec f la fréquence et j le nombre complexe défini tel que $j^2 = -1$. Nous pouvons alors définir notre signal en fonction de la fréquence maximale $f_{\max}$ souhaitée et retrouver le signal temporel grâce aux relations suivantes :

$$\tau = \frac{\sqrt{-\ln(a_A)}}{\pi f_{\max}}, \tag{7.3a}$$

$$t_A = \tau\sqrt{-\ln(a_0)}, \tag{7.3b}$$

où a_0 représente l'atténuation au temps initial ($t = 0$) et a_A représente l'atténuation au temps d'amplitude maximale t_A.

Ces paramètres d'atténuation permettent de réduire la durée du signal. Une impulsion gaussienne tend vers 0 aux infinis, l'atténuation au temps initial indique la tolérance à appliquer en $t = 0$ afin de diminuer t_A qui est le retard permettant d'avoir le début de la gaussienne en temps positif. L'atténuation au temps d'amplitude maximale permet de diminuer la valeur de τ (*i.e.* la durée de l'impulsion). Pour notre cas d'application nous choisissons :

- $A = 1$;
- $f_{\max} = 3$ GHz ;
- $a_0 = 0.001$ (ou 0.1 %).
- $a_A = 0.01$ (ou 1 %) ;

Avec ces valeurs, dans le cas adimensionné, nous avons $\tau \approx 0.0683$ et $t_A \approx 0.1795$, ce qui nous donne une durée de signal d'environ 1.2 ns.

La direction de propagation choisie est suivant la composante x_3. Le champ $\boldsymbol{E}$ varie selon la composante x_1 de l'espace et le champ $\boldsymbol{H}$ varie selon la composante x_2.

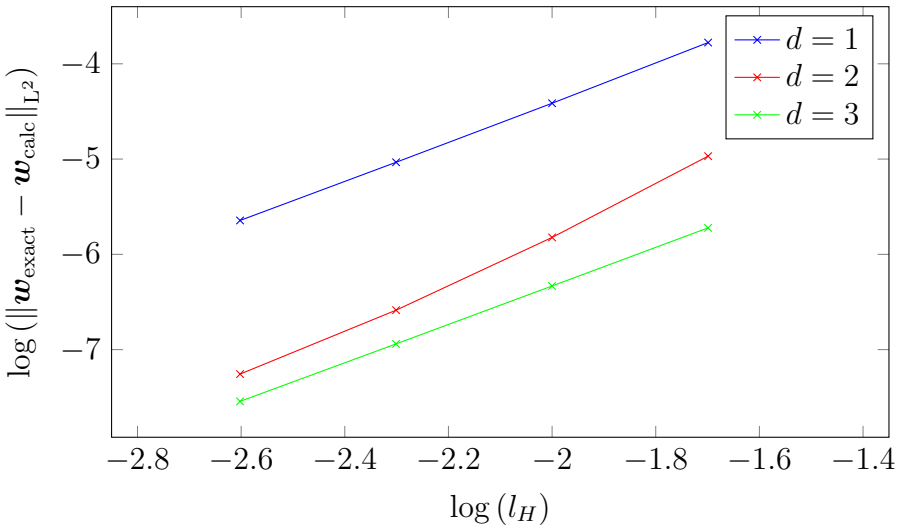
Maillage structuré

Nous considérons le domaine délimité par le cube de côté 0.1 m. Nous choisissons le temps maximal $T = 1$ ns. Ce domaine est maillé par des hexaèdres droits, tous identiques. Nous calculons les résultats de convergence pour une subdivision en 5, 10, 20 et 40 mailles (raffinements en puissances de 2) dans chaque direction pour les ordres d'interpolation 1 à 3.

Les résultats de convergence sont présentés dans la figure 7.2. Nous pouvons constater que cette méthode est d'ordre 2 en temps. Elle est même d'ordre plus élevé mais inférieur à 3 à l'ordre spatial $d = 2$.

Remarque 34. *Les résultats de convergence présentés dans les figures 7.2 et 7.7 sont là aussi ceux du schéma en temps théoriquement d'ordre 2 (remarque 31). Les résultats de convergence en espace de la méthode GD sont présentés dans la thèse de Strub [69] et sont toujours vérifiés. Nous mesurons ici seulement le taux de convergence du couplage espace-temps, qui est imposé par l'ordre de la méthode en temps.*

FIGURE 7.2 – Convergence du schéma couplé GD-RK2 appliqué aux équations de Maxwell en 3 dimensions sur maillage structuré.



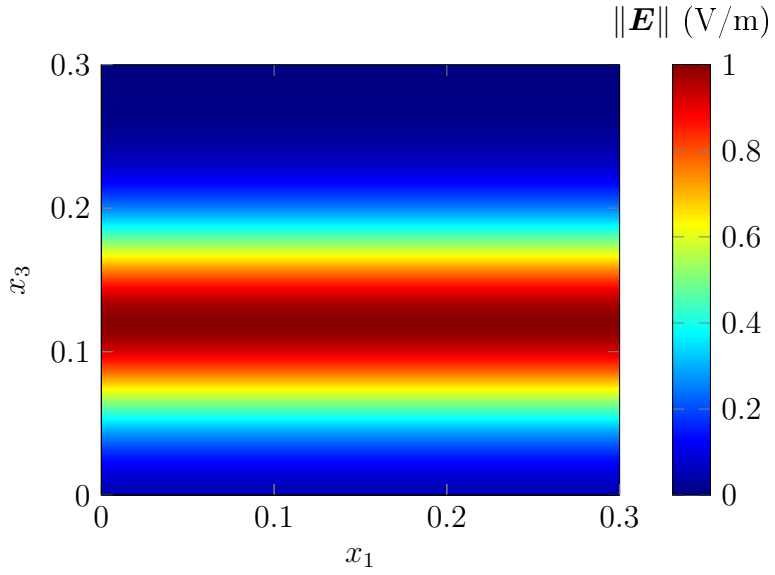
Nous mesurons aussi l'efficacité de la méthode en fonction de l'ordre d'interpolation. Pour cela nous simulons la même onde plane dans le domaine délimité par le cube de côté 0.3 m et jusqu'au temps maximal $T = 1$ ns (figure 7.3). Ce domaine est maillé par des hexaèdres droits, tous identiques. Nous utilisons la condition CFL (3.50). Nous mesurons les temps de calcul pour une subdivision en 30 mailles dans chaque direction à l'ordre 1, 15 mailles dans chaque direction à l'ordre 2 et 10 mailles dans chaque direction à l'ordre 3. Ces subdivisions génèrent des mailles dont la taille correspond à l'application de la règle empirique du « $\lambda_{\min}$ sur 10 ». Nous sommes donc placés dans un cas discrétisé de manière équivalente pour chaque ordre d'interpolation évalué. Nous notons $T_{\text{s}}(d)$ le temps de simulation à l'ordre d'interpolation $d > 0$ pour ce cas d'application.

Définition 7. *Dans les conditions décrites ci-avant, l'**efficacité en fonction de l'ordre** d'interpolation spatiale $d > 0$ du schéma couplé GD-RK2*

est donnée par le rapport :

$$E(d) = \frac{T_{\mathrm{s}}(1)}{T_{\mathrm{s}}(d)}. \tag{7.4}$$

FIGURE 7.3 – Plan de coupe (x_1Ox_3) en $x_2 = 0.15$ m de la norme du champ électrique de la solution exacte de l'onde plane (7.1) simulée, au temps final $T = 1$ ns.



Les résultats d'efficacité sont présentés dans le tableau 7.4. Nous pouvons constater qu'un ordre élevé est plus efficace et plus précis. Cette différence d'efficacité est due au fait que le pas de temps est proportionnel à la taille des mailles (plus grandes en ordre élevé) et à la décroissance cubique de leur nombre avec la croissance de l'ordre. Ces deux facteurs surcompensent l'augmentation de la complexité des calculs.

Maillage non structuré

Nous effectuons aussi les calculs de convergence du schéma couplé GD-RK2 sur maillage non structuré. La convergence de la méthode GD appliquée aux systèmes hyperboliques de lois de conservation sur maillages non structurés a initialement été démontrée par Szepessy *et al.* [41].

Nous maillons le même domaine délimité par le cube de côté 0.1 m en tétraèdres à l'aide du mailleur MeshGems commercialisé par Distene. Ce domaine est maillé par des tétraèdres dont nous imposons la taille l_T des arêtes à

FIGURE 7.4 – Efficacité du schéma couplé GD-RK2 en fonction de l'ordre d'interpolation spatiale.

d	$\#\mathcal{M}$	T_s (en s)	$E(d)$	Erreur L^2
1	30^3	11.6	1	$2.35 \cdot 10^{-4}$
2	15^3	2.06	5.63	$5.45 \cdot 10^{-5}$
3	10^3	1.30	8.92	$3.79 \cdot 10^{-5}$

0.04, 0.02, 0.01 et 0.005 m, et qui sont ensuite découpés en hexaèdres (figures 4.3 et 7.5) ayant respectivement une taille d'arête maximale, notée $\max l_H$, de 0.02, 0.01, 0.005 et 0.0025 m. Ces tailles d'arête maximale correspondent à celles obtenues avec les subdivisions utilisées en maillage structuré, à savoir des subdivisions respectives en 5, 10, 20 et 40 mailles par côté. Nous choisissons aussi le même temps maximal $T = 1$ ns.

Le nombre de mailles contenues dans ces maillages non structurés est généralement environ 4 fois plus important que celui des maillages structurés correspondants. De plus, la taille d'arête minimale est 4 à 6 fois inférieure à la taille d'arête maximale, facteur impactant directement le pas de temps et donc le nombre d'itérations de la méthode. Nous pouvons donc estimer un temps de calcul en moyenne 23 fois plus long sur maillage non structuré issu de tétraèdres découpés, par rapport à un maillage structuré équivalent (tableau 7.6).

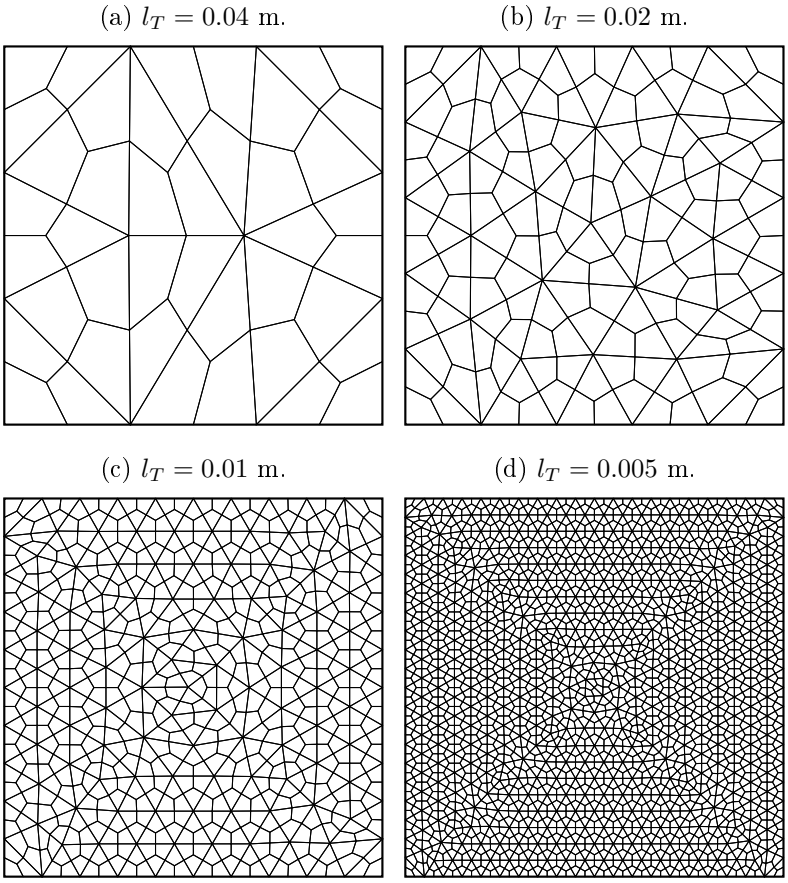
Les résultats de convergence sont présentés dans la figure 7.7 pour les ordres d'interpolation 1 à 3. Nous les présentons en fonction de la taille l_T des arêtes des tétraèdres, proportionnelle à la taille maximale $\max l_H$ des arêtes des hexaèdres, puisque ce paramètre est l'indicateur de discrétisation du maillage. Nous les présentons aussi en fonction de la taille minimale $\min l_H$ des arêtes des hexaèdres.

Ces résultats sont moins réguliers que sur maillage structuré (probablement) car nous ne sommes pas dans le cas de raffinements d'un même maillage initial. Nous constatons un ordre de convergence moyen d'environ 1.5 à l'ordre spatial $d = 1$ et supérieur à 2 (mais toujours inférieur à 3) pour les ordres $d = 2$ et $d = 3$.

7.2 Tête humaine simplifiée

Nous comparons le solveur `teta-clac` (Teta) au solveur Temsi-FD [30] (Temsi) qui implémente la méthode des différences finies dans le domaine

FIGURE 7.5 – Faces x_2Ox_3 en $x_1 = 0$ des maillages non structurés utilisés pour les différentes tailles de maille.



temporel (FDTD pour « *Finite Differences in Time Domain* ») et développé à l'institut de recherche Xlim de Limoges. Cette comparaison permet de confronter deux méthodes de calcul implémentées par des équipes différentes et sur des architectures distinctes car cet autre solveur utilise OpenMP (sur CPU).

Pour cela, nous comparons les résultats d'une simulation mettant en scène une tête humaine simplifiée à proximité de laquelle nous avons placé une antenne dipôle planaire de type ELPOSD (pour « *End-Loaded Planar Open-Sleeve Dipole* ») [67] conçue pour rayonner à une fréquence de 1 GHz. La particularité de cette antenne est que la source (la zone d'injection du signal) est un carré de côté 1 mm. Cette forte contrainte géométrique diminue considérablement le pas de temps de simulation.

FIGURE 7.6 – Comparaison des maillages structurés et non structurés utilisés dans le calcul de l'ordre de convergence du schéma couplé GD-RK2.

(a) Comparaison en nombre de mailles.

Structuré		**Non structuré**		**Facteur $\#\mathcal{M}$**
Subdiv	$\#\mathcal{M}$	l_T	$\#\mathcal{M}$	
5	125	0.04	508	4.064
10	1k	0.02	4352	4.352
20	8k	0.01	37408	4.676
40	64k	0.005	304916	4.764

(b) Comparaison en taille d'arêtes.

Structuré		**Non structuré**			Inverse du
Subdiv	l_H	l_T	$\max l_H$	$\min l_H$	**Facteur (min)l_H**
5	0.02	0.04	0.02	0.004373	4.574
10	0.01	0.02	0.01	0.002189	4.568
20	0.005	0.01	0.005	0.000829	6.031
40	0.0025	0.005	0.0025	0.000419	5.967

7.2.1 Description

Composition de la scène

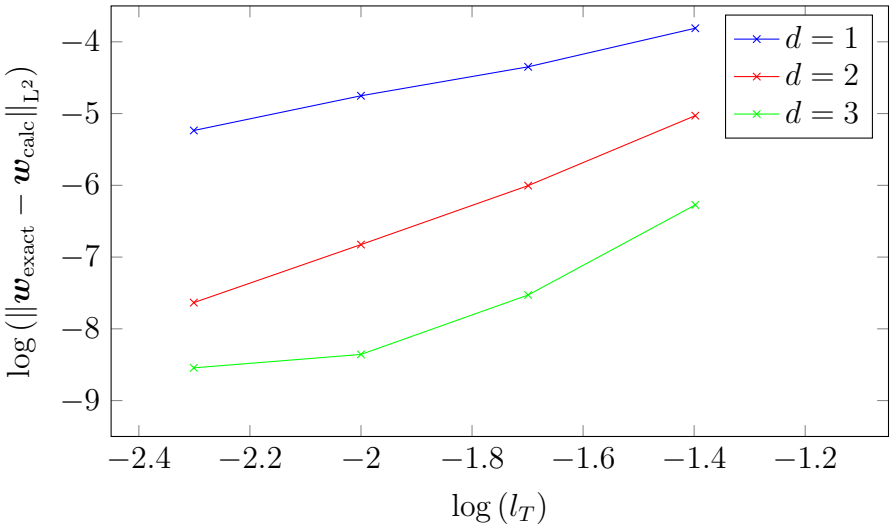
Cette simulation s'apparente au rayonnement d'une antenne de téléphone mobile au cours d'une communication téléphonique. L'antenne est placée à proximité de l'oreille de la tête simplifiée et rayonne en direction du cerveau.

Dans notre cas d'étude, l'antenne est orientée dans le sens de la longueur sur l'axe (Ox_3), dans le sens de la largeur sur l'axe (Ox_1) et rayonne dans la direction de l'axe (Ox_2). La tête simplifiée est placée de manière à ce que l'antenne rayonne au niveau de son oreille droite, à 3.5 cm de celle-ci. Le centre de l'antenne se situe en $(0, 0.113175, 0.12)$. La tête simplifiée contient un cerveau simplifié délimité par un ellipsoïde.

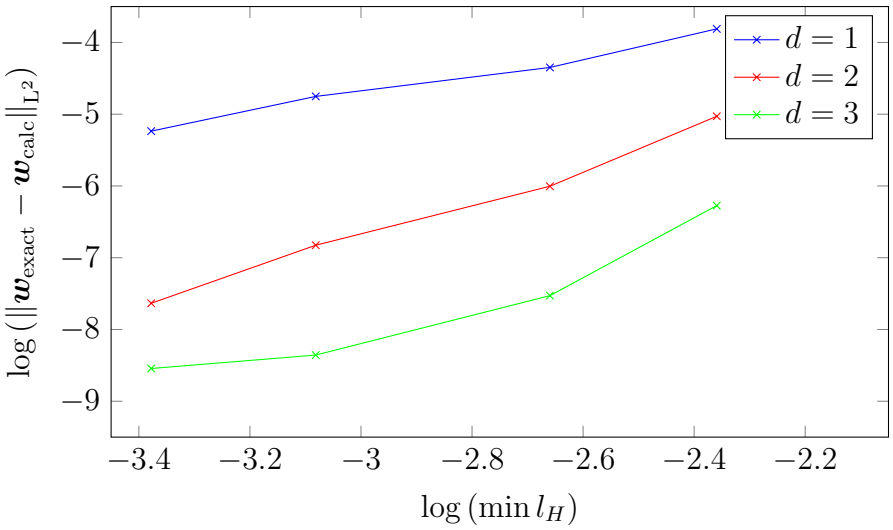
Le domaine de calcul pour toutes les simulations de ce cas d'étude est un pavé de dimensions $[-0.2\,;0.2] \times [-0.2\,;0.2] \times [-0.1\,;0.35]$ (en mètres), avant adjonction de couches de PML (section 2.1.5).

FIGURE 7.7 – Convergence du schéma couplé GD-RK2 appliqué aux équations de Maxwell en 3 dimensions sur maillage non structuré.

(a) Convergence en fonction de la taille l_T.



(b) Convergence en fonction de la taille $\min l_H$.

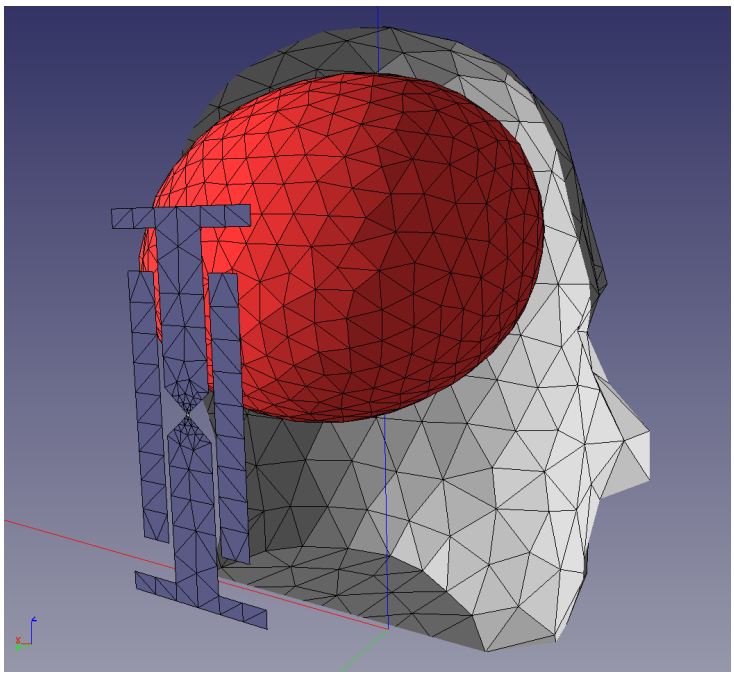


Matériaux

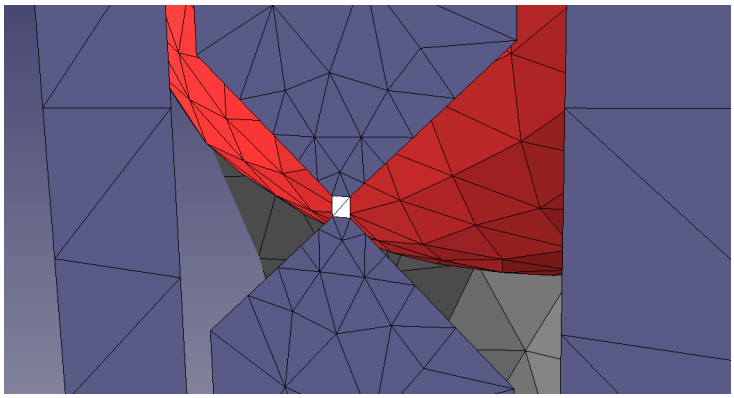
Nous appliquons la condition PEC (section 2.1.1) sur toute la surface de l'antenne, hormis la source. La source nous permet d'injecter le signal dans la scène.

FIGURE 7.8 – Maillage en surfaces de l'antenne ELPOSD (en gris sombre) placée à proximité de la tête simplifiée. Scène vue depuis le côté positif de l'axe (Ox_2). Une moitié de la surface délimitant la tête (en gris clair) est cachée pour laisser apparaître le cerveau (en rouge).

(a) Scène complète.



(b) Zoom sur la zone d'injection du signal (en blanc).



La tête contient des matériaux diélectriques dont les propriétés sont données dans le tableau 7.9. Nous utilisons les propriétés de la matière grise

dans le cerveau simplifié et celles de la matière osseuse dans le reste de la tête simplifiée [26]. Les propriétés électromagnétiques de ces matériaux sont globalement constantes sur la bande de fréquences utilisée (0.1 à 10 GHz).

FIGURE 7.9 – Propriétés électromagnétiques des matériaux utilisés dans la scène de la tête simplifiée.

Matériau	ε_r	μ_r	σ (S/m)	$\sigma^\star$
cerveau	51.8	1	1.521	0
os	11.41	1	0.43	0
vide	1	1	0	0

Signal source

Dans le cas de la méthode FDTD, la zone d'injection du signal, représentée par un carré de côté 1 mm situé au centre de l'antenne, est remplacée par un fil d'excitation de rayon 0.125 mm. Ce fil implémente un modèle dérivé de celui de Holland *et al.* [39] et qui peut s'appliquer aux fils obliques [31].

Dans le cas de notre solveur GD, nous utilisons le modèle de générateur de Thévenin (section 2.3.2) sur la zone d'injection. Pour les deux méthodes, nous appliquons le modèle d'excitation avec une impédance réelle de 50 Ohms.

Le signal que nous utilisons pour les deux méthodes est une impulsion gaussienne modulée définie sur l'intervalle $[0\,;T]$ par :

$$g : t \mapsto A \sin\left(2\pi f_0 \left(t - t_A\right)\right) \exp\left(-\left(\frac{t - t_A}{\tau}\right)^2\right), \tag{7.5}$$

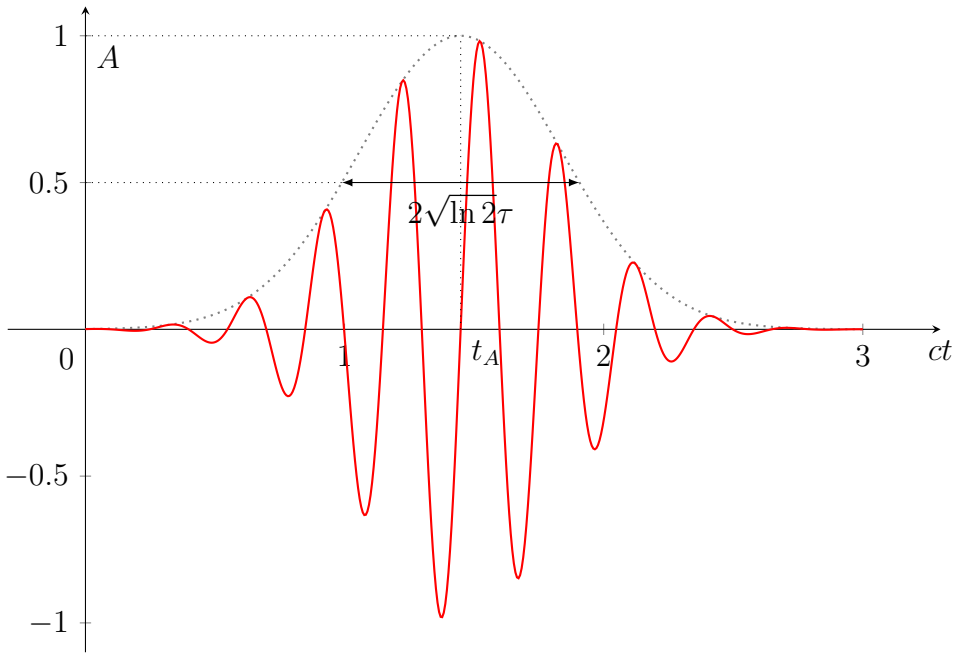
avec A l'amplitude de l'onde, f_0 la fréquence centrale du signal, τ donnant l'intervalle de temps entre les deux points de demie-amplitude (de largeur $2\sqrt{\ln 2}\tau$) de l'impulsion gaussienne non modulée et t_A l'antécédent de la valeur d'amplitude maximale (figure 7.10). Ce signal modulé est le produit d'une impulsion gaussienne (7.1) avec une sinusoïde.

Dans le cas de notre solveur GD, ce signal est injecté sur toute la zone d'injection par le champ électrique incident (section 2.3.2) :

$$\boldsymbol{E}_I = \frac{g(t)}{50}\frac{\overrightarrow{Ox_3}}{0.001} = 20 g(t) \overrightarrow{Ox_3} \tag{7.6}$$

avec une impédance de surface de 50 Ohm définie sur cette même surface. Le champ magnétique incident est nul.

FIGURE 7.10 – Signal adimensionné de la tension générée, utilisé au cours des simulations avec l'antenne ELPOSD.



Les paramètres τ et t_A sont donnés par les relations suivantes :

$$\tau = 2\frac{\sqrt{-\ln(a_A)}}{\pi\Delta_f}, \tag{7.7a}$$

$$t_A = \tau\sqrt{-\ln(a_0)}, \tag{7.7b}$$

où a_0 représente l'atténuation au temps initial ($t = 0$), a_A représente l'atténuation au temps d'amplitude maximale t_A et Δ_f représente la largeur de la bande de fréquences du signal considéré, centrée en f_0.

Pour notre cas d'application nous choisissons :

— $A = 1$;
— $f_0 = 1$ GHz ;
— $\Delta_f = 0.6$ GHz ;
— $a_0 = 0.001$ (ou 0.1 %).
— $a_A = 0.05$ (ou 5 %) ;

Avec ces valeurs, dans le cas adimensionné, nous avons $\tau \approx 0.5509$ et $t_A \approx 1.448$, ce qui nous donne une durée de signal d'environ 10 ns.

Maillage

La fréquence maximale considérée est de $f_{\max} = f_0 + \Delta_f/2 = 1.3$ GHz. Le tableau 7.11 présente la distance maximale entre deux points d'interpolation selon la règle empirique du « $\lambda_{\min}$ sur 10 » en fonction des propriétés électromagnétiques des matériaux à la fréquence $f_{\max} = 1.3$ GHz (tableau 7.9).

FIGURE 7.11 – Distances maximales entre deux points d'interpolation en fonction des matériaux à la fréquence 1.3 GHz.

Matériau	Δx (mm)	l_T, $d = 2$ (mm)
cerveau	3.2	12.8
os	6.8	27.2
vide	23	92

Pour obtenir la taille d'arête maximale d'un hexaèdre à partir de ces valeurs, nous devons les multiplier par l'ordre d'interpolation spatiale choisi. Dans le cas d'un maillage en tétraèdres, nous devons considérer le double des arêtes des hexaèdres, afin d'appliquer le découpage présenté en figure 4.3.

Pour une simulation à l'ordre d'interpolation spatiale $d = 2$, nous maillons donc la scène avec les longueurs d'arête de tétraèdre l_T données dans la dernière colonne du tableau 7.11. Pour cela, nous commençons par mailler les surfaces avec la plus petite taille d'arête à appliquer sur les volumes voisins (12.8 mm pour la peau du cerveau et 27.2 pour la peau de la tête), puis nous maillons les volumes.

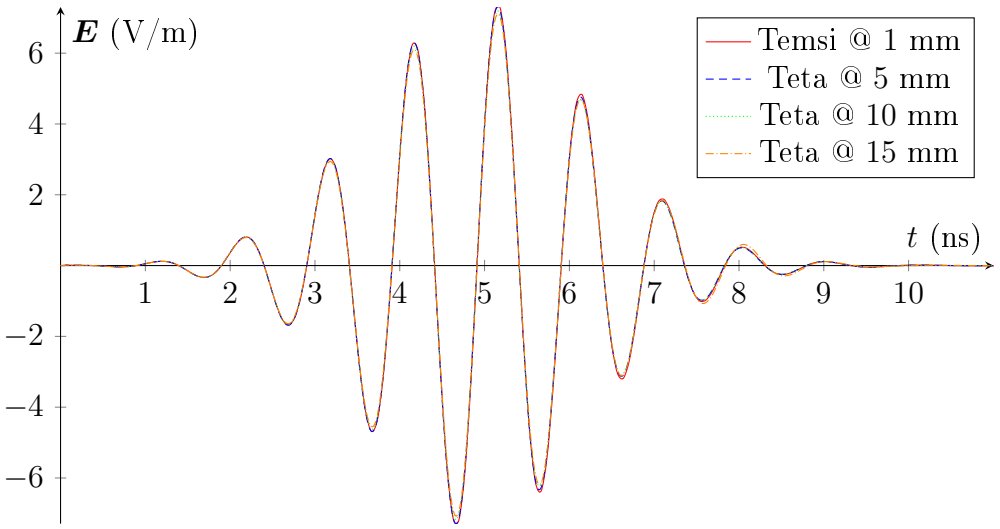
L'antenne ELPOSD a préalablement été maillée avec une taille d'arête maximale de 10 mm afin d'assurer un rayonnement de bonne qualité. Les champs électromagnétiques obtenus avec cette finesse de maillage correspondent à ceux obtenus sur une version maillée à une taille d'arête maximale de 5 mm (figure 7.12) ainsi que ceux obtenus dans le cas de la méthode FDTD. Dans le cas de l'antenne maillée à une taille d'arête maximale de 15 mm, nous observons une dégradation des résultats en fin de signal (figure 7.12b).

Après cette phase de maillage, nous obtenons un ensemble de 19413 tétraèdres. En résultent 77652 hexaèdres quelconques après le découpage des tétraèdres, auxquels s'ajoutent 4570 hexaèdres droits en bordure de domaine pour l'application du modèle PML. Les surfaces présentées dans la figure 7.8 sont issues de ce maillage.

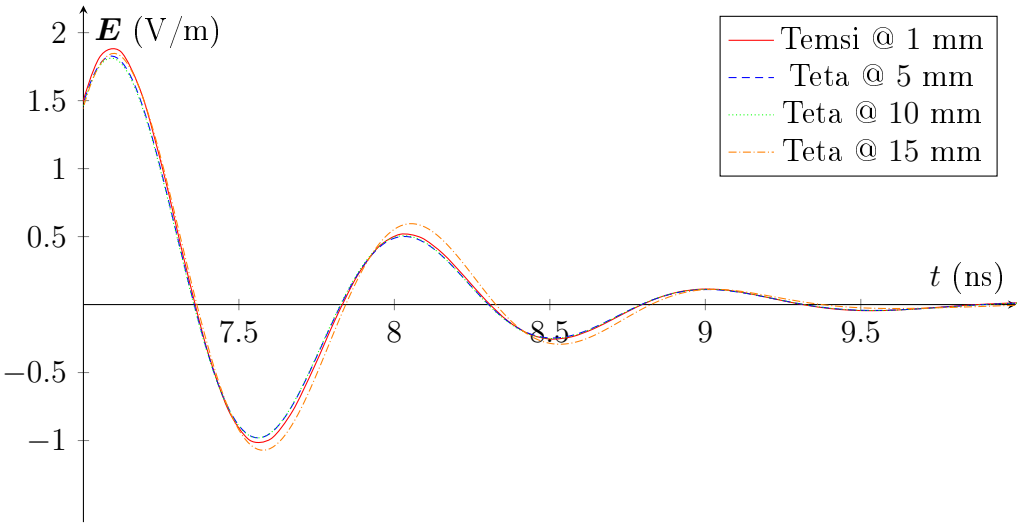
Dans le cas de la méthode FDTD, la zone d'injection du signal, représentée par un carré de côté 1 mm situé au centre de l'antenne, impose le pas

FIGURE 7.12 – Composante x_3 du champ $\boldsymbol{E}$ au point $(0, 0.095675, 0.12)$ (à mi-chemin de la distance antenne-oreille) dans une configuration « antenne seule ». Comparaison à la méthode FDTD pour différents raffinements de l'antenne dans le cas du schéma couplé GD-RK2.

(a) Signal complet.



(b) Agrandissement de la zone d'intérêt.



de discrétisation. Cette méthode de calcul nécessite de discrétiser la scène en mailles hexaédriques sous la forme d'une grille structurée. Le maillage résultant est composé d'environ 77.5 millions de mailles ($417 \times 428 \times 434$). Le

modèle PML ajoute environ 12 millions de mailles en bordure de domaine. Nous avons donc une grille structurée composée d'un peu moins de 90 millions de mailles de côté 1 mm. Soit un nombre de mailles 1087 fois plus important que pour la méthode GD.

7.2.2 Comparaison à la méthode FDTD

Mémoire

Du point de vue de la mémoire, la simulation GD occupe 380 Mo sur le GPU (avec environ 100 Mo supplémentaires pour l'allocation des instances OpenCL) contre environ 9 Go en RAM dans le cas de la simulation FDTD, soit 23 fois moins de mémoire utile pour la méthode GD.

Cette différence d'occupation en mémoire est principalement liée à la quantité de mailles et aux degrés de liberté par maille. Dans le cas de la méthode GD, la nature non structurée du maillage nécessite aussi de stocker une grande quantité de coordonnées et d'informations de connectivité entre les mailles.

Performances

En temps de calcul, la simulation FDTD sur grille structurée a nécessité 4.85 heures pour simuler 20 ns de temps physique sur 22 cœurs (logiques) d'un processeur Intel Dual Xeon E5645 (2×6 cœurs physiques hyperthreadés) cadencé à 2.40 GHz. Le temps de calcul sur 46 cœurs (logiques) du plus récent processeur Intel Dual Xeon E5-2650 v4 (2×12 cœurs physiques hyperthreadés) cadencé à 2.20 GHz est de 2.24 heures. Le coefficient CFL imposé par la largeur du fil d'excitation est de 0.92 et le pas de temps de la simulation est calculé à $1.772 \cdot 10^{-12}$ s par la condition CFL classique utilisée avec la méthode FDTD.

Dans le cas de la méthode GD sur maillage non structuré à l'ordre spatial $d = 2$, la simulation a nécessité 1.88 heures de calcul sur un GPU NVidia GeForce GTX 1070 cadencé à 1.721 GHz et 1.19 heures de calcul sur un GPU NVidia GeForce GTX 1080 Ti cadencé à 1.582 GHz.

L'application du diagnostic de la puissance itérée (section 3.3.4) nous a permis de constater que dans le cas ce cette simulation, la condition CFL de stabilité (3.50) sous-évalue le pas de temps de stabilité. Nous avons fixé le coefficient CFL à 3.1 et utilisé la formule (3.50) pour le calcul du pas de temps de la simulation. Le pas de temps ainsi obtenu est de $6.641 \cdot 10^{-14}$ s,

soit 27 fois plus petit que celui utilisé dans le cas de la méthode FDTD.

Les performances des deux solveurs peuvent donc être considérées du même ordre de grandeur sur ce cas d'application. Bien que la méthode GD soit présentée comme étant plus rapide que la méthode FDTD, le temps de restitution est fonction du support de calcul utilisé. Les temps de calculs obtenus sont résumés dans le tableau 7.13.

La méthode GD est très bien adaptée dans ce cas d'application compte tenu du facteur d'échelle (proche de 100) entre la plus petite et la plus grande maille du maillage. La méthode FDTD quant à elle, est directement affectée par la plus petite maille qui impose le pas spatial de la grille structurée.

FIGURE 7.13 – Temps de calcul des méthodes GD et FDTD en fonction du support d'exécution dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée.

Solveur	Support d'exécution	CFL	Temps (h)
Temsi	Dual Xeon E5645, 22 threads	0.92	4.85
	Dual Xeon E5-2650 v4, 46 threads	0.92	**2.24**
Teta	GeForce GTX 1070	3.1	1.88
	GeForce GTX 1080 Ti	3.1	**1.19**

Résultats

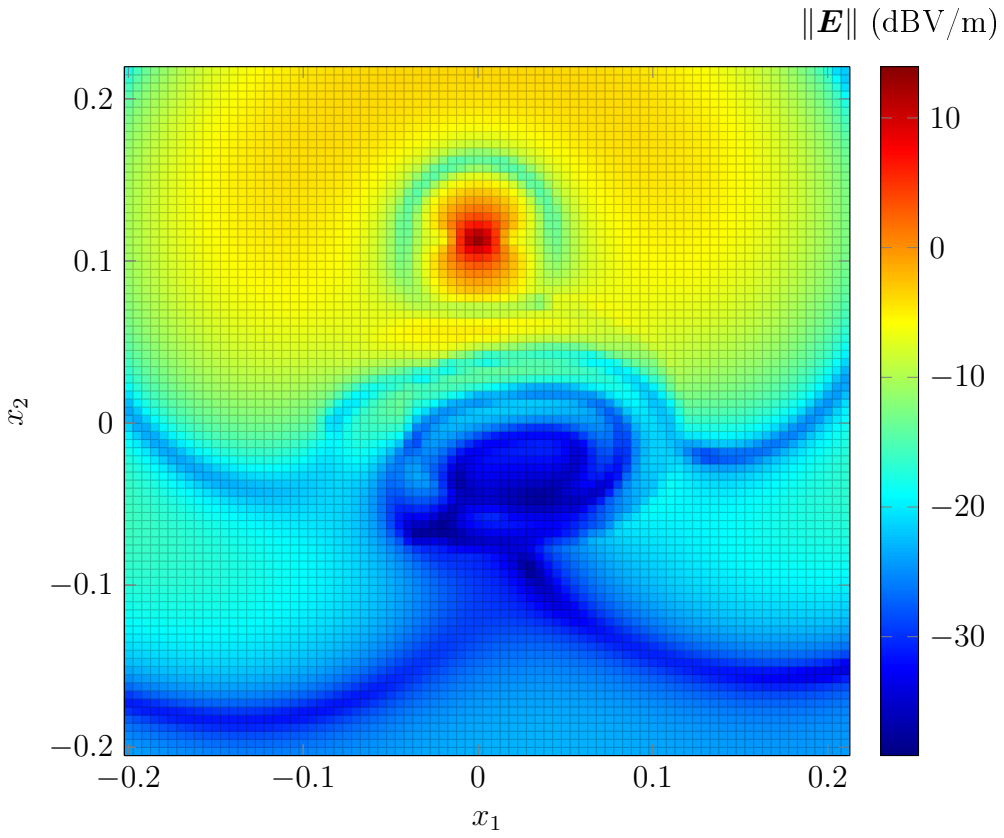
La figure 7.14 présente un plan de coupe des résultats de la simulation par la méthode GD.

Ces résultats sont très proches de ceux obtenus avec la méthode FDTD. Afin de faciliter la comparaison, nous présentons aussi la valeur de la composante x_3 du champ électrique mesuré en différents points d'observation (figure 7.15) :

- en $O_1 = (0, 0.095675, 0.12)$, à mi-chemin entre l'antenne et la tête ;
- en $O_2 = (0, 0, 0.12)$, à l'intérieur du cerveau, à hauteur de la zone source de l'antenne ;
- en $O_3 = (0, -0.113175, 0.12)$, le symétrique de la zone source de l'antenne par rapport au point O_2.

Nous constatons que les deux solveurs produisent sensiblement les mêmes résultats. Un léger décalage peut être constaté dans le cas des mesures effectuées à l'intérieur du cerveau et de l'autre côté de la tête par rapport à l'antenne. Entre la tête et l'antenne, les résultats sont identiques.

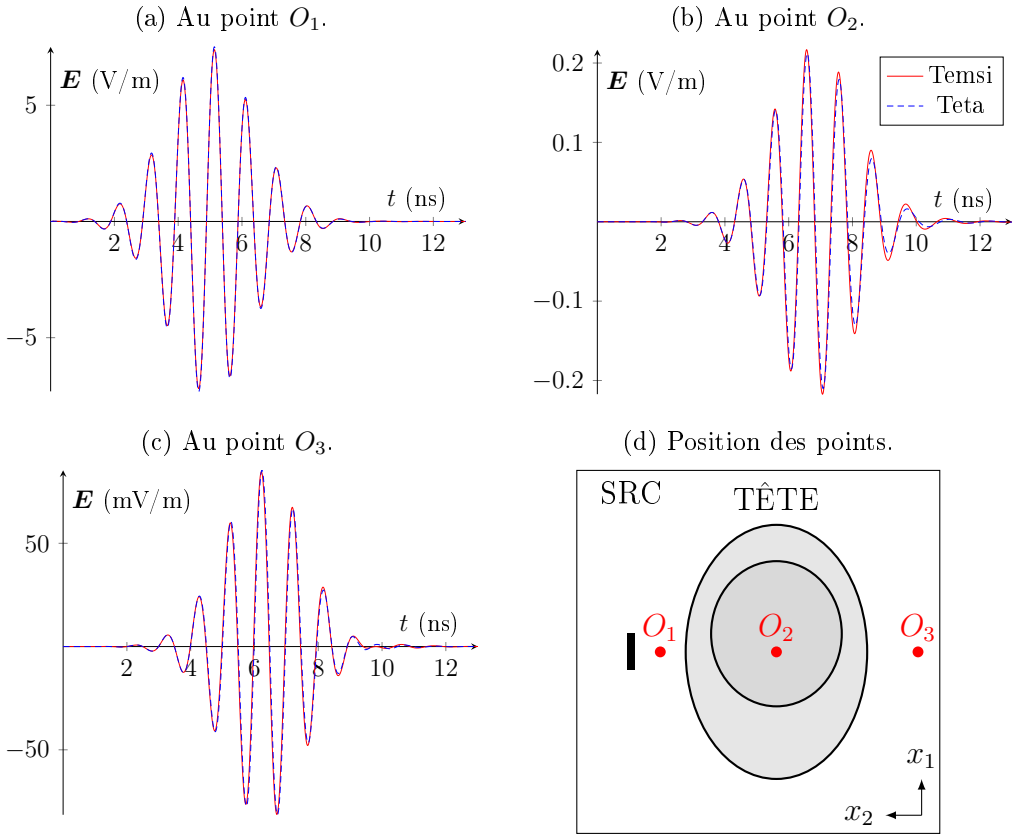
FIGURE 7.14 – Plan de coupe (x_1Ox_2) en $x_3 = 0.12$ m du logarithme (10dB) de la norme du champ électrique de la solution calculée dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée au temps $t = 3$ ns. Discrétisation à 5 mm. La zone rouge correspond au centre de l'antenne, la zone bleue foncée correspond à la tête simplifiée.



Nous pouvons aussi remarquer que l'onde électromagnétique atteint le côté opposé de la tête avant le centre du cerveau. Ces résultats sont dus à la forte permittivité des matériaux de la tête qui diminuent la vitesse de propagation des ondes. Cette propriété des matériaux génère le phénomène d'« enroulement » des ondes constaté au niveau de la zone du cerveau sur la figure 7.14.

Nous avons évoqué d'autres axes d'optimisation dans les chapitres précédents. Du point de vue de la discrétisation spatiale, nous pouvons réduire la masse de calculs en appliquant l'ordre spatial adaptatif. Nous avons aussi décrit une méthode d'intégration avec application d'un pas de temps local

FIGURE 7.15 – Valeur de la composante x_3 du champ électrique mesuré en différents points d'observation dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée. Comparaison de la méthode FDTD au schéma couplé GD-RK2.



par maille. Les résultats et performances obtenus avec ces deux méthodes sont présentés dans les sections suivantes.

7.2.3 Application de l'ordre spatial adaptatif

Le découpage des tétraèdres génère des hexaèdres dont la taille $\min l_H$ de la plus petite arête peut être jusqu'à 6 fois inférieure à la taille donnée par l'application de la règle du « $\lambda_{\min}$ sur 10 » (tableau 7.6b). Nous pouvons donc appliquer un ordre d'interpolation spatiale inférieur sur ces mailles (section 4.4) tout en conservant un raffinement suffisant du point de vue de l'évaluation du pas de temps de simulation (application du critère CFL (3.50) sur la plus petite arête de chaque maille devant respecter la règle

du « λ_{min} sur 10 »).

Rappelons que nous imposons un ordre spatial minimal à $d = 1$. Ainsi, la plupart des mailles de notre simulation passent à cet ordre d'interpolation. Seuls les hexaèdres droits générés en bordure de domaine pour l'application du modèle PML ont une taille suffisante pour conserver un ordre spatial 2. Du point de vue de la discrétisation spatiale (théorème de Shannon sur l'échantillonnage des signaux), le passage de l'ordre 2 à l'ordre 1 implique une discrétisation proche de « λ_{min} sur 5 » à proximité des arêtes les plus longues. Cet échantillonnage est suffisant pour représenter fidèlement les ondes.

Le maillage ainsi que les paramètres de simulation (hormis les ordres d'interpolation) sont identiques à ceux présentés dans la section 7.2.1. Nous obtenons donc une simulation composée de 77652 hexaèdres quelconques spatialement interpolés à l'ordre 1 et 4570 hexaèdres droits en bordure de domaine pour l'application du modèle PML interpolés à l'ordre 2.

Performances

Cette simulation a duré 0.31 heures sur NVidia GeForce GTX 1080 Ti avec l'application du coefficient CFL 4.5 déterminé par la méthode de la puissance itérée pour l'ordre d'interpolation spatiale $d = 1$. Les mailles du modèle PML sont suffisamment grandes pour être stables avec ce coefficient CFL. Le pas de temps de la simulation est de $1.446 \cdot 10^{-13}$ s.

Une accélération d'un facteur 3.8 par rapport à la même simulation interpolant spatialement à l'ordre $d = 2$ sur toutes les mailles est donc constatée.

Résultats

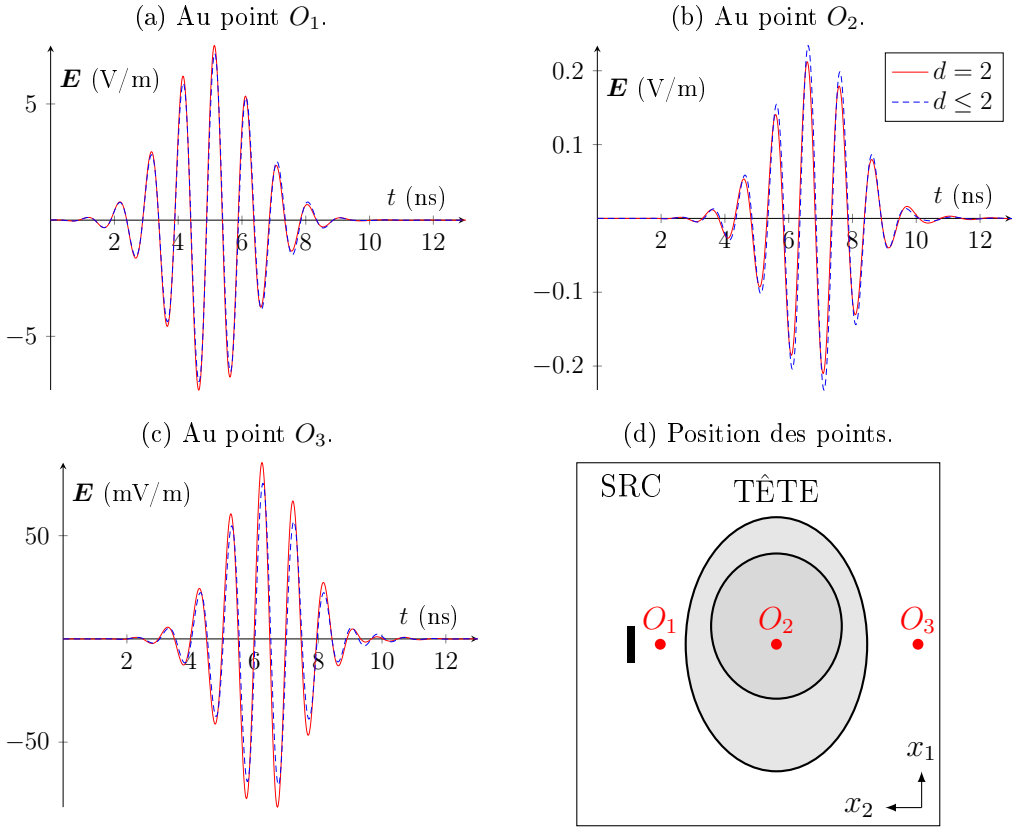
Les résultats sont proches de ceux obtenus avec une interpolation homogène à l'ordre 2. Nous constatons cependant une légère perte dans l'amplitude du signal (-5 %), dès son injection via le modèle de générateur de Thévenin (section 2.3.2).

L'amplitude du signal au niveau du point de mesure O_3, situé de l'autre côté de la tête, est plus faible que celle mesurée pour l'ordre 2. Ceci témoigne du fait que l'interpolation spatiale à l'ordre 1 est plus dissipative qu'une interpolation à l'ordre 2. En effet, nous passons de fonctions de base quadratiques à des fonctions linéaires, l'approximation est donc moins précise. L'amplitude du signal au niveau du point de mesure O_2, situé dans le cerveau, est quant à elle plus importante que celle mesurée pour l'ordre 2.

Toutefois, mises à part les différences d'amplitude, la forme du signal est fidèle au signal mesuré pour une interpolation à l'ordre 2. La perte de précision est naturellement induite par l'utilisation d'un ordre d'interpolation

inférieur, et spécialement dans le cas de l'ordre 1 qui linéarise (par morceaux) l'approximation de la solution. Il est donc préférable d'appliquer l'ordre adaptatif dans le cas d'une interpolation initiale d'ordre plus élevé.

FIGURE 7.16 – Valeur de la composante x_3 du champ électrique mesuré en différents points d'observation dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée. Comparaison du schéma couplé GD-RK2 pour $d = 2$ au même schéma avec application de l'ordre d'interpolation spatiale adaptatif ($d \leq 2$).



7.2.4 Application du schéma LRK2

Avant d'appliquer la méthode à pas de temps local, nous commençons par appliquer le schéma LRK2 décrit dans la section 5.2. Rappelons qu'il s'agit du schéma RK2 modifié pour effectuer une prédiction locale à chaque maille. L'étape de mise à jour reste inchangée. Le pas de temps de simulation est

homogène sur tout le maillage et nous reprenons la condition CFL (3.50).

Le coefficient CFL de ce schéma, déterminé par la méthode de la puissance itérée pour l'ordre d'interpolation spatiale $d = 2$, est de 1.9. La stabilité de ce schéma est donc inférieure au schéma RK2. Le pas de temps de la simulation est de $4.071 \cdot 10^{-14}$ s et le temps de calcul sur NVidia GeForce GTX 1080 Ti est de 1.71 heures.

Les résultats avec l'application de ce schéma sont équivalents à ceux obtenus avec le schéma RK2, malgré l'utilisation d'un prédicteur local. Cependant, cette méthode est 1.44 fois moins efficace compte tenu de son coefficient CFL de stabilité. Remarquons tout de même qu'à pas de temps égal, ce schéma serait plus efficace que le schéma RK2 de 12 %, cela grâce aux propriétés locales du terme de flux de l'étape de prédiction.

7.2.5 Application du schéma LTS2

Contrairement à une méthode à pas de temps homogène telle que les méthodes de type RK (section 3.3.1), le schéma LTS2 impose de vérifier un critère de type CFL sur chaque maille. Dans le cas d'une méthode à pas de temps homogène, il suffit de satisfaire la condition CFL sur la maille la plus contraignante (généralement une seule maille : la plus petite, ou dans certains cas une maille très déformée).

Une condition CFL de stabilité offrant les meilleurs performances est donc difficile à déterminer. En effet, dans le cas d'une une grande maille (déformée) sur laquelle le schéma en temps homogène était stable avec le pas de temps imposé par la plus petite maille du maillage, des instabilités peuvent apparaître lors de l'utilisation d'un pas de temps (calculé avec la même condition CFL) propre à chaque maille.

Nous présentons donc des résultats pour les coefficients CFL de stabilité maximaux dans le cas de l'application des critères CFL (3.50) et (3.51). Nous noterons respectivement ces critères CFL1 et CFL2. Pour le second critère, nous avons testé différentes valeurs pour l'exposant α. Ce paramètre permet de faire varier le gradient de la fonction qui associe à chaque maille son pas de temps et impacte directement la répartition des classes de pas de temps qui utilise la fonction $\log_2$.

Nous utilisons le schéma LTS2 sur le cas d'application de l'antenne dipolaire placée à proximité d'une tête humaine simplifiée (section 7.2.1). Nous choisissons l'ordre d'interpolation spatiale $d = 1$ et remplaçons la condition de bord PML par la condition de Silver-Müller (section 2.1.4). Dans ces conditions, la simulation à l'aide du schéma couplé GD-RK2 sur 20 ns de temps

physique a nécessité 0.44 heures de calcul sur un GPU NVidia GeForce GTX 1070. Le pas de temps de cette simulation est de $1.446 \cdot 10^{-13}$ s déterminé à l'aide du critère CFL (3.50). Le coefficient CFL de stabilité déterminé par la méthode de la puissance itérée (section 3.3.4) est de 4.5.

Mémoire

Du point de vue de la mémoire, la simulation GD-RK2 occupe 142 Mo sur le GPU contre en moyenne 155 Mo dans le cas des simulations GD-LTS2, soit environ 10 % de plus de mémoire utile pour le schéma LTS2.

Cette faible différence d'occupation en mémoire est principalement liée à la génération d'interfaces supplémentaires entre les classes de pas de temps et les zones de mailles tampon. En effet, l'implémentation actuelle de ce schéma utilise le découpage en zones homogènes pour séparer les classes de pas de temps et les mailles tampon.

Performances

Les performances en dimension 3 sont nettement inférieures à celles constatées en dimension 1 (section 5.4). En effet, en dimension 1, une interface (terme de flux) entre 2 classes de pas de temps ne représente qu'un seul degré de liberté contre des milliers en dimension 3. Ces interfaces sont des surfaces fermées topologiquement comparables à des frontières de boules.

Dans le cas de la simulation à pas de temps homogène, toutes les mailles sont regroupées en une unique zone homogène. Les kernels de calcul sont donc peu nombreux et traitent un grand nombre de mailles. Dans le cas des simulations à pas de temps local, nous avons entre 6 et 16 zones homogènes supplémentaires, issues du découpage en classes de pas de temps. Ce découpage multiplie d'autant le nombre de kernels exécutés et génère une interface entre chacune de ces zones (agencement en « peau d'oignon »). Ces zones et interfaces comptent un nombre de mailles réduit, les kernels sont donc moins efficaces, en plus d'augmenter le temps de latence OpenCL par leur nombre.

Néanmoins, malgré tous ces facteurs diminuant l'efficacité en OpenCL et dimension 3, nous avons obtenu de meilleures performances en temps de calcul à l'aide du schéma LTS2. Le tableau 7.17 présente les résultats en temps de calcul en fonction du critère et du coefficient de CFL.

Nous pouvons constater qu'avec un bon critère CFL, le pas de temps de stabilité du schéma LTS2 est seulement 2 fois plus petit que celui du schéma RK2. Nous observons aussi que les meilleures performances ne sont pas obtenues avec le plus grand nombre de classes de pas de temps. Rappelons que le nombre de classes de pas de temps augmente la quantité de kernels et

FIGURE 7.17 – Temps de calcul des méthodes RK2 et LTS2 en fonction du critère et du coefficient de CFL sur GPU NVidia GeForce GTX 1070 dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 1$. La dernière ligne correspond au temps de calcul obtenu en prenant pour chaque maille le plus grand pas de temps donné par tous les critères CFL précédents.

Méthode	CFL			$\Delta t^l_{\min}$ (s)	N_t	T_s (h)
	crit.	α	coeff.			
RK2	1	-	4.5	$1.446 \cdot 10^{-13}$	0	0.44
LTS2	1	-	0.22	$7.070 \cdot 10^{-15}$	7	1.10
	2	4/3	$1.2 \cdot 10^{3}$	$3.285 \cdot 10^{-15}$	8	2.21
		1	0.20	$8.601 \cdot 10^{-15}$	7	0.96
		3/4	$3.2 \cdot 10^{-4}$	$1.931 \cdot 10^{-14}$	6	0.54
		2/3	$3.1 \cdot 10^{-5}$	$2.094 \cdot 10^{-14}$	5	0.50
		1/2	$4.5 \cdot 10^{-7}$	$3.894 \cdot 10^{-14}$	4	0.39
		1/4	$6.5 \cdot 10^{-10}$	$7.721 \cdot 10^{-14}$	3	0.33
	$\max \Delta t$			$7.721 \cdot 10^{-14}$	4	0.28

le nombre d'interfaces. Le nombre de mailles par classe de pas de temps et par interface est donné dans le tableau 7.18.

Du point de vue de la répartition des classes de pas de temps, le cas « $\max \Delta t$ » est très proche de celui pour lequel nous appliquons le critère CFL2 avec le paramètre $\alpha = 1/4$. La principale différence réside dans le fait que l'utilisation du pas de temps maximal de toutes les conditions CFL précédentes déplace un certain nombre de mailles dans les classes supérieures.

Puisque la classe $\mathcal{C}_0$ contient très peu de mailles dans le cas « $\max \Delta t$ », nous avons relancé la simulation en imposant le maximum de $N_t = 3$ dans le but de réduire le nombre de kernels. Suite à cette modification, le gain de performances constaté est de 2.77 %. Ce gain peut paraître faible, mais n'a nécessité que très peu d'efforts.

En comparant les temps de calcul passés sur chaque kernel (tableau 7.19), nous constatons que les interfaces générées en pas de temps local représentent environ 30 % du temps de calcul. De plus, l'inactivité du GPU liée aux changements de kernel passe de moins de 2 % à environ 10 %.

De l'autre côté, nous pouvons à nouveau constater que le terme de flux local est beaucoup moins coûteux que le terme de flux classique. Remarquons aussi que, du fait que le plus petit pas de temps du schéma LTS2 est 2 fois plus petit que celui du schéma RK2, et puisque le kernel calculant le modèle

FIGURE 7.18 – Composition des zones homogènes pour les méthodes RK2 et LTS2 en fonction du critère et du coefficient de CFL dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 1$.

(a) Zones homogènes.

Méthode	RK2	LTS2							
CFL[-α]	1	1	2-4/3	2-1	2-3/4	2-2/3	2-1/2	2-1/4	max Δt
N_t	0	7	8	7	6	5	4	3	4
$\#\mathcal{C}_0$ (5.30)	77652	640	601	649	799	876	1063	4741	4560
$\#\mathcal{T}_0$ (5.34)	-	608	592	660	593	540	527	1552	1603
$\#\mathcal{C}_1 \setminus \mathcal{T}_0$	-	1036	1035	975	1289	1422	3239	21551	2414
$\#\mathcal{T}_1$	-	984	1016	984	1113	1099	1504	2056	2178
$\#\mathcal{C}_2 \setminus \mathcal{T}_1$	-	1156	1128	1167	1749	2812	11063	11169	9941
$\#\mathcal{T}_2$	-	1088	1068	1093	1269	1443	4548	5499	11074
$\#\mathcal{C}_3 \setminus \mathcal{T}_2$	-	1208	1065	1262	4843	9403	17074	31084	39058
$\#\mathcal{T}_3$	-	1036	924	1045	4352	4682	6204	-	6608
$\#\mathcal{C}_4 \setminus \mathcal{T}_3$	-	1704	833	1779	20045	18492	32430	-	216
$\#\mathcal{T}_4$	-	2224	895	2331	7332	5573	-	-	-
$\#\mathcal{C}_5 \setminus \mathcal{T}_4$	-	9888	1147	10259	30144	31310	-	-	-
$\#\mathcal{T}_5$	-	9828	1448	10331	4108	-	-	-	-
$\#\mathcal{C}_6 \setminus \mathcal{T}_5$	-	33384	3455	36614	16	-	-	-	-
$\#\mathcal{T}_6$	-	12476	5734	8443	-	-	-	-	-
$\#\mathcal{C}_7 \setminus \mathcal{T}_6$	-	392	15980	60	-	-	-	-	-
$\#\mathcal{T}_7$	-	-	17203	-	-	-	-	-	-
$\#\mathcal{C}_8 \setminus \mathcal{T}_7$	-	-	23528	-	-	-	-	-	-

(b) Interfaces.

Méthode	RK2	LTS2							
CFL[-α]	1	1	2-4/3	2-1	2-3/4	2-2/3	2-1/2	2-1/4	max Δt
N_t	0	7	8	7	6	5	4	3	4
$\#\mathcal{I}_0$ (5.35)	-	570	570	580	544	486	464	1428	1474
$\#\overline{\mathcal{T}_0} \cap \overline{\mathcal{C}_1 \setminus \mathcal{T}_0}$	-	582	610	544	486	474	464	1176	1202
$\#\mathcal{I}_1$	-	858	900	858	992	972	1356	1560	1802
$\#\overline{\mathcal{T}_1} \cap \overline{\mathcal{C}_2 \setminus \mathcal{T}_1}$	-	936	966	936	1004	992	1152	3168	2584
$\#\mathcal{I}_2$	-	972	966	976	1132	1276	3796	4298	9278
$\#\overline{\mathcal{T}_2} \cap \overline{\mathcal{C}_3 \setminus \mathcal{T}_2}$	-	996	1026	1030	1088	1294	5456	5772	9148
$\#\mathcal{I}_3$	-	978	906	1006	3570	3936	5222	-	7649
$\#\overline{\mathcal{T}_3} \cap \overline{\mathcal{C}_4 \setminus \mathcal{T}_3}$	-	888	790	922	4800	5532	6426	-	309
$\#\mathcal{I}_4$	-	1872	856	2040	6542	4430	-	-	-
$\#\overline{\mathcal{T}_4} \cap \overline{\mathcal{C}_5 \setminus \mathcal{T}_4}$	-	2910	1002	3048	6654	6270	-	-	-
$\#\mathcal{I}_5$	-	8172	1326	8522	4998	-	-	-	-
$\#\overline{\mathcal{T}_5} \cap \overline{\mathcal{C}_6 \setminus \mathcal{T}_5}$	-	9132	1820	9434	24	-	-	-	-
$\#\mathcal{I}_6$	-	12028	4831	8994	-	-	-	-	-
$\#\overline{\mathcal{T}_6} \cap \overline{\mathcal{C}_7 \setminus \mathcal{T}_6}$	-	480	7101	66	-	-	-	-	-
$\#\mathcal{I}_7$	-	-	13042	-	-	-	-	-	-
$\#\overline{\mathcal{T}_7} \cap \overline{\mathcal{C}_8 \setminus \mathcal{T}_7}$	-	-	13504	-	-	-	-	-	-

FIGURE 7.19 – Temps de calcul par kernel des méthodes RK2 et LTS2 (cas « $\max \Delta t$ ») sur GPU NVidia GeForce GTX 1070 dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 1$ sur 0.1 ns de temps physique.

Kernel	RK2		LTS2	
	Temps (ms)	Ratio (%)	Temps (ms)	Ratio (%)
Volume	1048	13.86	585	11.10
Flux	4716	62.35	923	17.51
Flux local	-	-	947	17.97
Interface	-	-	1536	29.14
Masse	821	10.85	373	7.08
Euler	303	4.00	139	2.64
Matériaux	495	6.55	159	3.02
Thévenin	41	0.54	71	1.34
Inactif	138	1.82	528	10.02

de Thévenin se trouve au niveau de l'antenne dans la classe $\mathcal{C}_0$, le temps de calcul passé sur ce kernel est 2 fois plus important dans le cas du schéma LTS2 (au *prorata* du temps de calcul global). Ce phénomène est inverse dans le cas du kernel appliquant les propriétés des matériaux diélectriques puisque ces derniers sont situés dans les classes de pas de temps supérieures.

FIGURE 7.20 – Temps de calcul des méthodes RK2 et LTS2 sur GPU NVidia GeForce GTX 1070 dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 2$.

Méthode	CFL		Δt^l_{min} (s)	N_t	T_{s} (h)
	crit.	coeff.			
RK2	1	3.1	$6.641 \cdot 10^{-14}$	0	1.82
LTS2	$0.49 \cdot \max \Delta t$		$3.350 \cdot 10^{-14}$	3	1.33

Les performances du schéma LTS2 par rapport au schéma RK2 à l'ordre d'interpolation spatiale $d = 2$ sont présentées dans le tableau 7.20. Nous constatons que là aussi le schéma à pas de temps local est plus efficace que le schéma à pas de temps homogène. Le critère CFL utilisé ici est celui du « $\max \Delta t$ » qui a donné les meilleures performances à l'ordre $d = 1$, avec un abaissement global de cette condition CFL d'un facteur 0.49 pour une

FIGURE 7.21 – Composition des zones homogènes pour les méthodes RK2 et LTS2 dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 2$.

(a) Zones homogènes.

Méthode	RK2	LTS2
N_t	0	3
$\#\mathcal{C}_0$ (5.30)	77652	4641
$\#\mathcal{T}_0$ (5.34)	-	1539
$\#\mathcal{C}_1 \setminus \mathcal{T}_0$	-	6684
$\#\mathcal{T}_1$	-	4677
$\#\mathcal{C}_2 \setminus \mathcal{T}_1$	-	15188
$\#\mathcal{T}_2$	-	8630
$\#\mathcal{C}_3 \setminus \mathcal{T}_2$	-	36293

(b) Interfaces.

Méthode	RK2	LTS2
N_t	0	3
$\#\mathcal{I}_0$ (5.35)	-	1400
$\#\overline{\mathcal{T}_0} \cap \overline{\mathcal{C}_1 \setminus \mathcal{T}_0}$	-	1220
$\#\mathcal{I}_1$	-	3834
$\#\overline{\mathcal{T}_1} \cap \overline{\mathcal{C}_2 \setminus \mathcal{T}_1}$	-	4996
$\#\mathcal{I}_2$	-	7069
$\#\overline{\mathcal{T}_2} \cap \overline{\mathcal{C}_3 \setminus \mathcal{T}_2}$	-	8479

FIGURE 7.22 – Temps de calcul par kernel des méthodes RK2 et LTS2 sur GPU NVidia GeForce GTX 1070 dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée pour l'ordre d'interpolation spatiale $d = 2$ sur 0.1 ns de temps physique.

Kernel	RK2		LTS2	
	Temps (ms)	Ratio (%)	Temps (ms)	Ratio (%)
Volume	3234	10.95	2106	9.61
Flux	19448	65.83	4639	21.15
Flux local	-	-	5673	25.89
Interface	-	-	4747	21.65
Masse	2230	7.54	1313	5.99
Euler	2215	7.49	1302	5.94
Matériaux	2017	6.82	689	3.14
Thévenin	137	0.46	268	1.22
Inactif	253	0.85	1157	5.27

simulation stable. Là aussi, cette condition CFL donne un pas de temps de stabilité du schéma LTS2 environ 2 fois plus petit que celui du schéma RK2.

Nous constatons que cette méthode est 1.37 fois plus rapide que le schéma RK2 à l'ordre d'interpolation $d = 2$, contre une accélération de 1.57 à l'ordre $d = 1$. Le critère CFL n'a pas été déterminé de manière optimale pour l'ordre 2, mais il permet d'effectuer une simulation stable.

Pour ce cas, le nombre de classes de pas de temps est naturellement de

4 ($N_t = 3$) avec un grand nombre de mailles dans la plus haute classe. Le nombre de mailles par classe de pas de temps et par interface est donné dans le tableau 7.21.

Du point de vue du temps passé sur les différents kernels (tableau 7.22), les résultats sont similaires à ceux de la simulation à l'ordre $d = 1$, à la différence près que les kernels du terme de flux (et d'interface) et de l'étape d'Euler prennent plus de temps. Ces résultats concernant le terme de flux étaient prévisibles puisque sa parallélisation est moins aisée que celle du terme de volume.

Les performances constatées avec ce schéma sont meilleures que pour le schéma RK2, mais nettement inférieures aux espérances formulées dans la section 5.4.4 suite aux expérimentations sur l'équation du transport en dimension 1.

Ce manque de performances est essentiellement lié à la petite taille du cas d'application de l'antenne dipolaire placée à proximité de la tête humaine simplifiée. Le facteur d'échelle entre la plus petite et la plus grande maille de cette simulation est d'environ 100, ce qui ne suffit pas à compenser les temps de latence OpenCL induits par la multiplication des kernels et la réduction de la masse de données qu'ils traitent.

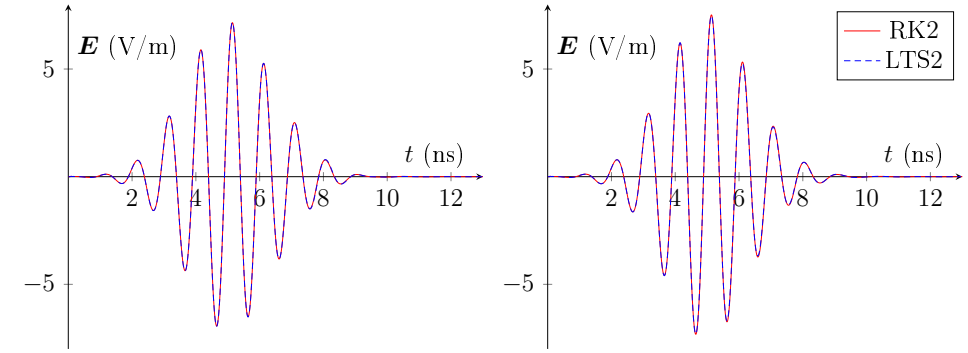
De bien meilleures performances peuvent être attendues sur un cas d'application tel que celui présenté dans la section 7.3 et pour lequel le facteur d'échelle est de l'ordre de 1000. Nous pourrons aussi gagner en performances en revoyant la stratégie de découpage en zones dans le but de réduire le nombre de kernels en les regroupant (toutes les mailles tampon dans une zone et les autres mailles dans une seconde zone, et de même pour les interfaces) et en leur passant en argument des tableaux contenant les propriétés temporelles des mailles. Ainsi, seule la masse de données traitée par les kernels changera en fonction de l'étape courante du schéma et leur quantité sera énormément réduite.

Résultats

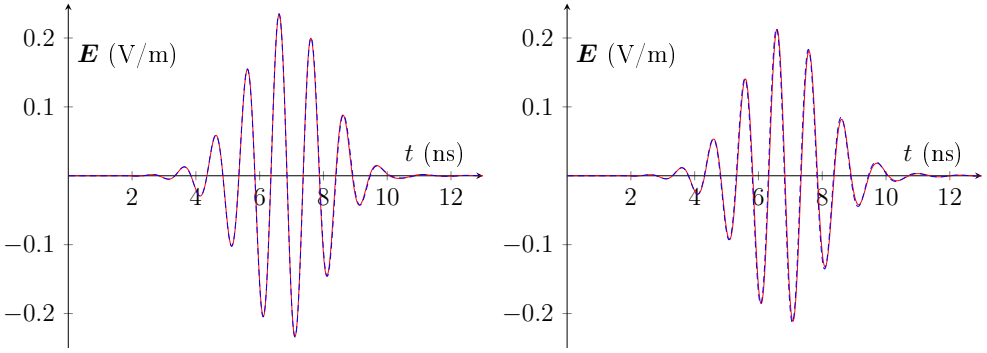
Les résultats de l'application du schéma LTS2 pour les ordres d'interpolation $d = 1$ et 2 sont identiques à ceux obtenus avec le schéma RK2 (figure 7.23).

FIGURE 7.23 – Valeur de la composante x_3 du champ électrique mesuré en différents points d'observation dans le cas de la simulation de l'antenne ELPOSD placée à proximité de la tête humaine simplifiée. Comparaison du schéma RK2 au schéma LTS2 pour les ordres d'interpolation $d = 1$ (à gauche) et 2 (à droite).

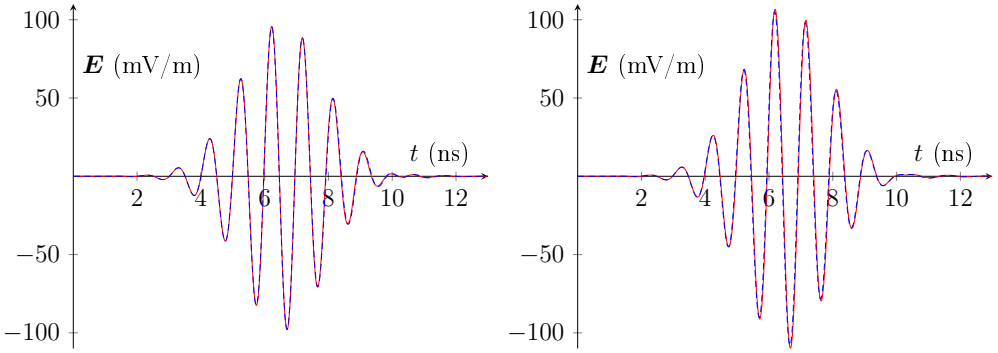
(a) Au point O_1.



(b) Au point O_2.



(c) Au point O_3.



7.3 Corps humain complet

L'objectif du projet HOROCH [33, 58] était de démontrer l'intérêt et la pertinence de l'utilisation du GD pour la conception et l'optimisation des objets connectés proches du corps humain par la confrontation mesure/simulation. Nous n'avons malheureusement pas de mesures à ce jour pour effectuer cette confrontation. Néanmoins, la qualité des résultats produits par notre solveur a été démontrée au cours de l'étude comparative réalisée dans la section 7.2.2 où nous le confrontons à un solveur FDTD largement utilisé dans l'industrie française. Ce cas d'application sur corps humain complet est donc uniquement présenté pour la performance technique qu'il représente.

Dans le secteur des objets connectés, il est d'usage, exception faite de certains besoins spécifiques, d'utiliser des technologies standards trouvées sur catalogue. En particulier pour les communications entre un dispositif de mesure physiologique et une station de base, généralement un téléphone, le protocole naturel est le Bluetooth basse consommation (ou BLE pour « *Bluetooth Low Energy* »).

Nous avons donc choisi une antenne de technologie LTCC (« *Low Temperature Cofired Ceramic* ») [72] du fabriquant Johanson Technology et ayant pour référence 2450AT18B100E. La figure 7.24 représente le maillage surfacique du dispositif rayonnant. Cette antenne rayonne à une fréquence centrale de 2.45 GHz et est composée d'un volume de céramique, incluant une cavité formée de bandes métalliques, et de vias représentés en simulation par la condition PEC (section 2.1.1). Le substrat de l'antenne situé entre le plan de masse et le dispositif rayonnant est composé d'un matériau 4.4 nommé FR4. Les propriétés électriques de ces matériaux sont données dans le tableau 7.25.

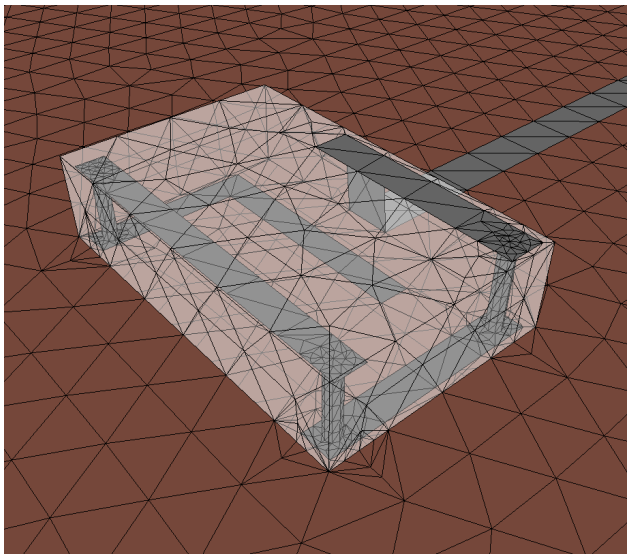
Le modèle d'excitation utilisé est le générateur de Thévenin (section 2.3.2) avec une impédance réelle de 50 Ohms. Le signal du générateur de tension que nous utilisons est une impulsion gaussienne modulée (7.5) dont les paramètres sont :

- $A = 1$;
- $f_0 = 2.4$ GHz ;
- $\Delta_f = 1$ GHz ;
- $a_0 = 0.001$ (ou 0.1 %).
- $a_A = 0.05$ (ou 5 %) ;

Avec ces valeurs, la durée du signal est d'environ 6 ns.

Dans cette configuration, nous étudions le rayonnement de l'antenne LTCC placée entre l'avant-bras et le modèle de corps humain complet (telle une montre connectée), c'est-à-dire avec 12 organes pour lesquels nous disposons

FIGURE 7.24 – Maillage du dispositif rayonnant de l'antenne de technologie LTCC utilisée. Le PEC est en gris, la céramique en transparence et le FR4 en marron.



des propriétés électriques représentatives du corps humain [25–27] : le cerveau, le cœur, les poumons (2), le foie, la vésicule biliaire, la rate, le pancréas, les reins (2), une portion du colon, la vessie ainsi que le squelette avec un haut niveau de détail, du cartilage et la peau. Le reste du corps est considéré comme étant du muscle. Les propriétés électriques de ces matériaux sont donnés dans le tableau 7.25.

Le maillage du corps complet est issu de la numérisation d'un mannequin anthropomorphique, le modèle PBU-60 produit par la société Kyoto Kagaku. L'IRCAD de Strasbourg a réalisé des images 3-d haute résolution à l'aide d'un scanner CT (pour « *Computerized Tomography* »). Ensuite, la société Visible Patient a procédé à la segmentation de ces images en surfaces. Nous avons par la suite retravaillé les géométries pour les rendre conformes aux besoins du calcul numérique (figure 7.26).

Le maillage de la scène simulée est composé de 5840122 tétraèdres, soit 23360488 hexaèdres après découpage (figure 4.3) auxquels sont ajoutés 47340 mailles de PML (section 2.1.5) aux limites du domaine de calcul. Cette simulation occupe 40 Go de mémoire GPU à l'ordre $d = 1$ (adaptatif), 100 Go à l'ordre $d = 2$ et 200 Go à l'ordre $d = 3$.

Le facteur d'échelle entre la plus petite ($5.076 \cdot 10^{-5}$ m) et la plus grande ($5.781 \cdot 10^{-2}$ m) arête de tétraèdre du maillage est de 1139. Ce facteur passe à 863 si nous considérons la plus petite arête de chaque tétraèdre. Après le

FIGURE 7.25 – Propriétés électromagnétiques des matériaux utilisés dans la scène du corps humain complet.

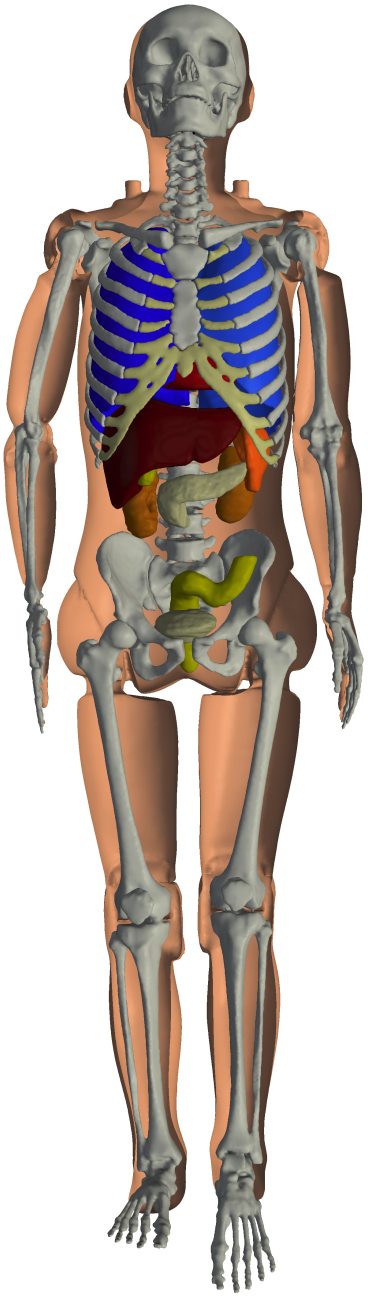
Matériau	ε_r	μ_r	σ (S/m)	$\sigma^\star$
céramique	7.8	1	0	0
FR4	4.4	1	0.01	0
cerveau	48.34	1	2.02	0
cœur	58.67	1	3.02	0
poumon	22	1	0.36	0
foie	41.82	1	1.9	0
bile	60	1	2	0
rate	56.75	1	2.46	0
pancréas	56.75	1	2.46	0
rein	56.83	1	2.62	0
colon	48.5	1	0.93	0
vessie	20	1	0.7	0
muscle	50	1	1.33	0
os	11.41	1	0.43	0
cartilage	36	1	1.6	0
vide	1	1	0	0

découpage en hexaèdres, nous obtenons un facteur d'échelle entre la plus petite ($7.006 \cdot 10^{-6}$ m) et la plus grande ($3.538 \cdot 10^{-2}$ m) arête du maillage de 5050 (hors PML). Ce facteur passe à 1373 si nous considérons la plus petite arête de chaque hexaèdre (toujours hors PML). Ces valeurs témoignent de la forte déformation des hexaèdres issus du découpage de tétraèdres.

La simulation a nécessité environ 2091 heures machine dont 2072 heures de calcul GPU pour réaliser 10 ns de temps physique à l'aide de la technique de l'ordre adaptatif sur la machine de calcul possédée par AxesSim. Le choix de l'ordre adaptatif a été fait car cette machine est composée de 8 GPU NVidia GeForce GTX 1080 Ti possédant chacun une mémoire de 11 Go, soit un total de 88 Go. Chaque GPU a donc traité environ 5 Go de données pendant 10.79 jours (homme) de calcul pour une durée totale de simulation de 10.89 jours en comptant les 2.34 heures de pré- et post-traitement et récupération des données en sortie.

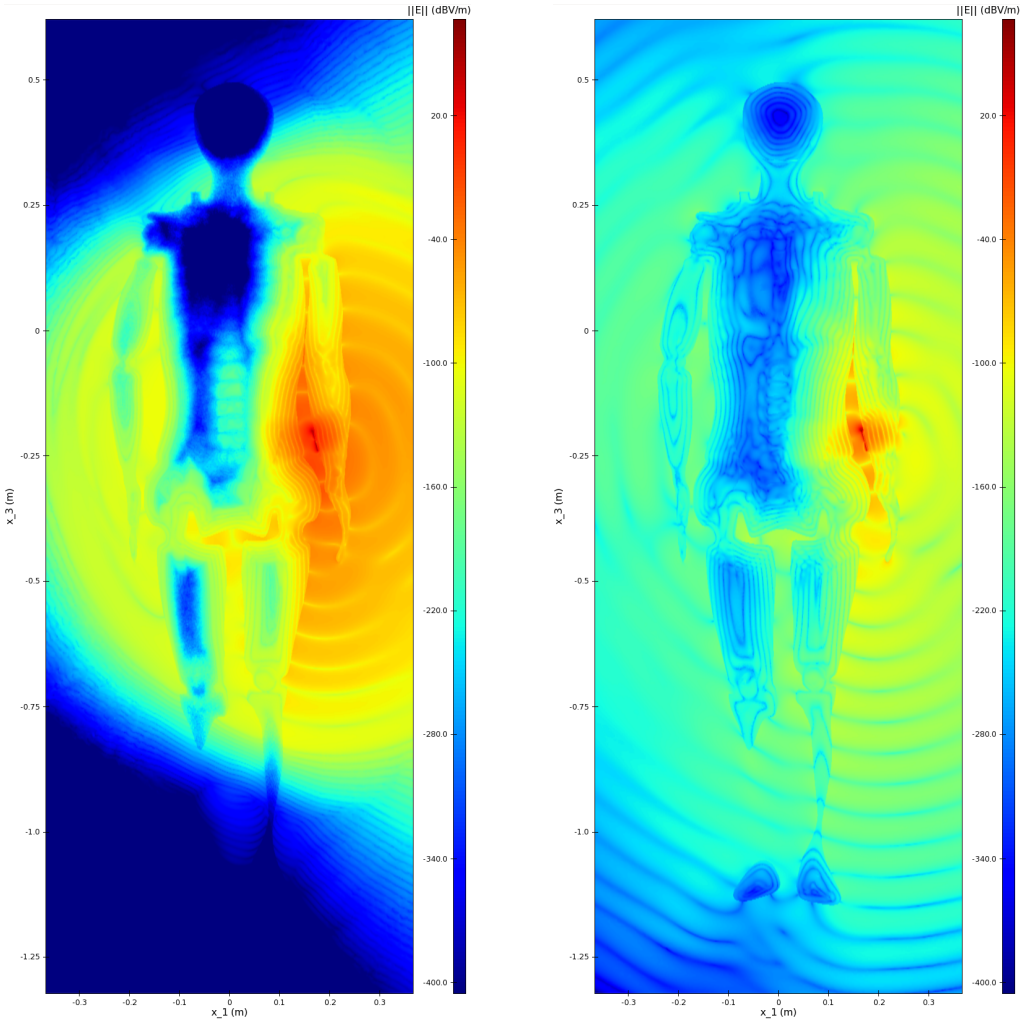
Les données en sortie, principalement sous forme de plans de coupe, représentent une taille mémoire de 55 Go. Un aperçu des résultats sous forme de plan de coupe est présenté dans la figure 7.27. Les autres plans de coupe,

FIGURE 7.26 – Représentation de la géométrie du corps humain complet avec ses 12 organes (le crâne contient un cerveau).



un par nanoseconde écoulée, sont présentés en annexe C.

FIGURE 7.27 – Résultats de la simulation sur corps humain complet en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) aux temps $t = 2$ ns (à gauche) et $t = 4$ ns (à droite).



Nous pouvons constater la compression de la longueur d'onde dans les matériaux de forte permittivité et leur absorption par les tissus présentant une conductivité électrique non nulle. Les ondes étant plus rapides dans le vide, elles pénètrent dans le corps du côté droit (opposé à l'antenne) avant l'arrivée des ondes qui le traversent de part en part dans le sens gauche-droite.

Des tests de scalabilité du solveur ont été effectués sur cette simulation

jusqu'à 256 GPU pour les ordres d'interpolation 1 à 3. Ces résultats sont présentés dans la section 7.4.4.

7.4 Scalabilité MPI

L'implémentation MPI du solveur GD que nous avons présenté dans la section 4.2.7 nécessite de partitionner le maillage de la scène simulée dans le but d'affecter une partie à chaque périphérique de calcul. Ce partitionnement génère des interfaces supplémentaires qui augmentent le coût de calcul de la simulation. Ce surcoût diminue l'efficacité du solveur qui doit être étudiée dans le but de valider l'implémentation MPI. Nous étudions l'efficacité MPI (ou scalabilité, de l'anglais « *scaling* ») dans ce chapitre.

Pour cela, outre la machine de calcul possédée par la société AxesSim, nous avons bénéficié d'heures de calcul sur le Mésocentre de l'université de Strasbourg et sur le supercalculateur Suisse dénommé PizDaint, 3$^{\text{ème}}$ au TOP500 [68] jusqu'en juin 2018 avec plus de 25 PFlop $\cdot$ s^{-1}, et passé 5$^{\text{ème}}$ depuis. Les heures de calcul sur PizDaint ont été obtenues grâce à l'appel à projet SHAPE (*SME HPC Adoption Programme in Europe*) lancé par PRACE (*Partnership for Advanced Computing in Europe*).

7.4.1 Théorie de la scalabilité

Deux types de scalabilité peuvent être évalués : les scalabilités faible et forte.

L'évaluation de la **scalabilité faible** consiste à traiter un problème de taille proportionnelle au nombre de périphériques de calcul investis. Pour ce type de scalabilité, le temps de calcul idéal est constant quelque soit le nombre de périphériques de calcul. En pratique, le temps de calcul augmente généralement avec le nombre de périphériques investis. Ainsi, le temps de simulation mesuré est de la forme :

$$T_{\text{s}}^{k}(k\mathcal{M}) = T_{\text{s}}^{1}(\mathcal{M}) + P_{k}^{f}(\mathcal{M}), \tag{7.8}$$

où T_{s}^{k} représente le temps de simulation sur k périphériques, $k\mathcal{M}$ représente un maillage dont le nombre de mailles est k fois plus important que le maillage $\mathcal{M}$ et avec $(P_{k}^{f})_{k}$ une suite généralement croissante représentant la perte d'efficacité sur k périphériques. Les valeurs de cette suite dépendent du problème initial.

L'évaluation de la **scalabilité forte** consiste à traiter un problème de taille fixe sur un nombre croissant de périphériques de calcul. Pour ce type de scalabilité, le temps de calcul idéal est divisé par le nombre de périphériques de calcul investis. Cependant, en pratique, le temps de calcul effectif est là aussi généralement plus long que le temps idéal. Ainsi, le temps de simulation mesuré dans ce cas est de la forme :

$$T_{\mathrm{s}}^{k}(\mathcal{M}) = \frac{T_{\mathrm{s}}^{1}(\mathcal{M})}{k} + P_{k}^{F}(\mathcal{M}), \tag{7.9}$$

avec $(P_k^F)_k$ une suite généralement croissante représentant la perte d'efficacité en divisant le problème sur k périphériques. Là aussi, les valeurs de cette suite dépendent du problème initial.

L'efficacité en scalabilité faible sur k périphériques est donnée par :

$$E_{k}^{f}(\mathcal{M}) = \frac{T_{\mathrm{s}}^{1}(\mathcal{M})}{T_{\mathrm{s}}^{k}(k\mathcal{M})} = \frac{1}{1 + \frac{P_{k}^{f}(\mathcal{M})}{T_{\mathrm{s}}^{1}(\mathcal{M})}} \leq 1, \tag{7.10}$$

terme de la suite $(E_k^f)_k$, généralement décroissante et positive.

L'efficacité en scalabilité forte sur k périphériques est donnée par :

$$E_{k}^{F}(\mathcal{M}) = \frac{T_{\mathrm{s}}^{1}(\mathcal{M})}{kT_{\mathrm{s}}^{k}(\mathcal{M})} = \frac{1}{1 + k\frac{P_{k}^{F}(\mathcal{M})}{T_{\mathrm{s}}^{1}(\mathcal{M})}} \leq 1, \tag{7.11}$$

terme de la suite $(E_k^F)_k$, généralement décroissante et positive.

Dans le cas de notre solveur, ces pertes d'efficacité sont essentiellement générées par l'augmentation du nombre d'interfaces MPI. Nous les mesurons dans la suite sur différentes machines de calcul.

7.4.2 Machine de calcul AxesSim

La machine de calcul possédée par AxesSim est composée d'un CPU Intel Dual Xeon E5-2650 v4 (24 cœurs physiques hyperthreadés cadencés à 2.20 GHz) et de 8 GPU NVidia GeForce GTX 1080 Ti, chacun regroupant 3584 cœurs CUDA cadencés à 1.582 GHz et possédant 11 Go de mémoire.

L'unique nœud de cette machine possède 256 Go de mémoire vive. Les processus MPI y sont donc tous hébergés en parallèle et les transferts mémoire sont des transferts internes à la mémoire vive du nœud, donc très rapides mais aussi très nombreux.

Nous avons mené des tests de scalabilité faible et forte jusqu'à 8 GPU sur cette machine. Les cas d'application utilisés pour ces tests sont semblables

à celui qui nous a permis de déterminer la convergence du schéma couplé GD-RK2 (section 7.1). Nous simulons une onde plane dans un cube dont nous paramétrons la taille et le raffinement selon le besoin. Les tailles des maillages utilisés sont approximativement les termes d'une suite géométrique de raison 2. Nous évaluons ainsi la scalabilité du solveur pour des tailles de maillage allant de 10 mille à 16 millions de mailles. Les résultats d'efficacité sont présentés dans le tableau 7.28.

FIGURE 7.28 – Efficacité du schéma couplé GD-RK2 à l'ordre spatial $d = 2$ sur GPU NVidia GeForce GTX 1080 Ti. La correspondance explicite en nombre de mailles et la taille mémoire correspondante sont uniquement précisés dans le second tableau.

(a) Scalabilité faible (maillage proportionnel).

1 GPU	2 GPU		4 GPU		8 GPU	
$\#\mathcal{M}$	$\#2\mathcal{M}$	E_2^f	$\#4\mathcal{M}$	E_4^f	$\#8\mathcal{M}$	E_8^f
22^3	28^3	0.728	35^3	0.620	44^3	0.494
30^3	38^3	0.769	-	-	-	-
37^3	47^3	0.807	-	-	-	-
43^3	55^3	0.743	-	-	-	-
50^3	63^3	0.885	-	-	-	-
63^3	80^3	0.891	-	-	-	-
80^3	101^3	0.965	127^3	0.954	160^3	0.928
100^3	126^3	0.963	159^3	0.969	200^3	0.944
126^3	159^3	0.993	201^3	0.989	252^3	0.994

(b) Scalabilité forte (maillage fixe).

$\#\mathcal{M}$	RAM (Go)	E_2^F	E_4^F	E_8^F
$22^3 \approx 10$k	0.037	0.347	0.188	0.072
$30^3 \approx 25$k	0.088	0.671	-	-
$37^3 \approx 50$k	0.156	0.725	-	-
$43^3 \approx 75$k	0.236	0.891	-	-
$50^3 \approx 125$k	0.356	0.689	-	-
$63^3 \approx 250$k	0.672	0.850	-	-
$80^3 \approx 500$k	1.302	0.883	-	-
$100^3 \approx 1$M	2.426	0.923	0.793	0.624
$126^3 \approx 2$M	4.852	0.962	0.845	0.734

Nous remarquons que le solveur est très efficace en scalabilité faible ($E_k^f >$

0.9) pour des tailles de maillage supérieures à 500k mailles par GPU. Cette taille de maillage correspond à environ 1.3 Go de mémoire par GPU à l'ordre d'interpolation spatiale $d = 2$, soit environ 12 % de la mémoire globale d'un GPU NVidia GeForce GTX 1080 Ti.

Les résultats en scalabilité forte rejoignent ceux de la scalabilité faible : le solveur est fortement scalable sur 2 GPU avec un maillage initial de taille supérieure à 1 million de mailles, soient 500k mailles par GPU.

7.4.3 Mésocentre de Strasbourg

Le Mésocentre de Strasbourg met à disposition des étudiants (gratuitement) et des professionnels de la simulation, des heures de calcul sur CPU et GPU de différents types. Nous avons notamment utilisé les GPU NVidia Tesla K20m (regroupant 2496 cœurs CUDA cadencés à 706 MHz et possédant 4.6 Go de mémoire) et les CPU Intel Xeon de la famille E5-26** (composés de 8 à 16 cœurs physiques cadencés à 2.6 GHz et possédant 64 ou 128 Go de mémoire vive).

Nous avons mené des tests de scalabilité faible et forte jusqu'à 16 GPU et 4 CPU (en OpenCL) sur ce calculateur. Les cas d'application utilisés pour ces tests sont identiques à ceux utilisés sur la machine de la société AxesSim (mis à part le dernier pour des raisons de taille mémoire). Nous évaluons la scalabilité du solveur pour des tailles de maillage allant de 10 mille à 8 millions de mailles.

Les résultats d'efficacité sur GPU sont présentés dans le tableau 7.29. Le solveur est très efficace sur ces derniers : l'efficacité en scalabilité faible est supérieure à 0.9 à partir d'environ 100k mailles par GPU et même supérieure à 1 à partir de 500k mailles par GPU ! Ce seuil d'efficacité est à nouveau vérifié en scalabilité forte. Rappelons que, contrairement à la machine de calcul possédée par AxesSim, tous les GPU ne sont pas associés au même nœud sur le Mésocentre. Les transferts MPI sont donc répartis entre les différents nœuds sollicités.

Sur CPU, les résultats en scalabilité faible sont inférieurs. Pour un maillage initial de 500k mailles (seuil identifié sur la machine de calcul possédée par AxesSim) l'efficacité est d'environ 0.8. Cette baisse est principalement due au fait que le masquage des communications MPI par l'exécution des kernels OpenCL n'est plus possible. En effet, lors de l'exécution d'un kernel OpenCL sur CPU, tous les cœurs sont sollicités. Le kernel et les communications MPI sont donc mis en concurrence et mutuellement ralentis. Dans les versions récentes d'OpenCL, un CPU peut être subdivisé pour libérer des cœurs de calcul. Il s'agit là d'une piste pour améliorer l'efficacité sur CPU.

FIGURE 7.29 – Efficacité du schéma couplé GD-RK2 à l'ordre spatial $d = 2$ sur GPU NVidia Tesla K20m. La correspondance explicite en nombre de mailles et la taille mémoire correspondante sont uniquement précisés dans le second tableau.

(a) Scalabilité faible (maillage proportionnel).

1 GPU	2 GPU		4 GPU		8 GPU		16 GPU	
$\#\mathcal{M}$	$\#2\mathcal{M}$	E_2^f	$\#4\mathcal{M}$	E_4^f	$\#8\mathcal{M}$	E_8^f	$\#16\mathcal{M}$	E_{16}^f
22^3	28^3	0.482	-	-	-	-	-	-
30^3	38^3	0.706	-	-	-	-	-	-
37^3	47^3	0.861	-	-	-	-	-	-
43^3	55^3	0.913	-	-	-	-	-	-
50^3	63^3	0.993	80^3	0.941	-	-	-	-
63^3	80^3	0.977	101^3	0.940	-	-	-	-
80^3	101^3	1.004	127^3	1.006	160^3	1.012	202^3	0.950
100^3	126^3	1.011	159^3	1.015	200^3	1.033	-	-
122^3	154^3	1.009	194^3	1.022	-	-	-	-

(b) Scalabilité forte (maillage fixe).

$\#\mathcal{M}$	RAM (Go)	E_2^F	E_4^F	E_8^F	E_{16}^F
$22^3 \approx 10$k	0.037	0.323	-	-	-
$30^3 \approx 25$k	0.088	0.515	-	-	-
$37^3 \approx 50$k	0.156	0.659	-	-	-
$43^3 \approx 75$k	0.236	0.761	-	-	-
$50^3 \approx 125$k	0.356	0.874	-	-	-
$63^3 \approx 250$k	0.672	0.925	0.689	-	-
$80^3 \approx 500$k	1.302	0.952	0.854	0.717	0.468
$100^3 \approx 1$M	2.426	0.961	0.861	0.754	0.597
$122^3 \approx 1.8$M	4.243	0.966	0.905	0.834	0.638

Toujours sur CPU, en scalabilité forte, l'efficacité est généralement supérieure à 0.9 à partir du seuil de 500k mailles par périphérique. Ce résultat pourrait s'expliquer par le fait que la réduction de la taille des kernels laisse progressivement plus de place aux communications MPI, elles-mêmes plus courtes, bien que plus nombreuses. Cela dit, la réduction de la taille des kernels peut aussi induire l'apparition d'effets de cache.

7.4.4 PizDaint via PRACE

Suite à l'appel à projet SHAPE lancé par PRACE dans le but de rendre les supercalculateurs européens accessibles aux PME européennes, la société AxesSim a obtenu 2000 heures de calcul sur le supercalculateur PizDaint. Cette machine est équipée de plus de 5000 GPU NVidia Tesla P100, équivalents en terme de performance au GPU NVidia GeForce GTX 1080 Ti.

L'objectif était d'utiliser ces heures de calcul pour effectuer des simulations sur le corps humain complet (avec organes). Cependant, après la première simulation effectuée sur la machine de calcul possédée par AxesSim, nous avons constaté que ce quota d'heures était insuffisant pour une unique simulation. Nous avons donc utilisé ces heures pour effectuer des tests de scalabilité forte sur la configuration présentée sans la section 7.3.

Nous sommes arrivés jusqu'à l'utilisation simultanée de 256 GPU [73], une première pour le solveur `teta-clac`. Les résultats sont présentés dans le tableau 7.30. Les temps de référence qui ont permis de calculer ces efficacités n'ont pas été obtenus sur un seul GPU mais sur un nombre croissant de GPU en fonction de l'ordre spatial. En effet, la taille mémoire nécessaire rend impossible la simulation sur un seul GPU.

FIGURE 7.30 – Efficacité du schéma couplé GD-RK2 sur la configuration du corps humain complet pour différents ordres d'interpolation sur GPU NVidia Tesla P100.

#GPU	$d = 1$ (*)	$d = 2$	$d = 3$
4	1 (**)	-	-
8	0.980	1 (**)	-
16	0.955	0.980	1 (**)
32	0.909	0.943	0.970
64	0.844	0.889	0.919
128	0.749	-	-
256	0.612	-	-

(*) *Simulations en ordre spatial adaptatif (PML à l'ordre 2).*
(**) *Simulation prise comme référence pour l'ordre courant.*

Nous constatons que la scalabilité du solveur est très bonne sur ce cas d'application. L'efficacité du solveur à l'ordre (adaptatif) 1 sur 256 GPU est de 0.612 par rapport à une exécution sur 4 GPU, soit une efficacité de 0.6 par rapport à 8 GPU. Ce qui signifie que le temps GPU nécessaire à l'exécution sur 256 GPU est d'environ 3454 heures pour réaliser 10 ns de temps physique mais avec un temps réel de restitution d'environ 13.5 heures (homme).

Notons que le coefficient CFL utilisé pour cette simulation est naïvement de 0.9 ($\Delta t = 3.165 \cdot 10^{-15}$ s) et pourrait très certainement être augmenté par la méthode de la puissance itérée (section 3.3.4). Ce temps pourra encore être réduit avec l'application du schéma LTS2 (section 5.3), lorsque ce dernier sera pleinement fonctionnel en MPI.

Conclusion

Dans ce dernier chapitre, nous avons présenté des résultats permettant de valider notre implémentation de solveur GD, autant par la précision des calculs que par leur vitesse d'exécution.

Dans un premier temps, nous avons présenté des résultats de convergence et de précision sur un cas académique de propagation d'une onde plane dans un cube pour différents ordres d'interpolation. L'ordre de convergence théorique du schéma en temps a été vérifié en pratique et l'erreur globale commise reste négligeable. Ces résultats ont été obtenus sur maillages structurés et non structurés et démontrent le bon fonctionnement du solveur ainsi que sa polyvalence du point de vue géométrique. Nous avons aussi comparé les caractéristiques des maillages étudiés, notamment leur nombre de mailles et la taille des arêtes afin de dégager un ordre de grandeur du malus dont bénéficient les maillages en tétraèdres.

Nous avons ensuite comparé notre solveur GD à un solveur implémentant la méthode des différences finies sur un cas d'application mettant en scène une antenne dipolaire à proximité de l'oreille d'une tête humaine simplifiée. Ce cas d'application présentant peu de formes rectilignes et une très petite source d'excitation avantage notre solveur GD en temps de calcul par le faible nombre de mailles non structurées requises. La grande quantité d'optimisations est cependant la principale raison des hautes performances produites par le solveur GD comme en témoignent les résultats en temps de calcul. Parmi ces optimisations, la méthode à pas de temps local présentée dans le chapitre 5 qui permettrait d'obtenir des accélérations encore plus importantes sur un cas tel que celui du corps humain complet.

Nous avons enfin présenté les résultats d'une simulation exécutée sur le modèle de corps humain complet dans le cadre du projet HOROCH. Le rayonnement d'une antenne de type Bluetooth placée telle une montre connectée à proximité du bras a été simulé sur 10 ns de temps physique en parallèle sur 8 GPU NVidia de dernière génération. Rappelons qu'il s'agit de la plus grande simulation réalisée par le solveur à ce jour. Celle-ci a aussi servi à effectuer des tests d'efficacité allant jusqu'à 256 GPU ce qui nous a permis

de pleinement valider son implémentation MPI. L'implémentation MPI de la méthode à pas de temps local n'a pas encore été finalisée mais devrait permettre de significativement réduire le temps de calcul.

Conclusion

Dans cette thèse, nous avons commencé par rappeler les équations de Maxwell et montré qu'elles constituent un système hyperbolique dit de Friedrichs (chapitre 1). Ce système a ensuite été présenté sous sa forme Galerkin Discontinue (GD) qui fait intervenir un flux numérique que nous avons choisi décentré afin d'améliorer la stabilité du schéma. Nous avons également caractérisé les conditions aux limites permettant d'assurer cette stabilité. La formulation GD a ensuite été présentée sous sa forme semi-discrète dont la solution converge vers celle du problème continu (annexe B).

Nous avons ensuite présenté plusieurs modèles mathématiques de simulation qui permettent de faciliter la génération de la scène simulée et de réduire le temps de calcul (chapitre 2).

Nous avons commencé par décrire des conditions aux limites respectant la condition d'existence de la solution, notamment la condition de Silver-Müller ainsi que les couches absorbantes PML très utilisées dans les applications industrielles. Grâce à ces conditions, le domaine de calcul peut être borné au plus près de la zone d'intérêt tout en limitant les réflexions qui viendraient polluer le domaine de calcul.

Nous avons aussi donné un ensemble de modèles permettant de simuler les plaques minces de matériaux. Ces modèles permettent de remplacer les volumes fins par des surfaces, sont complémentaires sur une large bande de fréquences, et réduisent les temps de simulation tout en facilitant la phase de maillage.

L'implémentation de la méthode GD que nous avons présentée (chapitre 3) utilise un espace d'approximation hexaédrique discrétisé par les points de Gauss-Legendre et muni des produits tensoriels des polynômes de Lagrange qui s'appuient sur ces points. Les propriétés de cet espace d'approximation simplifient les formules de quadrature par l'annulation du gradient des fonctions de base en de nombreux points d'interpolation. Ces simplifications n'apparaissent pas dans les mailles tétraédriques.

Nous pourrions aussi utiliser les points d'interpolation de Gauss-Lobatto qui produiraient d'importantes simplifications au niveau du terme de flux. Dans ce cas les points de bord sont placés sur les interfaces et l'extrapolation/application des champs sur le volume n'est plus nécessaire. Ils permettent aussi d'appliquer un schéma de type « *split form* » qui conserve mieux l'empreinte fréquentielle du signal. Ces points feront l'objet de futurs travaux sur le solveur.

Nous avons aussi présenté les schémas de type Runge-Kutta qui nous permettent d'intégrer temporellement la solution afin d'avancer cette dernière jusqu'au temps final souhaité. Néanmoins, la stabilité de la méthode RK2 et directement liée à la taille et la forme des mailles, ce qui complique son utilisation dans les applications en tétraèdres découpés. Nous avons donc présenté 2 conditions de stabilité de type CFL ainsi que 2 diagnostics de stabilité : le calcul des valeurs propres du système et la puissance itérée. Ce second diagnostic permet de déterminer avec précision le coefficient CFL de stabilité. Une implémentation OpenCL du calcul de la norme utilisée dans cette méthode nous permettra de déterminer ce coefficient rapidement et systématiquement en procédant à une dichotomie.

Cette implémentation de la méthode GD a été programmée à destination des GPU en utilisant la bibliothèque OpenCL (chapitre 4). La parallélisation sur plusieurs GPU est assurée via une implémentation du standard MPI. Le partitionnement du maillage est automatiquement généré à l'exécution afin de répartir la charge de calcul de façon équilibrée entre les accélérateurs. Toutes ces capacités permettent au solveur `teta-clac` de facilement résoudre des problèmes de grande taille sur un nombre quelconque de GPU.

Nous avons aussi décrit les adaptations qui ont été menées dans le but d'optimiser les kernels de calcul OpenCL à destination des CPU. Les performances ont été nettement améliorées sur ce type de périphériques mais restent dépendantes des caractéristiques du périphérique en lui-même.

Enfin, nous avons présenté la méthode de l'ordre d'interpolation spatiale adaptatif qui permet de corriger automatiquement la discrétisation dans les zones sur- ou sous-maillées. Cette méthode nous permet de mailler avec précisions (par de petites mailles) des géométries présentant un haut niveau de détail : l'ordre d'interpolation y sera automatiquement abaissé pour réduire le temps de calcul. Nous avons néanmoins constaté qu'un ordre d'interpolation trop faible risque de dénaturer le signal. Il est donc préférable de projeter l'utilisation d'un ordre d'interpolation élevé afin de conserver un ordre abaissé supérieur à 1.

Nous avons ensuite énoncé une variante locale de la méthode GD qui

permet d'alléger les calculs de flux et de découpler l'avancée en temps des mailles (chapitre 5). Cet opérateur local nous a permis de décrire deux schémas temporels. Premièrement, le schéma LRK2 qui est une implémentation à prédiction locale du schéma RK2. Ce schéma utilise un pas de temps homogène sur tout le maillage. Il a expérimentalement été démontré convergeant et stable.

Nous avons aussi présenté le schéma LTS2 pour lequel le pas de temps est local à chaque maille. Les accélérations obtenues à l'aide de cette méthode sont très proches de l'accélération théorique en dimension 1. Nous constatons aussi une accélération des calculs en dimension 3, mais plus éloignée des valeurs théoriques. Ce schéma n'est adapté qu'aux géométries composées de mailles de tailles très hétérogènes. Dans le cas d'un facteur d'échelle entre la plus petite et la plus grande maille de l'ordre de 100, le surcoût engendré par la multiplication des interfaces est tout juste amorti et un faible gain de temps est constaté (≈ 35 % à l'ordre $d = 1$ et ≈ 27 % à l'ordre $d = 2$). Cette méthode devrait cependant montrer de bien meilleures accélérations sur un cas tel que celui du corps humain complet qui présente un facteur d'échelle 10 fois plus important.

Pour répondre à la problématique des architectures hybrides et devant la complexité de certains graphes des tâches, nous avons décrit l'utilisation de la bibliothèque StarPU pour l'exécution et l'ordonnancement des tâches de calcul du solveur GD `schnaps` (chapitre 6). Les résultats sur CPU multicœurs se sont avérés excellents : une efficacité presque optimale quelque soit le nombre de cœurs sollicités et pour diverses tailles (divers raffinements) du problème étudié.

Les résultats en configuration hybride sont eux aussi de bonne qualité. Nous avons constaté que l'ajout des cœurs CPU au GPU a permis d'améliorer les performances du GPU seul. Dans le cas des plus petites tailles de problème, l'accélération constatée est au-delà de l'accélération théorique donnée par les temps sur CPU seul et GPU seul.

Nous sommes ensuite revenus sur le solveur `teta-clac` et avons présenté des résultats permettant de valider notre implémentation de solveur GD, autant par la précision des calculs que par leur vitesse d'exécution (chapitre 7).

Dans un premier temps, nous avons présenté des résultats de convergence et de précision sur un cas académique de propagation d'une onde plane dans un cube pour différents ordres d'interpolation. L'ordre de convergence théorique du schéma en temps a été vérifié en pratique et l'erreur globale commise reste négligeable. Ces résultats ont été obtenus sur maillages structurés et non structurés et démontrent le bon fonctionnement du solveur ainsi que sa

polyvalence du point de vue géométrique.

Nous avons ensuite comparé notre solveur GD à un solveur implémentant la méthode des différences finies (FD) sur un cas d'application mettant en scène une antenne dipolaire à proximité de l'oreille d'une tête humaine simplifiée. La grande quantité d'optimisations a permis à notre solveur GD de produire de hautes performances et de meilleurs temps de calcul que le solveur FD.

Nous avons enfin présenté les résultats d'une simulation exécutée sur le modèle de corps humain complet dans le cadre du projet HOROCH. Une simulation de 10 ns de temps physique en parallèle sur 8 GPU NVidia de dernière génération. Cette simulation démontre la capacité du solveur `teta-clac` à traiter des problèmes de rayonnement électromagnétique d'objets connectés placés à proximité (ou à l'intérieur) du corps humain.

Précisons que le temps de simulation sur corps humain complet donné à 11 jours est issu d'un premier essai. L'utilisation d'un coefficient CFL plus adaptée devrait permettre de réduire ce temps de restitution (facteur d'accélération proche de 5 dans le cas de la tête humaine simplifiée à l'ordre d'interpolation 1). La finalisation de l'implémentation MPI du schéma LTS2 devrait aussi permettre de constater un important gain de temps compte tenu du facteur d'échelle proche de 1000 entre la plus petite et la plus grande maille.

Cette simulation a aussi permis d'effectuer des tests de scalabilité MPI allant jusqu'à 256 GPU. Nous avons alors évalué qu'elle pouvait être réalisée en moins d'un jour sur 256 GPU. Ces tests témoignent du fait que le temps de calcul d'une simulation peut être compressé en investissant plus de ressources de calcul et ainsi passer de plus de 10 jours à seulement quelques heures d'attente avant la restitution des données. C'est l'amélioration constante du solveur GD `teta-clac` qui nous a permis d'obtenir ces résultats encore hypothétiques il y a 3 ans.

Annexe A

Existence de la solution du problème d'évolution

A.1 Théorie des opérateurs linéaires

Le théorème de Hille-Yosida est un théorème abstrait de la théorie des opérateurs linéaires qui permet de démontrer l'existence et l'unicité de la solution d'un problème d'évolution. Nous donnons les notions minimales pour comprendre l'énoncé de ce théorème. Pour plus de détails, nous renvoyons au livre de Brézis [8].

Nous considérons un espace de Hilbert H muni du produit scalaire $\langle \cdot, \cdot \rangle$. En général, $\boldsymbol{A}$ désignera un opérateur non-borné de H à domaine dense, c'est à dire une application linéaire d'un sous-espace vectoriel $D(\boldsymbol{A})$ de H à valeurs dans H. Le sous-espace vectoriel $D(\boldsymbol{A})$ est appelé domaine de $\boldsymbol{A}$. Si H est un Hilbert de dimension infinie, $D(\boldsymbol{A})$ n'est généralement pas fermé. Nous supposons que le domaine de $\boldsymbol{A}$ est dense, c'est à dire que $\overline{D(\boldsymbol{A})} = H$ pour la topologie forte de H. La topologie forte est la topologie associée à la norme du produit scalaire de H. Le graphe de $\boldsymbol{A}$, noté $G(\boldsymbol{A})$ est l'ensemble :

$$G(\boldsymbol{A}) = \{(\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}) : \boldsymbol{u} \in D(\boldsymbol{A})\} \subset H^2. \tag{A.1}$$

En général, $\overline{G(\boldsymbol{A})}$ n'est pas le graphe d'un opérateur univoque, c'est à dire que $(\boldsymbol{u}, \boldsymbol{f}) \in \overline{G(\boldsymbol{A})}$ et $(\boldsymbol{u}, \boldsymbol{g}) \in \overline{G(\boldsymbol{A})}$ n'implique pas forcément $\boldsymbol{f} = \boldsymbol{g}$. Si $\overline{G(\boldsymbol{A})}$ est le graphe d'un opérateur univoque, nous dirons que $\boldsymbol{A}$ est un opérateur non-borné fermable. Nous appellerons fermeture de $\boldsymbol{A}$ et nous noterons $\overline{\boldsymbol{A}}$ l'opérateur tel que $G(\overline{\boldsymbol{A}}) = \overline{G(\boldsymbol{A})}$.

Définition 8. *Soit H un espace de Hilbert de produit scalaire $\langle \cdot, \cdot \rangle$. Soit $\boldsymbol{A}$ un opérateur linéaire de $D(\boldsymbol{A})$, sous-espace vectoriel de H, dans H. $\boldsymbol{A}$ est*

dit ***monotone*** *si :*

$$\forall \boldsymbol{u} \in D(\boldsymbol{A}), \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{u} \rangle \geq 0. \tag{A.2}$$

L'opérateur $\boldsymbol{A}$ *est dit* ***maximal monotone*** *si de plus* $\boldsymbol{I} + \boldsymbol{A}$ *est surjectif de* $D(\boldsymbol{A})$ *sur* H*.*

Considérons le problème d'évolution abstrait d'inconnue $\boldsymbol{u}(t) \in D(\boldsymbol{A})$:

$$\partial_t \boldsymbol{u} + \boldsymbol{A}\boldsymbol{u} = 0, \tag{A.3a}$$

$$\boldsymbol{u}(0) = \boldsymbol{u}_0 \in D(\boldsymbol{A}). \tag{A.3b}$$

Le théorème de Hille-Yosida donne une condition pour que ce problème soit bien posé. Dans le cas qui nous intéresse, l'opérateur $\boldsymbol{A}$ sera formellement l'opérateur aux dérivées partielles en espace $\boldsymbol{A}_i \partial_i$ et son domaine $D(\boldsymbol{A})$ sera l'ensemble des $\boldsymbol{w}$ qui satisfont les conditions aux limites.

Théorème 5 (Hille-Yosida)**.** *Si* $\boldsymbol{A}$ *est maximal monotone, alors le problème* (A.3) *admet une unique solution* $\boldsymbol{u}$ *dans* $\mathcal{C}^1([0\,;T], H) \cap \mathcal{C}([0\,;T], D(\boldsymbol{A}))$ *et pour tout* t *:*

$$\|\boldsymbol{u}(t)\| \leq \|\boldsymbol{u}_0\|\,,\ \|\partial_t \boldsymbol{u}(t)\| = \|\boldsymbol{A}\boldsymbol{u}(t)\| \leq \|\boldsymbol{A}\boldsymbol{u}_0\|\,. \tag{A.4}$$

Remarque 35. $D(\boldsymbol{A})$ *est muni de la norme dite du graphe :*

$$\|\boldsymbol{u}\|_{D(\boldsymbol{A})} = \|\boldsymbol{u}\|_H + \|\boldsymbol{A}\boldsymbol{u}\|_H\,. \tag{A.5}$$

Remarque 36. *L'application qui à* $\boldsymbol{u}_0$ *associe* $\boldsymbol{u}(t)$ *peut donc être prolongée par densité en un semi-groupe de contractions de* H *dans* H*. Il est donc possible de définir des solutions du problème d'évolution lorsque* $\boldsymbol{u}_0 \in H$*.*

Nous allons maintenant introduire quelques outils qui permettront de démontrer la surjectivité de $\boldsymbol{I} + \boldsymbol{A}$. Les démonstrations seront basées sur des techniques d'intégration par parties et de passage à l'adjoint.

Définition 9. *Soit* $\boldsymbol{A}$ *un opérateur linéaire de* H *dans* H *de domaine* $D(\boldsymbol{A})$ *dense.* $\boldsymbol{A}^\#$ *est un* ***adjoint formel*** *de* $\boldsymbol{A}$ *si* $\boldsymbol{A}^\#$ *est à domaine dense et :*

$$\forall \boldsymbol{u} \in D(\boldsymbol{A}), \forall \boldsymbol{v} \in D(\boldsymbol{A}^\#),\ \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle = \left\langle \boldsymbol{u}, \boldsymbol{A}^\# \boldsymbol{v} \right\rangle. \tag{A.6}$$

Proposition 12. *Si* $\boldsymbol{A}$*, un opérateur linéaire de* H*, admet un adjoint formel* $\boldsymbol{A}^\#$*, alors* $\boldsymbol{A}$ *est fermable, c'est à dire que la fermeture du graphe de* $\boldsymbol{A}$ *dans* H^2*,* $\overline{G(\boldsymbol{A})}$*, définit un opérateur univoque.*

Démonstration. Soit $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ une suite définie sur $D(\boldsymbol{A})$ telle que $\boldsymbol{u}_n \to \boldsymbol{u}$. Soient $(\boldsymbol{f}_n)_{n\in\mathbb{N}}$ et $(\boldsymbol{g}_n)_{n\in\mathbb{N}}$ des suites définies telles que, pour tout n nous ayons $(\boldsymbol{u}_n, \boldsymbol{f}_n), (\boldsymbol{u}_n, \boldsymbol{g}_n) \in G(\boldsymbol{A})$ et, $\boldsymbol{f}_n \to \boldsymbol{f}$ et $\boldsymbol{g}_n \to \boldsymbol{g}$. Ainsi, $(\boldsymbol{u}, \boldsymbol{f})$ et $(\boldsymbol{u}, \boldsymbol{g})$ appartiennent à $\overline{G(\boldsymbol{A})}$. Puisque $\boldsymbol{A}$ admet un adjoint formel $\boldsymbol{A}^\#$, pour tout $\boldsymbol{v} \in \boldsymbol{A}^\#$ et tout n, nous avons :

$$\langle \boldsymbol{f}_n, \boldsymbol{v} \rangle = \langle \boldsymbol{g}_n, \boldsymbol{v} \rangle = \left\langle \boldsymbol{u}_n, \boldsymbol{A}^\# \boldsymbol{v} \right\rangle . \tag{A.7}$$

S'ensuit, par linéarité à gauche du produit scalaire :

$$\langle \boldsymbol{f}_n, \boldsymbol{v} \rangle - \langle \boldsymbol{g}_n, \boldsymbol{v} \rangle = \langle \boldsymbol{f}_n - \boldsymbol{g}_n, \boldsymbol{v} \rangle = 0, \tag{A.8}$$

et donc $\boldsymbol{f}_n = \boldsymbol{g}_n$. En passant à la limite nous obtenons le résultat souhaité. □

Définition 10. *Soit $\boldsymbol{A}$ un opérateur de H dans H à domaine dense. L'**adjoint** (non formel) de $\boldsymbol{A}$, noté $\boldsymbol{A}^\star$, est un opérateur de H dans H défini tel que :*

$$D(\boldsymbol{A}^\star) = \{v \in H : \exists C \geq 0 : \forall \boldsymbol{u} \in D(\boldsymbol{A}), |\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle| \leq C \|\boldsymbol{u}\|\} . \tag{A.9}$$

En d'autres termes, $D(\boldsymbol{A}^\star)$ est l'ensemble des $\boldsymbol{v} \in H$ tel que la forme linéaire $\varphi : \boldsymbol{u} \mapsto \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle$ est continue sur $D(\boldsymbol{A})$ pour la norme de H.
De plus, si $\boldsymbol{v} \in D(\boldsymbol{A}^\star)$, alors φ se prolonge (par densité de $D(\boldsymbol{A})$ dans H) en une forme linéaire continue sur H, c'est à dire un élément de l'espace dual de H. Comme H est un espace de Hilbert, par le théorème de représentation de Riesz, il existe un unique $\boldsymbol{f} \in H$ tel que $\langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle = \langle \boldsymbol{f}, \boldsymbol{u} \rangle$. Alors par définition $\boldsymbol{f} = \boldsymbol{A}^\star \boldsymbol{v}$.

Remarque 37. *En général, $\boldsymbol{A}^\star \neq \boldsymbol{A}^\#$. Cependant, pour tout adjoint formel $\boldsymbol{A}^\#$, $G(\boldsymbol{A}^\#) \subset G(\boldsymbol{A}^\star)$ et ainsi $D(\boldsymbol{A}^\#) \subset D(\boldsymbol{A}^\star)$.*

Définition 11. *Soit $\boldsymbol{A}$ un opérateur fermable. Soit $\boldsymbol{u}$ un élément de H. $\boldsymbol{u}$ est **solution forte** de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ si $\overline{\boldsymbol{A}}\boldsymbol{u} = \boldsymbol{f}$.*

Une solution forte $\boldsymbol{u}$ de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ est telle que $(\boldsymbol{u}, \boldsymbol{f}) \in \overline{G(\boldsymbol{A})} = G(\overline{\boldsymbol{A}})$. En d'autres termes, $\boldsymbol{u}$ est solution forte de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ si, et seulement si, il existe une suite $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ d'éléments de $D(\boldsymbol{A})$ telle que $\boldsymbol{u}_n \to \boldsymbol{u}$ et $\boldsymbol{A}\boldsymbol{u}_n \to \boldsymbol{f}$.

Définition 12. *Soit $\boldsymbol{u}$ un élément de H. $\boldsymbol{u}$ est **solution faible** de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ si $\boldsymbol{A}$ admet un adjoint formel $\boldsymbol{A}^\#$ et :*

$$\forall \boldsymbol{v} \in D(\boldsymbol{A}^\#), \langle \boldsymbol{f}, \boldsymbol{v} \rangle = \left\langle \boldsymbol{u}, \boldsymbol{A}^\# \boldsymbol{v} \right\rangle . \tag{A.10}$$

Proposition 13. *Si $\boldsymbol{A}$ admet un adjoint formel, alors toute solution forte de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ est aussi solution faible.*

Démonstration. Soit $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ une suite d'éléments de $D(\boldsymbol{A})$ telle que $\boldsymbol{u}_n \to \boldsymbol{u}$ et $\boldsymbol{A}\boldsymbol{u}_n \to \boldsymbol{f}$. Puisque $\boldsymbol{A}$ admet un adjoint formel $\boldsymbol{A}^\#$, pour tout $\boldsymbol{v} \in \boldsymbol{A}^\#$ et tout n, nous avons :

$$\langle \boldsymbol{A}\boldsymbol{u}_n, \boldsymbol{v}\rangle = \left\langle \boldsymbol{u}_n, \boldsymbol{A}^\#\boldsymbol{v}\right\rangle . \tag{A.11}$$

En passant à la limite nous obtenons le résultat souhaité. □

Proposition 14. *Si $\boldsymbol{A}$ et $\boldsymbol{A}^\#$ sont deux opérateurs à domaines denses, adjoints formels et si toute solution faible de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ est aussi une solution forte, alors $\overline{\boldsymbol{A}} = (\boldsymbol{A}^\#)^\star$ et $\overline{\boldsymbol{A}^\#} = (\overline{\boldsymbol{A}})^\star$.*

Démonstration. Montrer l'égalité d'opérateurs revient à démontrer l'égalité des graphes. Montrons d'abord que $G(\overline{\boldsymbol{A}}) \subset G((\boldsymbol{A}^\#)^\star)$. Soit $(\boldsymbol{u}, \boldsymbol{f}) \in G(\overline{\boldsymbol{A}})$, alors $\boldsymbol{u}$ est solution forte de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$. Il s'ensuit que $\boldsymbol{u}$ est aussi solution faible c'est à dire :

$$\forall \boldsymbol{v} \in D(\boldsymbol{A}^\#), \left\langle \boldsymbol{u}, \boldsymbol{A}^\#\boldsymbol{v}\right\rangle = \langle \boldsymbol{f}, \boldsymbol{v}\rangle . \tag{A.12}$$

La forme linéaire $\boldsymbol{v} \mapsto \left\langle \boldsymbol{u}, \boldsymbol{A}^\#\boldsymbol{v}\right\rangle = \langle \boldsymbol{f}, \boldsymbol{v}\rangle$ est donc continue. Nous avons aussi $\boldsymbol{u} \in D((\boldsymbol{A}^\#)^\star)$ et $(\boldsymbol{A}^\#)^\star\boldsymbol{u} = \boldsymbol{f}$ donc $(\boldsymbol{u}, \boldsymbol{f}) \in G((\boldsymbol{A}^\#)^\star)$.
Montrons que $G((\boldsymbol{A}^\#)^\star) \subset G(\overline{\boldsymbol{A}})$. En effet, soit $(\boldsymbol{u}, \boldsymbol{f}) \in G((\boldsymbol{A}^\#)^\star)$. Alors la forme linéaire $\boldsymbol{v} \in D(\boldsymbol{A}^\#) \mapsto \left\langle \boldsymbol{u}, \boldsymbol{A}^\#\boldsymbol{v}\right\rangle = \langle \boldsymbol{f}, \boldsymbol{v}\rangle$ est continue et :

$$\left\langle (\boldsymbol{A}^\#)^\star\boldsymbol{u}, \boldsymbol{v}\right\rangle = \langle \boldsymbol{f}, \boldsymbol{v}\rangle = \left\langle \boldsymbol{u}, \boldsymbol{A}^\#\boldsymbol{v}\right\rangle . \tag{A.13}$$

Par conséquent, $\boldsymbol{u}$ est solution faible de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$. Comme toute solution faible est forte, $\boldsymbol{u}$ est aussi solution forte et donc $(\boldsymbol{u}, \boldsymbol{f}) \in G(\overline{\boldsymbol{A}})$.
La seconde égalité découle de résultats généraux sur les adjoints d'opérateurs dans les espaces de Hilbert. D'abord l'adjoint d'un opérateur fermable est égal à l'adjoint de sa fermeture donc $\overline{\boldsymbol{A}} = (\boldsymbol{A}^\#)^\star = (\overline{\boldsymbol{A}^\#})^\star$. D'autre part, l'adjoint de l'adjoint d'un opérateur fermé à domaine dense est l'opérateur lui-même donc $(\overline{\boldsymbol{A}})^\star = \overline{\boldsymbol{A}^\#}$.

□

Proposition 15. *Si $\boldsymbol{A}$ est un opérateur à domaine dense fermable, alors $\boldsymbol{A}^\star = (\overline{\boldsymbol{A}})^\star$.*

Démonstration. Nous procédons à nouveau sur les graphes. D'une part, $G(\boldsymbol{A}^\star) \subset G((\overline{\boldsymbol{A}})^\star)$. En effet, soit $(\boldsymbol{v}, \boldsymbol{A}^\star\boldsymbol{v}) \in G(\boldsymbol{A}^\star)$. Alors la forme linéaire $\boldsymbol{u} \mapsto \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v}\rangle = \langle \boldsymbol{u}, \boldsymbol{A}^\star\boldsymbol{v}\rangle$ est linéaire continue sur $D(\boldsymbol{A})$ et donc aussi sur $D(\overline{\boldsymbol{A}})$, en prolongeant par continuité, et donc $(\boldsymbol{v}, \boldsymbol{A}^\star\boldsymbol{v}) \in G((\overline{\boldsymbol{A}})^\star)$. De même, $G((\overline{\boldsymbol{A}})^\star) \subset G(\boldsymbol{A}^\star)$ car $D(\boldsymbol{A}) \subset D(\overline{\boldsymbol{A}})$.

□

Proposition 16. *Soit $\boldsymbol{A}$ un opérateur fermé à domaine dense tel que $D(\boldsymbol{A}^\star)$ est dense. Alors $\boldsymbol{A}^{\star\star} = \boldsymbol{A}$.*

Ce résultat repose sur des propriétés de la topologie faible des espaces de Hilbert. Une suite $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ d'éléments de H converge faiblement vers $\boldsymbol{u}$ si, et seulement si, pour tout $\boldsymbol{v} \in H$, $\langle \boldsymbol{u}_n, \boldsymbol{v}\rangle \to \langle \boldsymbol{u}, \boldsymbol{v}\rangle$. La convergence forte implique la convergence faible, mais la réciproque est fausse. Nous avons donc à notre disposition deux topologies sur H : la topologie forte associée à la convergence forte des suites, au sens de la norme de H, et la topologie faible associée à la convergence faible.

Considérons maintenant un ensemble $K \subset H$. Si K est fermé pour la topologie faible, alors K est aussi fermé pour la topologie forte. En général, la réciproque est fausse, si K est fermé pour la topologie forte il n'est pas forcément fermé pour la topologie faible. Un résultat fondamental de l'analyse fonctionnelle est le suivant :

Théorème 6. *Soit K un convexe fermé de H pour la topologie forte. Alors K est fermé pour la topologie faible.*

Utilisons ce théorème pour démontrer la proposition 16.

Démonstration. Soit $(\boldsymbol{u}, \boldsymbol{A}\boldsymbol{u}) \in G(\boldsymbol{A})$. Alors :

$$\forall \boldsymbol{v} \in D(\boldsymbol{A}^\star), \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v}\rangle = \langle \boldsymbol{u}, \boldsymbol{A}^\star\boldsymbol{v}\rangle\,. \tag{A.14}$$

Par conséquent, la forme linéaire $\boldsymbol{v} \mapsto \langle \boldsymbol{u}, \boldsymbol{A}^\star\boldsymbol{v}\rangle$ est continue et $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}^{\star\star}\boldsymbol{u}$ et donc $(\boldsymbol{u}, \boldsymbol{A}\ U) \in G(\boldsymbol{A}^{\star\star})$.
Réciproquement, soit $(\boldsymbol{u}, \boldsymbol{A}^{\star\star}\boldsymbol{u}) \in G(\boldsymbol{A}^{\star\star})$. Pour tout $\boldsymbol{v} \in D(\boldsymbol{A}^\star)$ la forme linéaire $\boldsymbol{v} \mapsto \langle \boldsymbol{u}, \boldsymbol{A}^\star\boldsymbol{v}\rangle = \langle \boldsymbol{A}^{\star\star}\boldsymbol{u}, \boldsymbol{v}\rangle$ est continue. Or, $D(\boldsymbol{A})$ est dense dans H, il existe donc une suite $(\boldsymbol{u}_n)_{n\in\mathbb{N}}$ d'éléments de $D(\boldsymbol{A})$ telle que $\boldsymbol{u}_n \to \boldsymbol{u}$. Nous avons alors d'après la définition de l'adjoint $\boldsymbol{A}^\star$ que $\langle \boldsymbol{u}_n, \boldsymbol{A}^\star\boldsymbol{v}\rangle = \langle \boldsymbol{A}\boldsymbol{u}_n, \boldsymbol{v}\rangle$. Il s'ensuit que $\langle \boldsymbol{A}\boldsymbol{u}_n, \boldsymbol{v}\rangle \to \langle \boldsymbol{A}^{\star\star}\boldsymbol{u}, \boldsymbol{v}\rangle$. Comme $D(\boldsymbol{A}^\star)$ est dense, nous en déduisons que $\boldsymbol{A}\boldsymbol{u}_n$ tend vers $\boldsymbol{A}^{\star\star}\boldsymbol{u}$ pour la topologie faible. □

Lemme 1. *Soit $\boldsymbol{A}$ un opérateur de H à domaine dense. Les propriétés suivantes sont équivalentes :*

1. *$\boldsymbol{A}$ est surjectif ;*
2. *$\exists C \geq 0 : \forall \boldsymbol{v} \in D(\boldsymbol{A}^\star), \|\boldsymbol{v}\| \leq C\,\|\boldsymbol{A}^\star\boldsymbol{v}\|$;*
3. *$\ker(\boldsymbol{A}^\star) = \{0\}$ et l'image de $\boldsymbol{A}^\star$ est fermée.*

Ce lemme est une généralisation fondamentale d'un résultat bien connu d'algèbre linéaire en dimension finie : pour montrer l'existence de la solution d'un problème linéaire il suffit de démontrer l'unicité de la solution du problème adjoint. Pour la démonstration, nous renvoyons au livre de Brézis [8].

Théorème 7. *Si $\boldsymbol{A}$ et $\boldsymbol{A}^{\#}$ sont adjoints formels, monotones et si toute solution faible de $\boldsymbol{Au} = \boldsymbol{f}$ est forte, alors $\overline{\boldsymbol{A}}$ et $\overline{\boldsymbol{A}^{\#}}$ sont maximaux monotones.*

Démonstration. Pour tout $\boldsymbol{u} \in D(\boldsymbol{A})$, $\langle (\boldsymbol{I} + \boldsymbol{A})\boldsymbol{u}, \boldsymbol{u}\rangle \geq \langle \boldsymbol{u}, \boldsymbol{u}\rangle$ car $\boldsymbol{A}$ est monotone. Par densité, nous montrons aussi que pour tout $\boldsymbol{u} \in D(\overline{\boldsymbol{A}})$, $\left\langle (\boldsymbol{I} + \overline{\boldsymbol{A}})\boldsymbol{u}, \boldsymbol{u}\right\rangle \geq \langle \boldsymbol{u}, \boldsymbol{u}\rangle$. En appliquant l'inégalité de Cauchy-Schwarz nous obtenons :

$$\forall \boldsymbol{u} \in D(\overline{\boldsymbol{A}}), \left\|(\boldsymbol{I} + \overline{\boldsymbol{A}})\boldsymbol{u}\right\| \geq \|\boldsymbol{u}\|\,. \tag{A.15}$$

De même :

$$\forall \boldsymbol{v} \in D(\overline{\boldsymbol{A}^{\#}}), \left\|(\boldsymbol{I} + \overline{\boldsymbol{A}^{\#}})\boldsymbol{v}\right\| \geq \|\boldsymbol{v}\|\,. \tag{A.16}$$

Mais puisque $\overline{\boldsymbol{A}^{\#}} = (\overline{\boldsymbol{A}})^{\star}$, il s'ensuit :

$$\forall \boldsymbol{v} \in D((\overline{\boldsymbol{A}})^{\star}), \left\|(\boldsymbol{I} + (\overline{\boldsymbol{A}})^{\star})\boldsymbol{v}\right\| \geq \|\boldsymbol{v}\|\,. \tag{A.17}$$

Donc, en appliquant le lemme 1, $\boldsymbol{I} + \overline{\boldsymbol{A}}$ et $\boldsymbol{I} + (\overline{\boldsymbol{A}})^{\star}$ sont surjectifs. Nous obtenons bien que $\overline{\boldsymbol{A}}$ et $\overline{\boldsymbol{A}^{\#}}$ sont maximaux monotones. □

A.2 Application au problème d'évolution

Dans cette section nous supposons que Ω est un ouvert de bord $\partial\Omega$ régulier. Pour x un point du bord $\partial\Omega$, nous définissons $V_b(x)$ un espace vectoriel de dimension $q \leq m$. Les conditions aux limites en x seront de la forme $\boldsymbol{w}(x,t) \in V_b(x)$.

Nous considérons le problème d'évolution (1.14) pour un système de Friedrichs (1.5). Afin de simplifier les écritures, nous considérons le problème sans sources ($\boldsymbol{C} = 0$ et $\boldsymbol{S} = 0$) et nous supposerons que la matrice $\boldsymbol{A}_t$ est égale à l'identité.

Rappelons alors la formulation (simplifiée) du problème d'évolution :

$$\begin{aligned} \partial_t \boldsymbol{w} + \boldsymbol{A}_i \partial_i \boldsymbol{w} = 0 \quad & \text{sur } Q, \\ \boldsymbol{w}(x,0) = \boldsymbol{w}_0(x) \quad & \text{sur } \Omega, \\ \boldsymbol{w} \in V_b \quad & \text{sur } \partial Q. \end{aligned}$$

Rappelons aussi que dans un tel système, les matrices $\boldsymbol{A}_i$ sont symétriques et donc que le système est bien hyperbolique. Nous choisissons $H = \mathrm{L}^2(\Omega)$. Nous prenons :

$$D(\boldsymbol{A}) = \left\{\boldsymbol{u} \in \mathcal{C}^1(\overline{\Omega}) : \forall x \in \partial\Omega, \boldsymbol{u}(x) \in V_b(x)\right\}, \tag{A.19}$$

et si $\boldsymbol{u} \in D(\boldsymbol{A})$, alors $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{A}_i \partial_i \boldsymbol{u}$.

Pour le problème adjoint, nous commençons par définir une condition aux limites adjointe $V_b^{\#}(x) = (\boldsymbol{A}_i n_i V_b(x))^{\top}$. Nous prenons :

$$D(\boldsymbol{A}^{\#}) = \left\{ \boldsymbol{v} \in \mathcal{C}^1(\overline{\Omega}) : \forall x \in \partial\Omega, \boldsymbol{v}(x) \in V_b^{\#}(x) \right\}, \tag{A.20}$$

et si $\boldsymbol{v} \in D(\boldsymbol{A}^{\#})$, alors $\boldsymbol{A}^{\#}\boldsymbol{v} = -\boldsymbol{A}_i \partial_i \boldsymbol{v}$.

Proposition 17. *Les opérateurs $\boldsymbol{A}$ et $\boldsymbol{A}^{\#}$ ainsi définis sont adjoints formels.*

Démonstration. Il suffit de montrer que :

$$\forall \boldsymbol{u} \in D(\boldsymbol{A}), \forall \boldsymbol{v} \in D(\boldsymbol{A}^{\#}), \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle = \left\langle \boldsymbol{u}, \boldsymbol{A}^{\#}\boldsymbol{v} \right\rangle. \tag{A.21}$$

En écrivant le premier terme à l'aide du produit scalaire défini sur $\mathrm{L}^2(\Omega)$, puis en intégrant par parties, nous obtenons :

$$\int_\Omega \boldsymbol{A}_i \partial_i \boldsymbol{u} \cdot \boldsymbol{v} dx = -\int_\Omega \boldsymbol{u} \cdot \boldsymbol{A}_i \partial_i \boldsymbol{v} dx + \int_{\partial\Omega} \boldsymbol{A}_i n_i \boldsymbol{u} \cdot \boldsymbol{v} ds. \tag{A.22}$$

Or, par définition de V_b et $V_b^{\#}$, le terme de bord apparaissant dans le second membre est nul. Nous avons donc bien l'égalité recherchée. □

Le théorème qui suit est issu des travaux de Lax-Phillips [51] et Rauch [60]. Ce résultat est assez technique mais fondamental.

Théorème 8 (Lax-Phillips-Rauch)**.** *Soit Ω un domaine de $\mathbb{R}^k$ de frontière $\partial\Omega$ de classe $\mathcal{C}^1$. Pour $x \in \partial\Omega$, notons $\boldsymbol{n}(x)$ le vecteur normal à $\partial\Omega$ en x. Nous supposons que $x \mapsto V_b(x)$ est une application Lipschitzienne par rapport à x. Si $\dim \ker \boldsymbol{A}_i n_i(x)$ est constant sur chaque composante connexe de $\partial\Omega$ et si pour tout $x \in \partial\Omega$, $\ker \boldsymbol{A}_i n_i(x) \subset V_b(x)$, alors toute solution faible de $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ est aussi une solution forte.*

Dans cet énoncé, remarquons que l'espace vectoriel $V_b(x)$ peut être défini par une base :

$$V_b(x) = \mathrm{vect}\{\boldsymbol{b}_1(x), \ldots, \boldsymbol{b}_q(x)\}. \tag{A.23}$$

Dire que $V_b(x)$ est Lipschitzien par rapport à x signifie que les applications $x \mapsto \boldsymbol{b}_i(x)$ sont Lipschitziennes. L'espace $V_b(x)$ peut aussi être défini au moyen du noyau d'une matrice $\boldsymbol{M}(x)$ telle que :

$$\boldsymbol{w} \in V_b(x) \Leftrightarrow \boldsymbol{M}(x)\boldsymbol{w} = 0. \tag{A.24}$$

Dans ce cas, c'est $x \mapsto \boldsymbol{M}(x)$ qui est Lipschitzienne.

Définition 13. *L'espace vectoriel des conditions aux limites V_b est dit* ***positif*** *par rapport à $\boldsymbol{A}_i n_i$ si :*

$$\forall \boldsymbol{u} \in V_b, \langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{u} \rangle \geq 0. \tag{A.25}$$

Il est dit ***maximal positif*** *si de plus la dimension de V_b est égale au nombre de valeurs propres positives ou nulles de $\boldsymbol{A}_i n_i$ en comptant leurs multiplicités.*

Nous allons vérifier que la positivité maximale exprime qu'il n'existe pas d'espace vectoriel positif contenant V_b et strictement plus grand que V_b. D'où la notion de maximalité. Nous allons aussi montrer que si V_b est maximal positif, alors c'est aussi le cas pour $V_b^{\#}$.

Comme $\boldsymbol{A}_i n_i$ est une matrice symétrique, elle admet une base orthonormée de vecteurs propres. Dans un premier temps, nous allons considérer le cas où $\boldsymbol{A}_i n_i$ est inversible.

Si $\boldsymbol{A}_i n_i$ n'est pas inversible, on peut se ramener au cas inversible avec un argument d'espace vectoriel quotienté par le noyau de $\boldsymbol{A}_i n_i$ [60]. Notons que dans le cas des équations de Maxwell, $\boldsymbol{A}_i n_i$ n'est pas inversible...

Nous supposons donc que les valeurs propres $(\lambda_i)_{i \in [\![1\,;m]\!]}$ sont rangées de la façon suivante :

$$\lambda_1 \geq \ldots \geq \lambda_p > 0 > \lambda_{p+1} \geq \ldots \geq \lambda_m. \tag{A.26}$$

Par conséquent, l'entier p est le nombre de valeurs propres positives de $\boldsymbol{A}_i n_i$ en tenant compte de leurs multiplicités. Nous notons $(\boldsymbol{r}_i)_{i \in [\![1\,;m]\!]}$ une base orthonormée de vecteurs propres correspondants. La matrice $\boldsymbol{P} = (\boldsymbol{r}_1 \ \ldots \ \boldsymbol{r}_m)$ est orthogonale (*i.e.* $\boldsymbol{P}^{-1} = \boldsymbol{P}^{\mathrm{T}}$) et :

$$\boldsymbol{D} = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{A}_i n_i \boldsymbol{P} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m). \tag{A.27}$$

Remarquons alors que dans la nouvelle base, le produit scalaire $\langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{v} \rangle$ devient $\langle \boldsymbol{D}\boldsymbol{u}', \boldsymbol{v}' \rangle$, avec $\boldsymbol{u}' = \boldsymbol{P}\boldsymbol{u}$ et $\boldsymbol{v}' = \boldsymbol{P}\boldsymbol{v}$.

Nous notons E_+ l'espace vectoriel engendré par $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_p\}$ et E_- l'espace engendré par $\{\boldsymbol{r}_{p+1}, \ldots, \boldsymbol{r}_m\}$. Par conséquent, $\dim E_+ = p$ et $\dim E_- = m - p$. De plus, $\mathbb{R}^m = E_+ \oplus E_-$ et $\boldsymbol{A}_i n_i E_\pm = E_\pm$. Nous notons Π_+, respectivement Π_-, la projection orthogonale sur l'espace vectoriel E_+, respectivement E_-.

Lemme 2. *Si V_b est un espace positif par rapport à $\boldsymbol{A}_i n_i$, alors $V_b \cap E_- = \{0\}$ et $q = \dim V_b \leq p$.*

Démonstration. $V_b \cap E_- = \{0\}$ découle du fait que :

$$\forall \boldsymbol{u} \in E_- \setminus \{0\}, \langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{u} \rangle \leq \max_{k \in [\![p+1\,;m]\!]} \lambda_k \left\| \boldsymbol{u} \right\|^2 < 0. \tag{A.28}$$

Raisonnons par l'absurde et supposons que $\dim V_b = q > p$. Considérons une base $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_q)$ de V_b. Il existe des réels $\alpha_1, \ldots, \alpha_q$ non tous nuls tels que :

$$\sum_{i=1}^{q} \alpha_i \Pi_+ \boldsymbol{b}_i = 0. \tag{A.29}$$

En effet, $\{\Pi_+\boldsymbol{b}_1, \ldots, \Pi_+\boldsymbol{b}_q\}$ est un ensemble de q vecteurs dans E_+, espace de dimension $p < q$. Considérons alors le vecteur $\boldsymbol{u} = \sum_{i=1}^{q} \alpha_i \boldsymbol{b}_i$. Par construction, $\boldsymbol{u} \in V_b \cap E_-$, donc $\boldsymbol{u} = 0$. Comme les $\boldsymbol{b}_i$ sont des vecteurs linéairement indépendants, les α_i sont tous nuls, ce qui conduit à une contradiction. □

La condition aux limites V_b et la condition aux limites adjointe $V_b^\#$ jouent des rôles symétriques. Lorsque $\boldsymbol{A}_i n_i$ est inversible, comme $V_b^\# = (\boldsymbol{A}_i n_i V_b)^{\mathrm{T}}$, nous avons $\dim V_b^\# = m - \dim V_b$. Nous en déduisons le lemme suivant.

Lemme 3. *Si $V_b^\#$ est un espace positif par rapport à $-\boldsymbol{A}_i n_i$, alors $V_b^\# \cap E_+ = \{0\}$ et $\dim V_b^\# \leq m - p$.*

Une conséquence intéressante des lemmes 2 et 3 est que si les conditions aux limites V_b et $V_b^\#$ sont positives alors elles sont toutes les deux aussi maximales puisque $\dim V_b + \dim V_b^\# = m$. De ce qui précède nous pouvons déduire la suite.

Lemme 4. *Si V_b est un espace maximal positif par rapport à $\boldsymbol{A}_i n_i$, alors $V_b^\#$ est aussi maximal positif par rapport à $-\boldsymbol{A}_i n_i$.*

Démonstration. Supposons que V_b est maximal positif pour $\boldsymbol{A}_i n_i$. Montrons alors que $V_b^\#$ est maximal positif pour $-\boldsymbol{A}_i n_i$. Sinon, il existerait $\boldsymbol{v} \in V_b^\#$ tel que $\langle \boldsymbol{A}_i n_i \boldsymbol{v}, \boldsymbol{v} \rangle > 0$. De plus, $\boldsymbol{v}$ ne peut pas aussi appartenir à V_b car sinon nous aurions $\langle \boldsymbol{A}_i n_i \boldsymbol{v}, \boldsymbol{v} \rangle = 0$.
Considérons l'espace vectoriel $U = \{\boldsymbol{u} + t\boldsymbol{v} : \boldsymbol{u} \in V_b, t \in \mathbb{R}\}$. U contient V_b et, puisque $\boldsymbol{v} \notin V_b$, $\dim U = 1 + \dim V_b = p + 1$. Montrons que U est un espace positif. Soit $\boldsymbol{w} = \boldsymbol{u} + t\boldsymbol{v} \in U$ avec $t \in \mathbb{R}$ et $\boldsymbol{u} \in V_b$ alors :

$$\begin{aligned} &\langle \boldsymbol{A}_i n_i (\boldsymbol{u} + t\boldsymbol{v}), (\boldsymbol{u} + t\boldsymbol{v}) \rangle \\ &= \langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{u} \rangle + 2t \langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{v} \rangle + t^2 \langle \boldsymbol{A}_i n_i \boldsymbol{v}, \boldsymbol{v} \rangle \\ &= \langle \boldsymbol{A}_i n_i \boldsymbol{u}, \boldsymbol{u} \rangle + t^2 \langle \boldsymbol{A}_i n_i \boldsymbol{v}, \boldsymbol{v} \rangle \geq 0, \end{aligned} \tag{A.30}$$

donc U est bien un espace positif. D'après le lemme 2, nous devrions alors avoir $\dim U \leq p$. Or, $\dim U = p + 1$, ce qui conduit à une contradiction. Par conséquent, $V_b^\#$ est bien positif pour $-\boldsymbol{A}_i n_i$. D'autre part, $\dim V_b^\# = m - \dim V_b = m - p$ et donc $V_b^\#$ est aussi maximal positif. □

Nous pouvons maintenant regrouper les résultats de cette section dans le théorème suivant.

Théorème 9. *Supposons que V_b est un espace maximal positif par rapport à $\boldsymbol{A}_i n_i$. Supposons aussi que les hypothèses du théorème 8 soient satisfaites. Alors les opérateurs $\overline{A}$ et $\overline{A^{\#}}$ sont maximaux monotones.*

Ce théorème assure que si les conditions aux limites sont maximales positives alors les problèmes d'évolution :

$$\partial_t \boldsymbol{u} + \boldsymbol{A}\boldsymbol{u} = 0, \tag{A.31a}$$
$$\boldsymbol{u}(\cdot, 0) = \boldsymbol{u}_0(\cdot) \tag{A.31b}$$

et

$$\partial_t \boldsymbol{v} + \boldsymbol{A}^{\#}\boldsymbol{v} = 0, \tag{A.32a}$$
$$\boldsymbol{v}(\cdot, T) = \boldsymbol{v}_0(\cdot) \tag{A.32b}$$

sont bien posés.

Annexe B

Démonstration de la convergence du schéma semi-discret

Rappelons l'énoncé du théorème de convergence 4 :

Théorème. *La solution du problème semi-discret* (1.59) *converge vers la solution du problème continu* (1.14) *lorsque* h *tend vers zéro.*

Nous allons démontrer ce théorème dans le cas d'une condition de bord de type Silver-Müller (2.1.4) représentée par la matrice :

$$\boldsymbol{M}_{\mathrm{SM}} = -\boldsymbol{A}_i {n_i}^{-}. \tag{B.1}$$

Cette condition aux limites correspond à l'application du flux de bord :

$$F_{\mathrm{SM}}\left(\boldsymbol{w}, \boldsymbol{n}\right) = (\boldsymbol{A}_i n_i + \boldsymbol{M})\boldsymbol{w} = \boldsymbol{A}_i {n_i}^{+}\boldsymbol{w}. \tag{B.2}$$

Cette démonstration de la convergence de la méthode GD est une adaptation de celle qui est présentée dans les travaux de Lesaint *et al.* [52] et Johnson *et al.* [42]. Nous commençons par définir la forme bilinéaire B définie sur $\mathcal{H}_h^m \times \mathcal{H}_h^m$ (1.57) et à valeurs dans $\mathbb{R}$:

$$\begin{aligned} B(\boldsymbol{\varphi}, \boldsymbol{\psi}) = & \int_{\Omega \setminus \Sigma_h} (\partial_i \boldsymbol{A}_i \boldsymbol{\varphi}) \cdot \boldsymbol{\psi} dx \\ & + \int_{\Sigma_h \setminus \partial\Omega} (\boldsymbol{A}_i {n_i}^{-} (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L)) \cdot \boldsymbol{\psi}_L ds \\ & + \int_{\Sigma_h \setminus \partial\Omega} (\boldsymbol{A}_i {n_i}^{+} (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L)) \cdot \boldsymbol{\psi}_R ds \\ & - \int_{\partial\Omega} (\boldsymbol{A}_i {n_i}^{-} \boldsymbol{\varphi}_L) \cdot \boldsymbol{\psi}_L ds, \end{aligned} \tag{B.3}$$

avec h le paramètre de finesse du maillage et Σ_h l'union des interfaces entre les mailles de ce maillage.

B est une forme bilinéaire définie positive et, en convenant que $\boldsymbol{\varphi}_R = \boldsymbol{\psi}_R = 0$ sur $\partial\Omega$ puis en reprenant les étapes de la section 1.3.3, nous obtenons :

$$B(\boldsymbol{\varphi}, \boldsymbol{\varphi}) = \frac{1}{2} \int_{\Sigma_h} (|\boldsymbol{A}_i n_i| (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L)) \cdot (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L) ds. \tag{B.4}$$

Avec cette forme bilinéaire, le problème semi-discret peut être mis sous la forme : trouver $\mathbf{w}$ dans $\mathcal{C}^1([0\,;T], \mathcal{H}_h^m)$ tel que pour tout $\boldsymbol{\psi}$ dans $\mathcal{C}^1([0\,;T], \mathcal{H}_h^m)$:

$$\int_\Omega \partial_t \mathbf{w} \cdot \boldsymbol{\psi} dx + B(\mathbf{w}, \boldsymbol{\psi}) = 0, \tag{B.5a}$$

$$\int_\Omega \mathbf{w}(x,0) \cdot \boldsymbol{\psi} dx = \int_\Omega \boldsymbol{w}_0 \cdot \boldsymbol{\psi} dx. \tag{B.5b}$$

Cette dernière formulation est équivalente à celle présentée précédemment (1.59), mais il est plus aisé de considérer des fonctions tests vectorielles plutôt que scalaires dans la démonstration.

Nous pouvons également mettre le problème continu (1.48) sous cette forme :

$$\int_\Omega \partial_t \boldsymbol{w} \cdot \boldsymbol{\psi} dx + B(\boldsymbol{w}, \boldsymbol{\psi}) = 0, \tag{B.6a}$$

$$\int_\Omega \boldsymbol{w}(x,0) \cdot \boldsymbol{\psi} dx = \int_\Omega \boldsymbol{w}_0 \cdot \boldsymbol{\psi} dx. \tag{B.6b}$$

Afin de démontrer le théorème de convergence 4, nous utiliserons le résultat suivant.

Lemme 5. *Il existe une constante positive λ telle que pour tout $\boldsymbol{\varphi}$ et $\boldsymbol{\psi}$ appartenant à $\mathrm{L}^2(\Sigma_h)$:*

$$\left| \int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) \cdot \boldsymbol{\psi} ds \right| \leq \sqrt{2\lambda B(\boldsymbol{\varphi}, \boldsymbol{\varphi})} \, \|\boldsymbol{\psi}\|_{\mathrm{L}^2(\Sigma_h)}. \tag{B.7}$$

Démonstration. L'inégalité de Cauchy-Schwarz donne la majoration :

$$\begin{aligned} \left| \int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) \cdot \boldsymbol{\psi} ds \right| &\leq \|\boldsymbol{A}_i n_i \boldsymbol{\varphi}\|_{\mathrm{L}^2(\Sigma_h)} \|\boldsymbol{\psi}\|_{\mathrm{L}^2(\Sigma_h)} \\ &= \left(\int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) \cdot (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) ds \right)^{\frac{1}{2}} \|\boldsymbol{\psi}\|_{\mathrm{L}^2(\Sigma_h)}. \end{aligned} \tag{B.8}$$

Puisque $\boldsymbol{A}_i n_i$ est symétrique, elle est diagonalisable en base orthonormée, *i.e.* il existe une matrice orthogonale $\boldsymbol{P}$ et une matrice diagonale $\boldsymbol{D}$ telles que $\boldsymbol{A}_i n_i = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\mathrm{T}}$. Alors,

$$\begin{aligned}(\boldsymbol{A}_i n_i \boldsymbol{\varphi}) \cdot (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) &= ((\boldsymbol{A}_i n_i)^2 \boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} \\ &= (\boldsymbol{P}\boldsymbol{D}^2\boldsymbol{P}^{\mathrm{T}} \boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} \\ &\leq (\boldsymbol{P}\,|\lambda|\,|\boldsymbol{D}|\,\boldsymbol{P}^{\mathrm{T}} \boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} \\ &= |\lambda|\,(\boldsymbol{P}\,|\boldsymbol{D}|\,\boldsymbol{P}^{\mathrm{T}} \boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} \\ &= \lambda(|\boldsymbol{A}_i n_i|\,\boldsymbol{\varphi}) \cdot \boldsymbol{\varphi},\end{aligned} \tag{B.9}$$

avec $\lambda > 0$ la plus grande valeur propre de $\boldsymbol{A}_i n_i$. Dans le cas des équations de Maxwell, cette valeur propre est la vitesse de la lumière, notée c. Nous avons donc :

$$\begin{aligned}\left| \int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}) \cdot \boldsymbol{\psi} ds \right| &\leq \left(c \int_{\Sigma_h} (|\boldsymbol{A}_i n_i|\,\boldsymbol{\varphi}) \cdot \boldsymbol{\varphi} ds \right)^{\frac{1}{2}} \|\boldsymbol{\psi}\|_{\mathrm{L}^2(\Sigma_h)} \\ &= \sqrt{2cB(\boldsymbol{\varphi}, \boldsymbol{\varphi})}\, \|\boldsymbol{\psi}\|_{\mathrm{L}^2(\Sigma_h)}\,.\end{aligned} \tag{B.10}$$

□

Nous pouvons maintenant démontrer le théorème 4.

Démonstration du théorème 4. Notons π la projection L^2 sur l'espace d'approximation $\mathcal{H}_h^m$:

$$\forall \boldsymbol{\varphi} \in \mathcal{H}_h^m, \int_\Omega \boldsymbol{w}\boldsymbol{\varphi} = \int_\Omega \pi(\boldsymbol{w})\boldsymbol{\varphi}. \tag{B.11}$$

Soient alors $\Delta = \pi(\boldsymbol{w}) - \boldsymbol{w} \in \mathcal{H}^m$ l'écart entre la solution continue et sa projection sur $\mathcal{H}_h^m$ et $\Delta_h = \pi(\boldsymbol{w}) - \mathbf{w} \in \mathcal{H}_h^m$ l'écart entre la solution discrète et la projection de la solution continue. Nous avons donc $\boldsymbol{w} - \mathbf{w} = \Delta_h - \Delta$.

En soustrayant le problème semi-discret au problème continu et en prenant l'écart Δ_h comme fonction test, nous obtenons :

$$\int_\Omega \partial_t(\Delta_h - \Delta) \cdot \Delta_h dx + B(\Delta_h - \Delta, \Delta_h) = 0. \tag{B.12}$$

Remarquons, par définition de la projection, que l'écart Δ est orthogonal à l'espace $\mathcal{H}_h^m$ qui contient Δ_h, $\partial_i \boldsymbol{A}_i \Delta_h$ et $\partial_t \Delta_h$, soit :

$$\int_\Omega \Delta \cdot \Delta_h dx = \int_\Omega \Delta \cdot (\partial_i \boldsymbol{A}_i \Delta_h) dx = \int_\Omega \Delta \cdot (\partial_t \Delta_h) dx = 0. \tag{B.13}$$

Ainsi :

$$\begin{aligned} 0 &= \partial_t \int_\Omega \Delta \cdot \Delta_h dx \\ &= \int_\Omega (\partial_t \Delta) \cdot \Delta_h dx + \int_\Omega \Delta \cdot (\partial_t \Delta_h) dx \\ &= \int_\Omega (\partial_t \Delta) \cdot \Delta_h dx, \end{aligned} \tag{B.14}$$

et l'équation (B.12) devient :

$$\int_\Omega (\partial_t \Delta_h) \cdot \Delta_h dx + B(\Delta_h, \Delta_h) = B(\Delta, \Delta_h). \tag{B.15}$$

Pour démontrer la convergence du problème semi-discret, nous souhaitons majorer les membres de cette équation. Afin d'appliquer l'inégalité de Cauchy-Schwarz au second membre, nous devons déterminer la partie anti-symétrique de B, soit la moitié de la différence symétrique suivante :

$$\begin{aligned} B(\boldsymbol{\varphi}, \boldsymbol{\psi}) - B(\boldsymbol{\psi}, \boldsymbol{\varphi}) = &\int_{\Omega \setminus \Sigma_h} (\partial_i \boldsymbol{A}_i \boldsymbol{\varphi}) \cdot \boldsymbol{\psi} dx - \int_{\Omega \setminus \Sigma_h} (\partial_i \boldsymbol{A}_i \boldsymbol{\psi}) \cdot \boldsymbol{\varphi} dx \\ &+ \int_{\Sigma_h} (\boldsymbol{A}_i n_i^- (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L)) \cdot \boldsymbol{\psi}_L ds \\ &+ \int_{\Sigma_h} (\boldsymbol{A}_i n_i^+ (\boldsymbol{\varphi}_R - \boldsymbol{\varphi}_L)) \cdot \boldsymbol{\psi}_R ds \\ &- \int_{\Sigma_h} (\boldsymbol{A}_i n_i^- (\boldsymbol{\psi}_R - \boldsymbol{\psi}_L)) \cdot \boldsymbol{\varphi}_L ds \\ &- \int_{\Sigma_h} (\boldsymbol{A}_i n_i^+ (\boldsymbol{\psi}_R - \boldsymbol{\psi}_L)) \cdot \boldsymbol{\varphi}_R ds, \end{aligned} \tag{B.16}$$

avec $\boldsymbol{\varphi}_R = \boldsymbol{\psi}_R = 0$ sur $\partial\Omega$. En appliquant la formule de Green (1.39) sur le premier terme de volume puis, en simplifiant les termes d'interface, nous

obtenons :

$$
\begin{aligned}
B(\boldsymbol{\varphi},\boldsymbol{\psi})-B(\boldsymbol{\psi},\boldsymbol{\varphi}) = &-2\int_{\Omega\setminus\Sigma_h} \boldsymbol{\varphi}\cdot(\partial_i \boldsymbol{A}_i\boldsymbol{\psi})dx \\
&+\int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}_L)\cdot\boldsymbol{\psi}_L ds - \int_{\Sigma_h} (\boldsymbol{A}_i n_i \boldsymbol{\varphi}_R)\cdot\boldsymbol{\psi}_R ds \\
&+\int_{\Sigma_h} (\boldsymbol{A}_i {n_i}^{-}(\boldsymbol{\varphi}_R-\boldsymbol{\varphi}_L))\cdot\boldsymbol{\psi}_L ds \\
&+\int_{\Sigma_h} (\boldsymbol{A}_i {n_i}^{+}(\boldsymbol{\varphi}_R-\boldsymbol{\varphi}_L))\cdot\boldsymbol{\psi}_R ds \\
&-\int_{\Sigma_h} (\boldsymbol{A}_i {n_i}^{-}(\boldsymbol{\psi}_R-\boldsymbol{\psi}_L))\cdot\boldsymbol{\varphi}_L ds \\
&-\int_{\Sigma_h} (\boldsymbol{A}_i {n_i}^{+}(\boldsymbol{\psi}_R-\boldsymbol{\psi}_L))\cdot\boldsymbol{\varphi}_R ds \\
= &-2\int_{\Omega\setminus\Sigma_h} \boldsymbol{\varphi}\cdot(\partial_i \boldsymbol{A}_i\boldsymbol{\psi})dx \\
&+\int_{\Sigma_h} (\boldsymbol{A}_i n_i(\boldsymbol{\varphi}_L+\boldsymbol{\varphi}_R))\cdot(\boldsymbol{\psi}_L-\boldsymbol{\psi}_R) ds.
\end{aligned}
\tag{B.17}
$$

En appliquant ensuite l'inégalité triangulaire suivie de l'inégalité de Cauchy-Schwarz, nous obtenons la majoration suivante :

$$
\begin{aligned}
|B(\Delta,\Delta_h)| &= |B_s(\Delta,\Delta_h)+B_a(\Delta,\Delta_h)| \\
&= \left|B_s(\Delta,\Delta_h)+\frac{1}{2}(B(\Delta,\Delta_h)-B(\Delta_h,\Delta))\right| \\
&\leq |B_s(\Delta,\Delta_h)| + \left|\frac{1}{2}(B(\Delta,\Delta_h)-B(\Delta_h,\Delta))\right| \\
&\leq \sqrt{B(\Delta,\Delta)}\sqrt{B(\Delta_h,\Delta_h)} \\
&\quad + \left|\frac{1}{2}\int_{\Sigma_h} (\boldsymbol{A}_i n_i(\Delta_L+\Delta_R))\cdot(\Delta_{hL}-\Delta_{hR})ds\right|
\end{aligned}
\tag{B.18}
$$

où B_s, respectivement B_a, est la partie symétrique, respectivement antisymétrique, de B. En appliquant le lemme 5, l'inégalité devient :

$$
|B(\Delta,\Delta_h)| \leq \sqrt{B(\Delta_h,\Delta_h)}\left(\sqrt{B(\Delta,\Delta)}+K_1\left\|\Delta\right\|_{\mathrm{L}^2(\Sigma_h)}\right), \tag{B.19}
$$

avec K_1 une constante réelle.

Nous disposons aussi des estimations de projection données par Raviart *et al.* [61] qui permettent de mesurer l'écart entre une solution $\boldsymbol{w}$ et son projeté

L^2 sur l'espace d'approximation spatiale $\mathcal{H}_h^m$:

$$\|\Delta\|_{\mathrm{L}^2(\Omega)} \leq Ch^{d+1} \|\boldsymbol{w}\|_{W^2_{d+1}(\mathcal{M})}, \tag{B.20a}$$

$$\|\Delta\|_{\mathrm{L}^2(\Sigma_h)} \leq Ch^{d+\frac{1}{2}} \|\boldsymbol{w}\|_{W^2_{d+1}(\mathcal{M})}, \tag{B.20b}$$

où C est une constante dépendant de Ω, d est l'ordre d'approximation spatiale, h est le paramètre de finesse du maillage $\mathcal{M}$ et la norme $\|.\|_{W^2_{d+1}(\mathcal{M})}$ est une norme de Sobolev par morceaux définie par :

$$\|\boldsymbol{w}\|_{W^2_{d+1}(\mathcal{M})} = \sum_{i=0}^{N-1} \|\boldsymbol{w}\|_{W^2_{d+1}(L_i)} = \sum_{i=0}^{N-1} \left(\sum_{|\alpha|\leq d+1} \left\|\boldsymbol{w}^{(\alpha)}\right\|^2_{\mathrm{L}^2(L_i)} \right)^{\frac{1}{2}}. \tag{B.21}$$

Cette estimation est valable à condition que les mailles ne s'étirent pas lorsque h tend vers 0. C'est le cas par exemple, si pour le passage à la limite sur h, on considère des raffinements uniformes d'un maillage fixé initialement.

Nous obtenons ainsi la majoration :

$$\frac{1}{2}\partial_t \|\Delta_h\|^2_{\mathrm{L}^2(\Omega)} + B(\Delta_h, \Delta_h) \leq K_2 \sqrt{B(\Delta_h, \Delta_h)} h^{d+\frac{1}{2}} \|\boldsymbol{w}\|_{W^2_{d+1}(\mathcal{M})}, \tag{B.22}$$

avec K_2 une constante réelle. Puis, en appliquant l'inégalité :

$$\forall a, b \in \mathbb{R}, ab \leq \frac{1}{2}(a^2 + b^2), \tag{B.23}$$

avec $a = \sqrt{B(\Delta_h, \Delta_h)}$ et $b = K_2 h^{d+\frac{1}{2}} \|\boldsymbol{w}\|_{W^2_{d+1}(\mathcal{M})}$, nous avons :

$$\partial_t \|\Delta_h\|^2_{\mathrm{L}^2(\Omega)} + B(\Delta_h, \Delta_h) \leq K_2^2 h^{2d+1} \|\boldsymbol{w}\|^2_{W^2_{d+1}(\mathcal{M})}, \tag{B.24}$$

et finalement en intégrant par rapport au temps :

$$\sqrt{\|\Delta_h\|^2_{\mathrm{L}^2(\Omega)} + \int_0^T B(\Delta_h, \Delta_h) dt} \leq K_2 h^{d+\frac{1}{2}} \sqrt{\int_0^T \|\boldsymbol{w}\|^2_{W^2_{d+1}(\mathcal{M})} dt}. \tag{B.25}$$

Puisque $\boldsymbol{w} - \mathbf{w} = \Delta_h - \Delta$,nous pouvons conclure par :

$$\|\boldsymbol{w} - \mathbf{w}\|_{\mathrm{L}^2(\Omega)} \leq K_2 h^{d+\frac{1}{2}} \sqrt{\int_0^T \|\boldsymbol{w}\|^2_{W^2_{d+1}(\mathcal{M})} dt}. \tag{B.26}$$

□

Remarque 38. *L'inégalité* (B.25) *est plus précise que cette dernière inégalité puisqu'elle comprend un contrôle des discontinuités sur les interfaces entre les mailles. Ce contrôle montre qu'en un certain sens, le saut de* $\boldsymbol{A}_i n_i \boldsymbol{w}$ *sur l'ensemble des discontinuités* Σ_h *tend vers* 0 *en norme* $\mathrm{L}_2(\Sigma_h)$ *lorsque* h *tend vers* 0*. Comme la longueur de l'ensemble* Σ_h *tend vers l'infini, ce contrôle est assez fort. Dans le cas de l'utilisation d'un flux centré, ce terme de contrôle n'est pas présent. Ainsi, les oscillations de Gibbs sont en général plus importantes avec un tel flux.*

Annexe C

Résultats sur corps humain complet

Dans cette annexe nous présentons les résultats de la simulation présentée dans la section 7.3. Cette simulation met en scène un modèle de corps humain complet (comprenant 12 organes, le squelette et la peau) avec une antenne de technologie LTCC placée entre le bras gauche et le corps. Le temps physique simulé est de 10 ns. Nous présentons les résultats en plan de coupe à chaque nanoseconde écoulée.

FIGURE C.1 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 1$ ns.

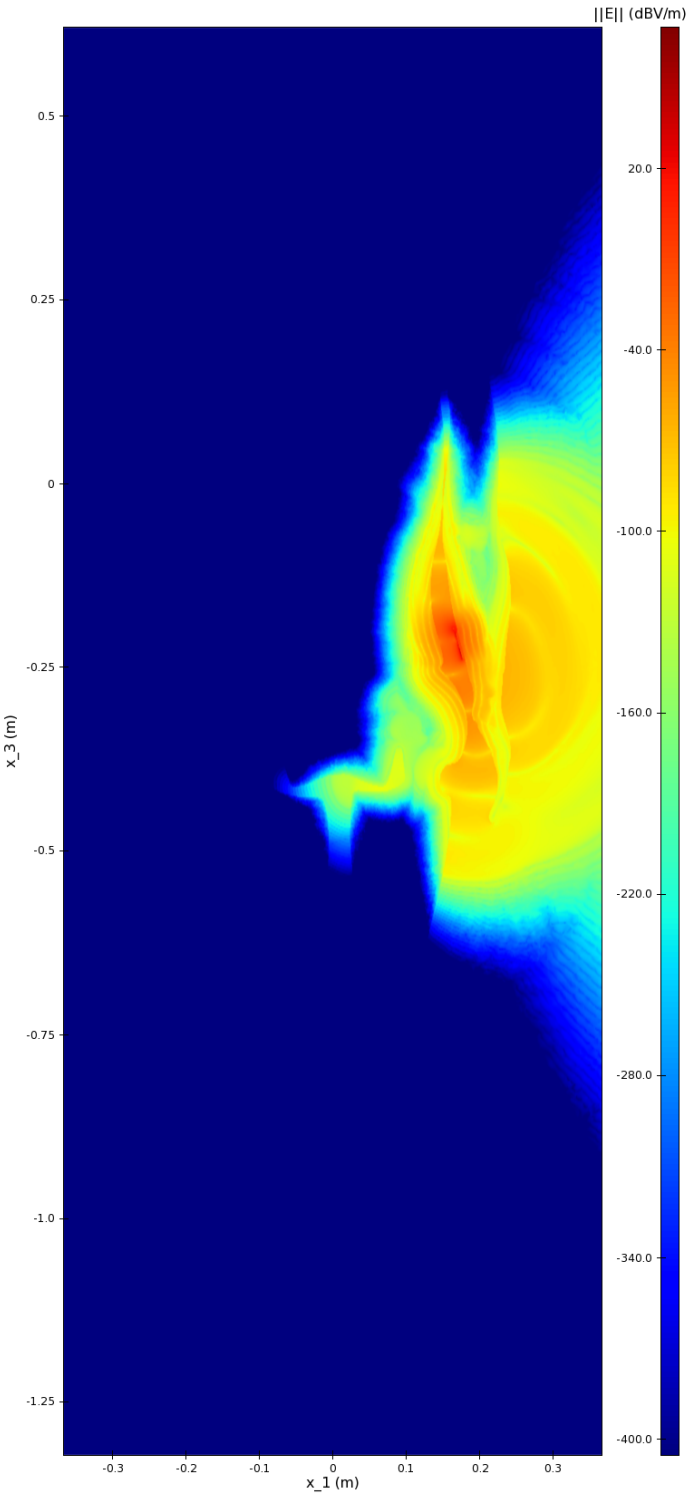


FIGURE C.2 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 2$ ns.

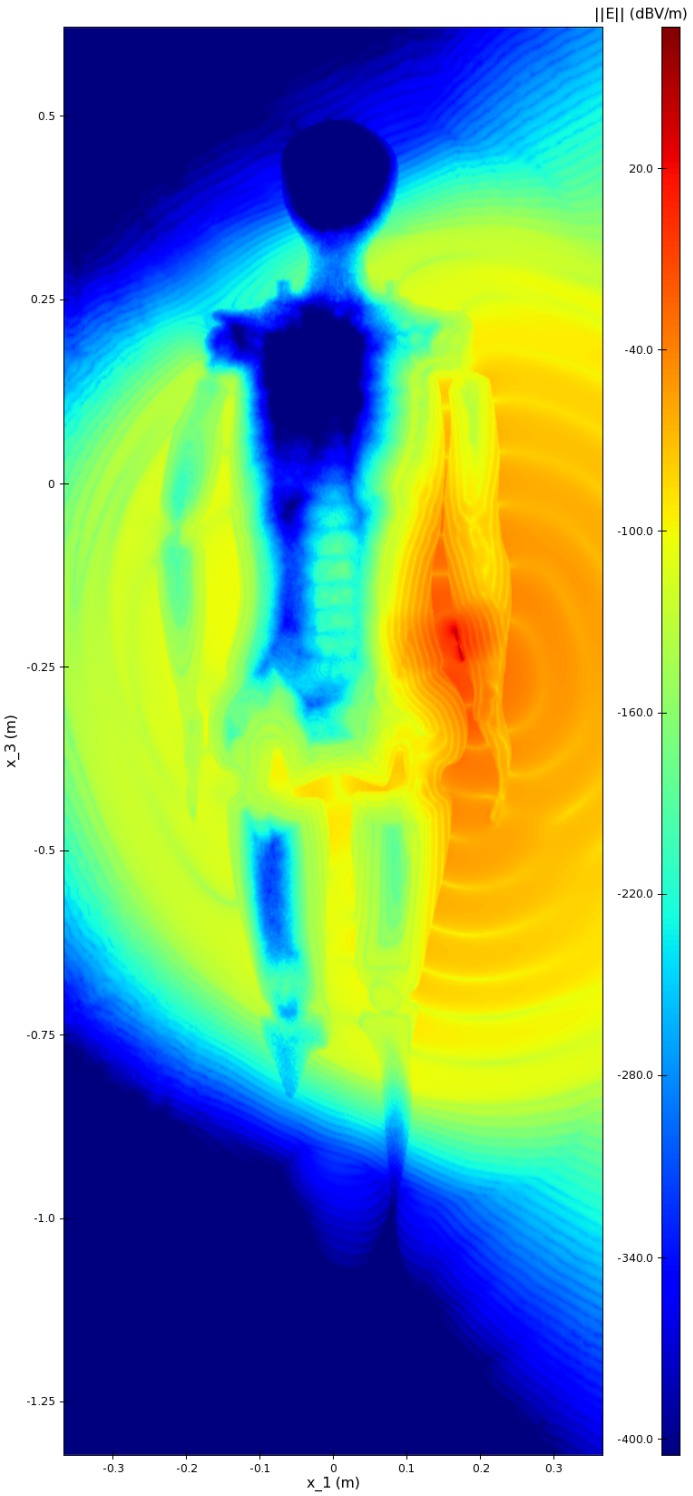


FIGURE C.3 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 3$ ns.

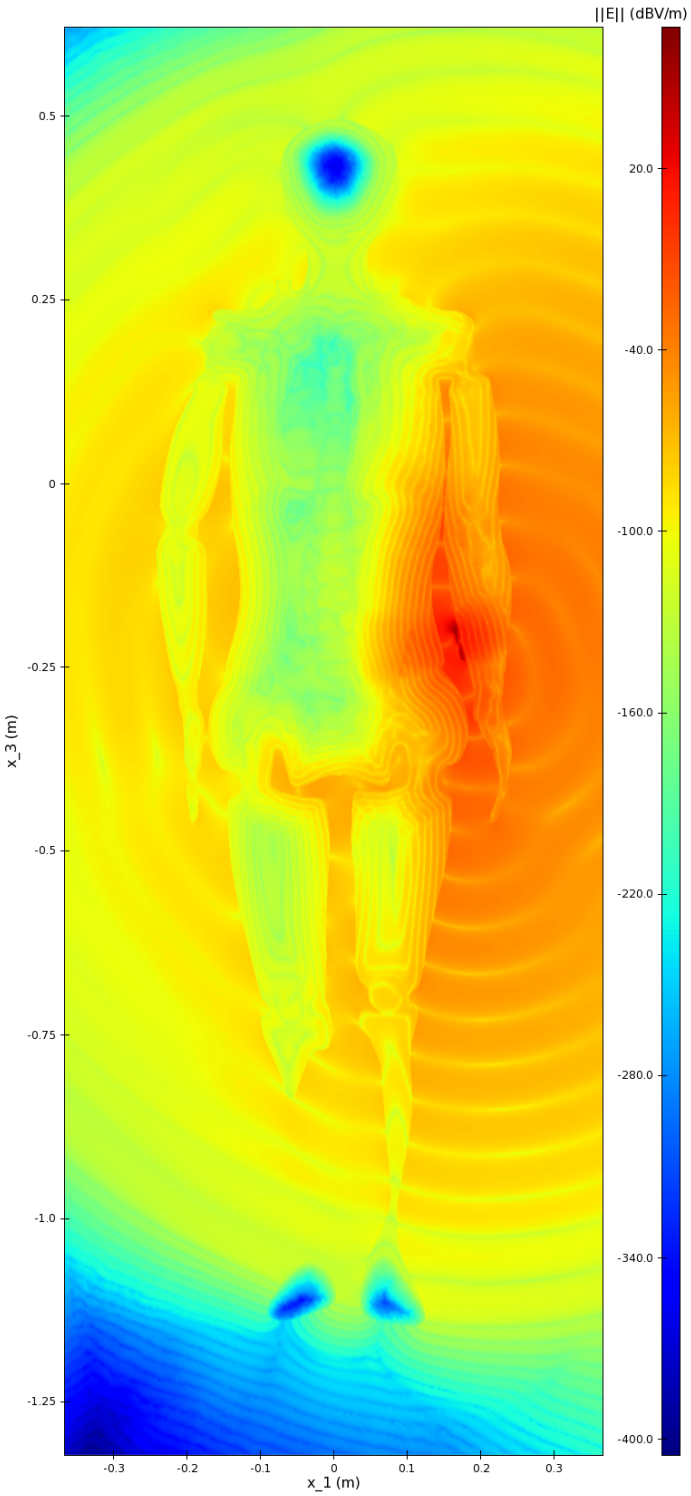


FIGURE C.4 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 4$ ns.

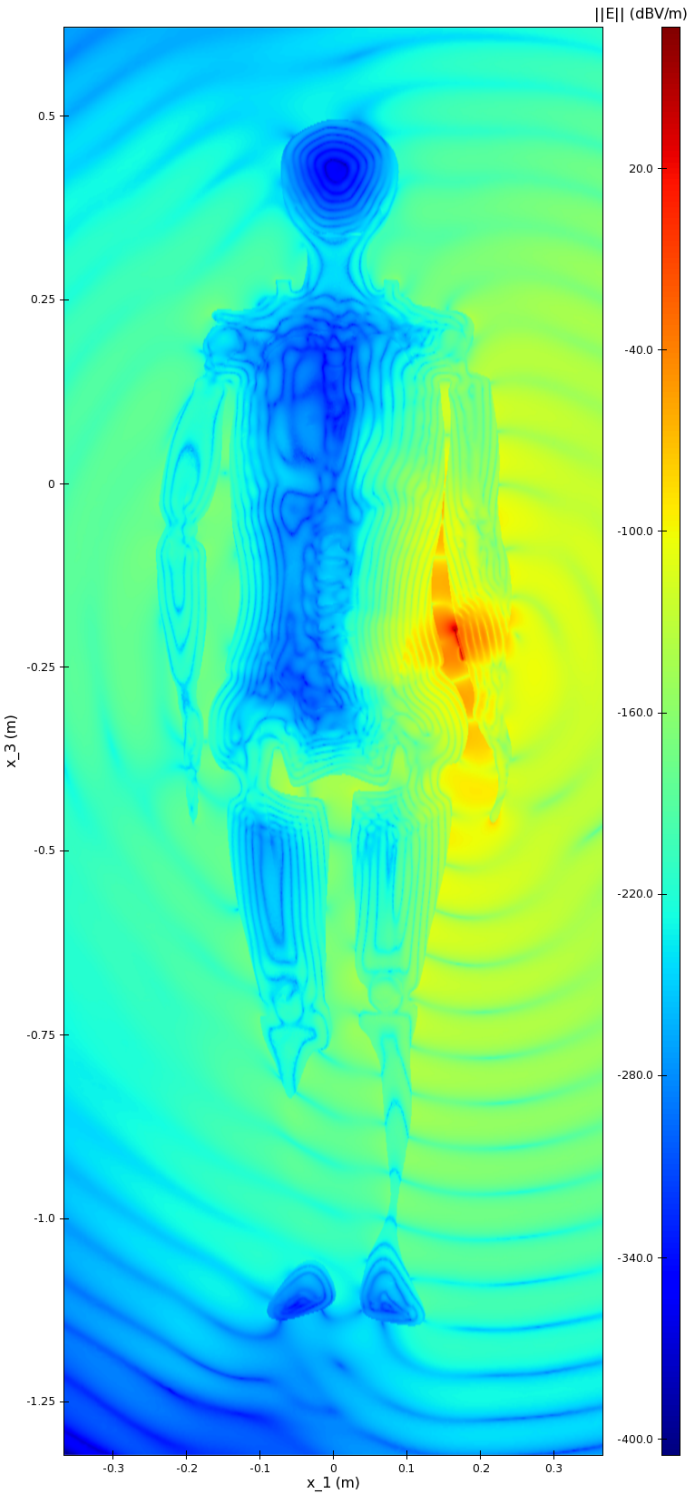


FIGURE C.5 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 5$ ns.

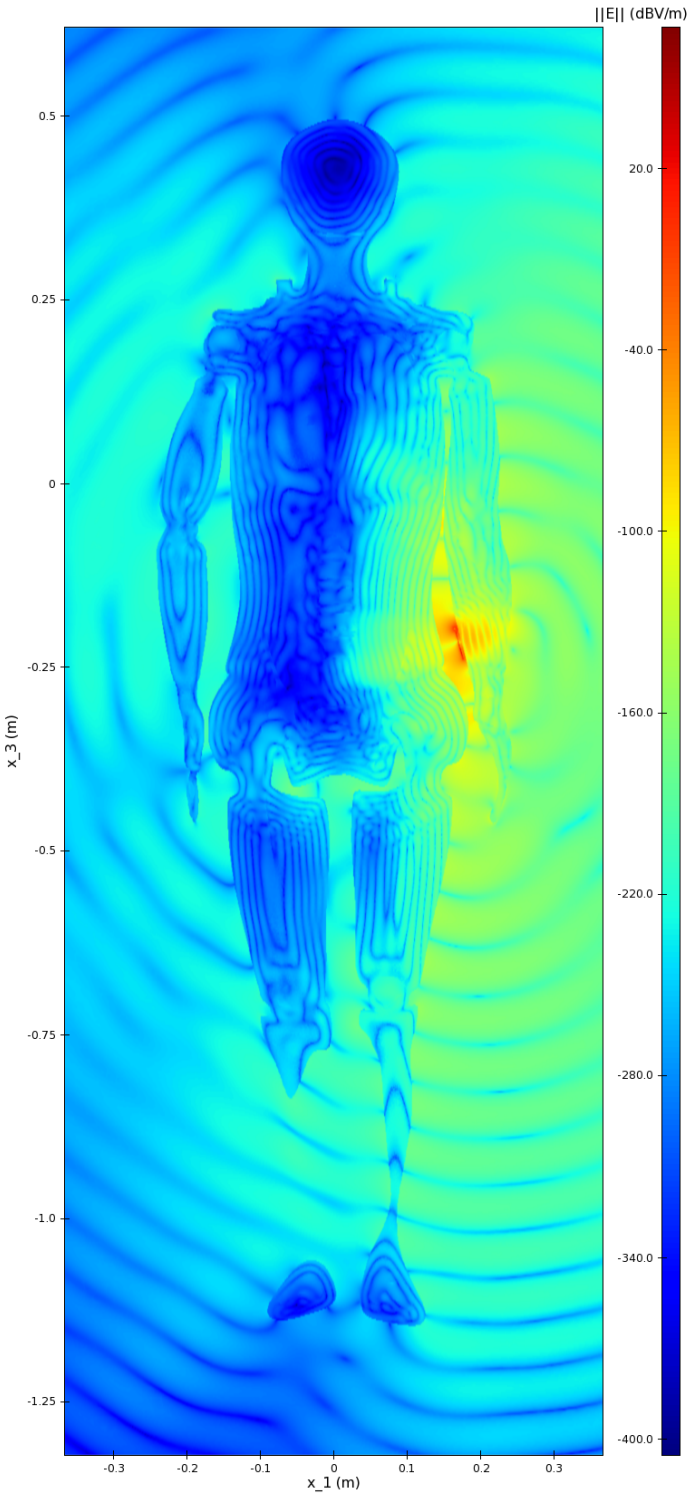


FIGURE C.6 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 6$ ns.

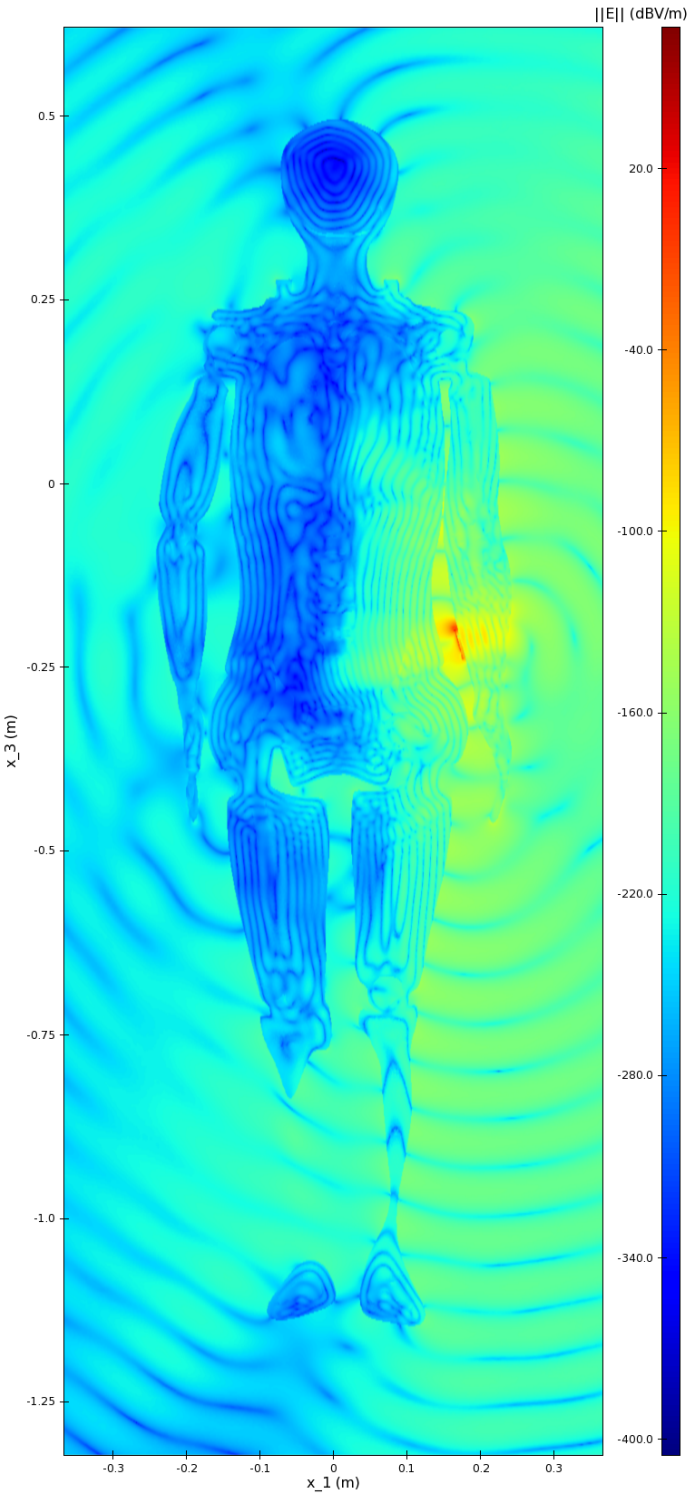


FIGURE C.7 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 7$ ns.

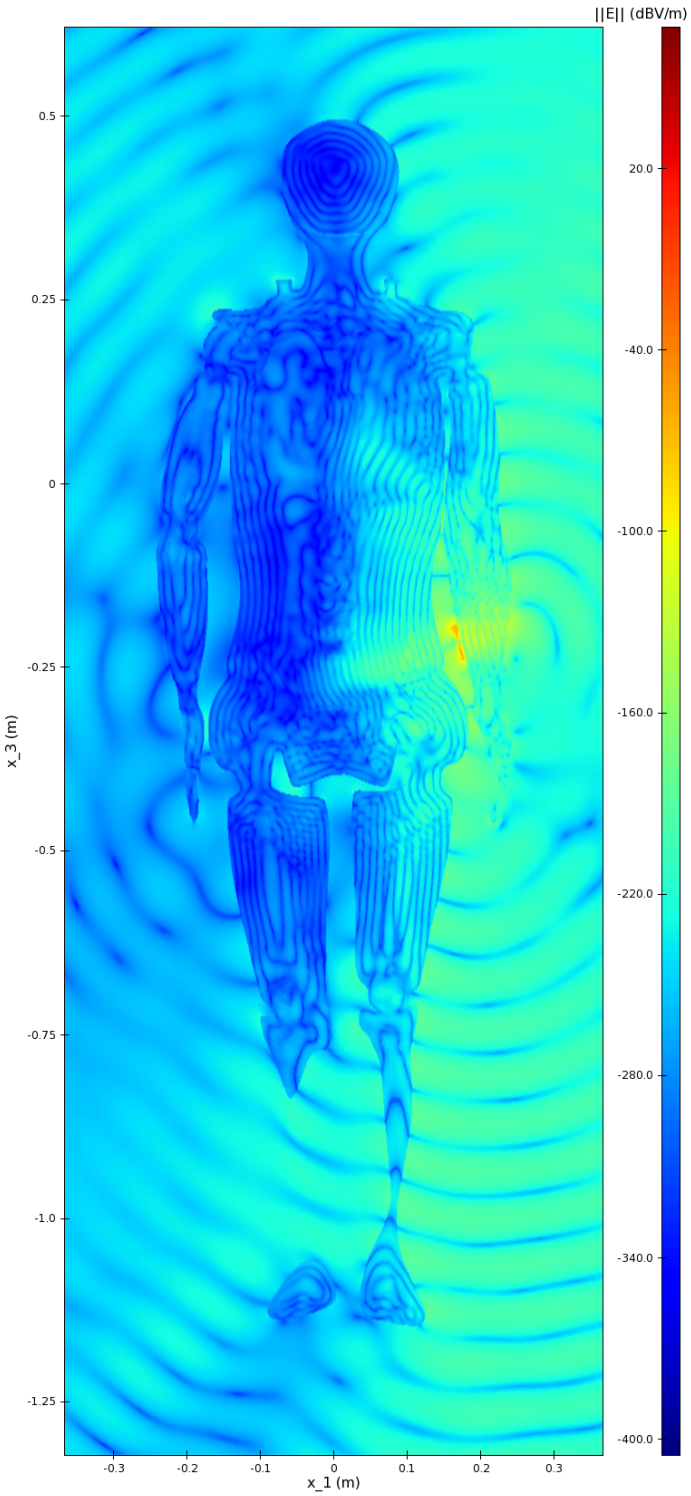


FIGURE C.8 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 8$ ns.

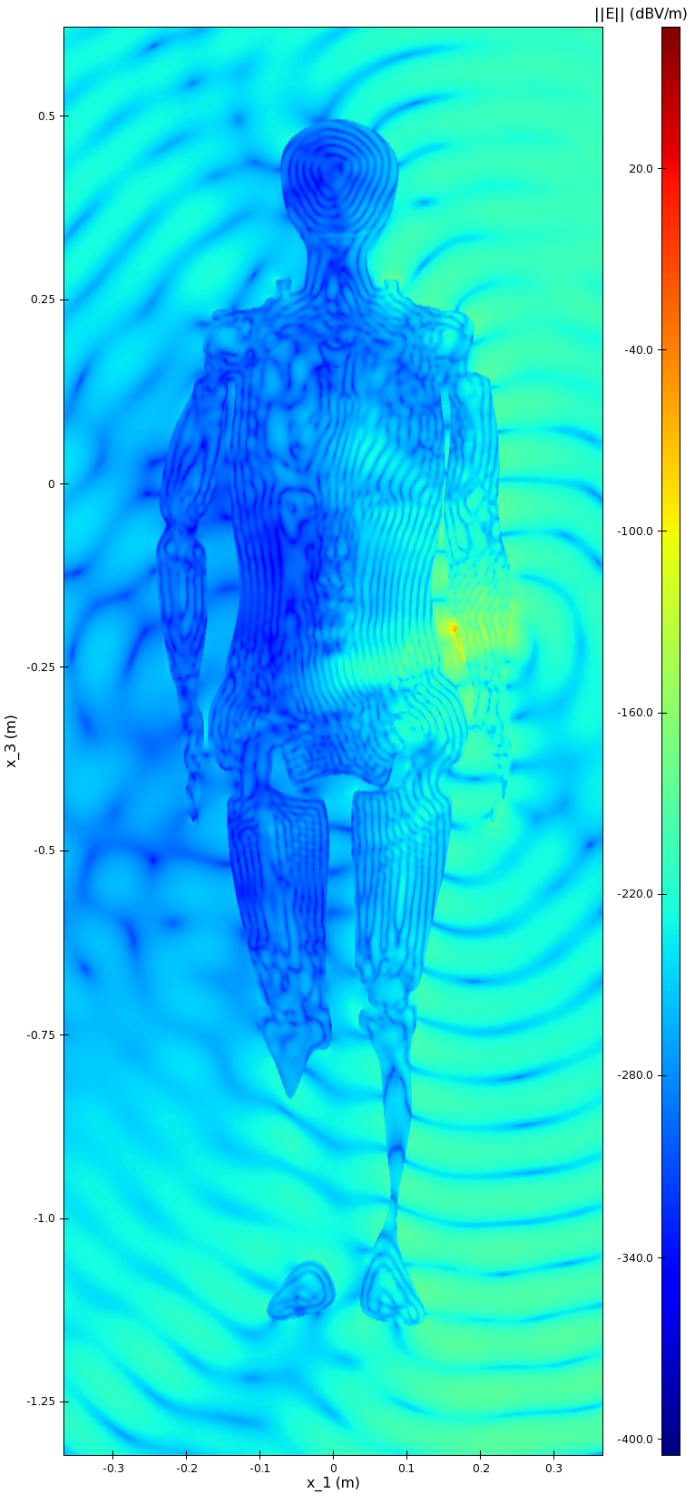


FIGURE C.9 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 9$ ns.

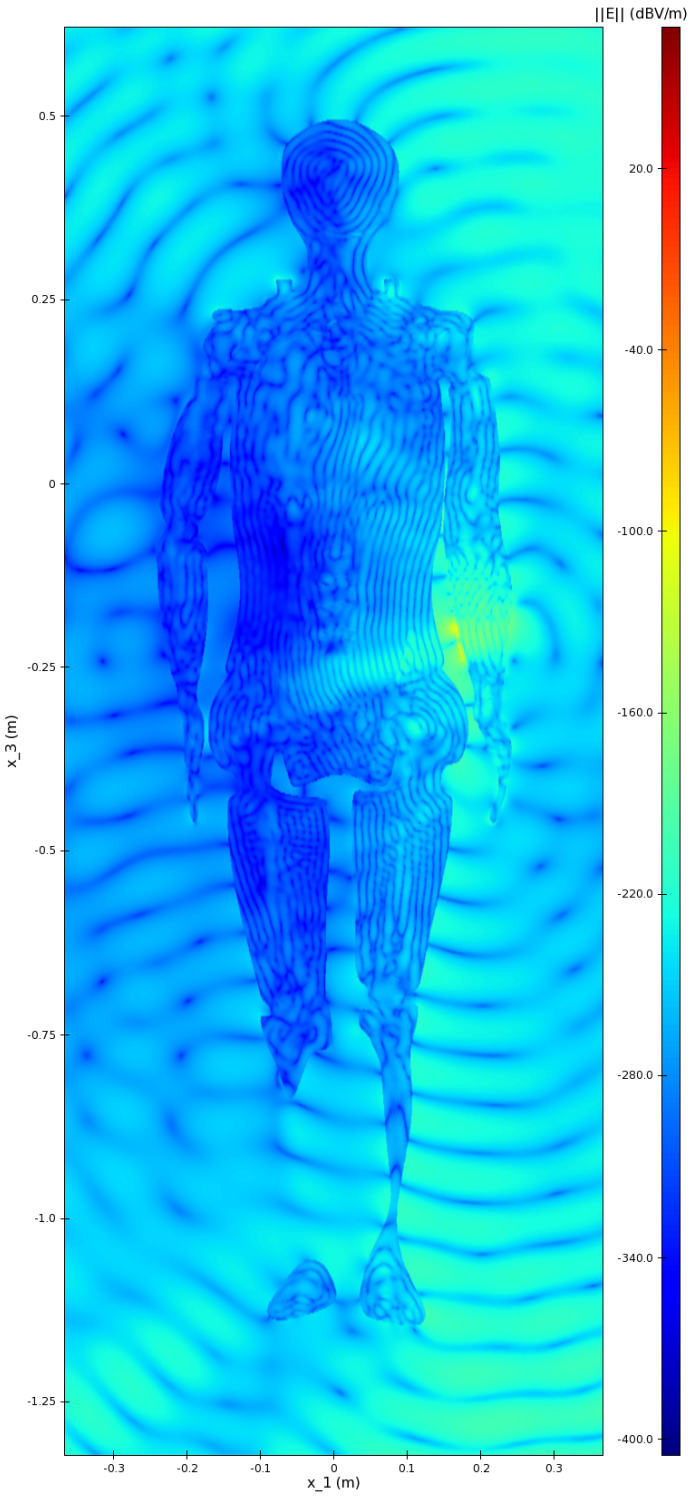
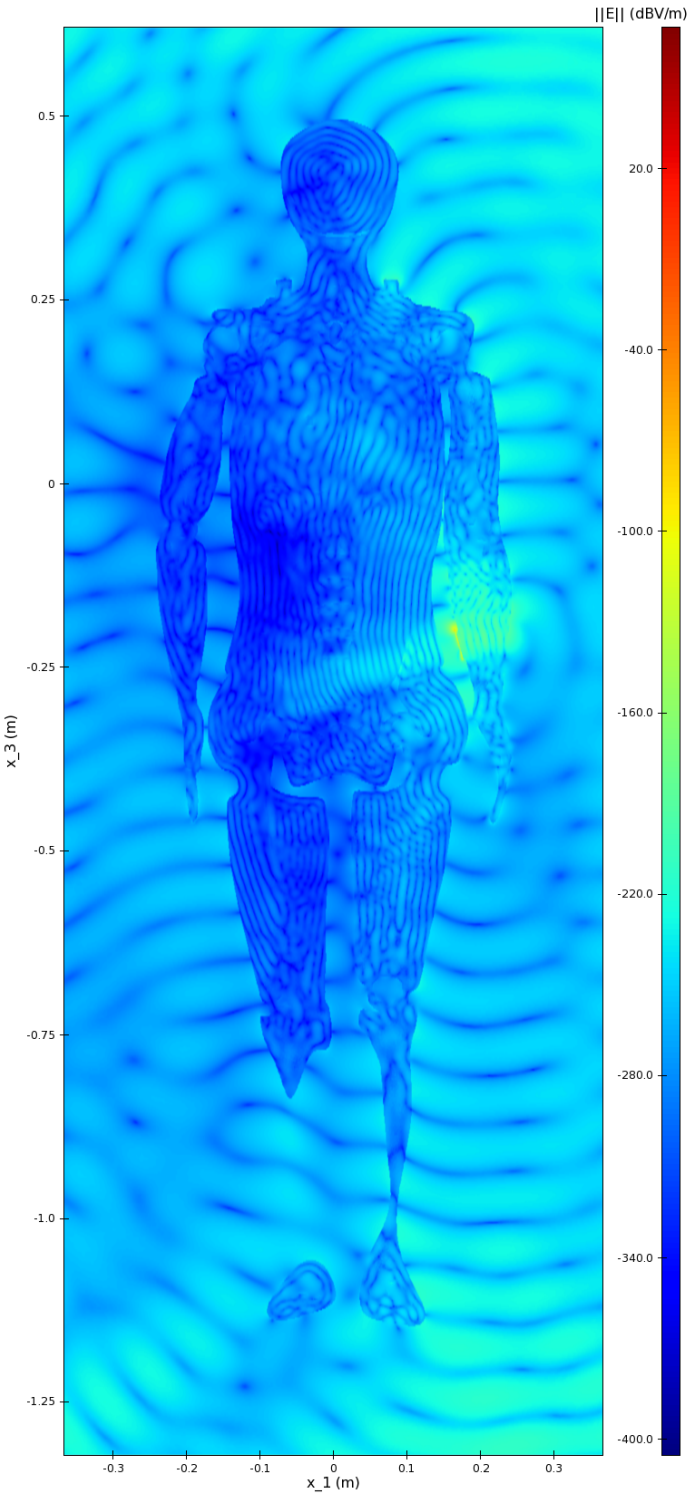


FIGURE C.10 – Résultats en plan de coupe (x_1Ox_3) en $x_2 = -0.1255$ m (antenne) au temps $t = 10$ ns.



Bibliographie

[1] AMD. *AMD APP SDK. OpenCL Optimization Guide.*, 2015.

[2] Cédric Augonnet, Olivier Aumage, Nathalie Furmento, Raymond Namyst, and Samuel Thibault. Starpu-mpi : Task programming over clusters of machines enhanced with accelerators. In *European MPI Users' Group Meeting*, pages 298–299. Springer, 2012.

[3] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. Starpu : a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation : Practice and Experience*, 23(2) :187–198, 2011.

[4] H. Barucq and B. Hanouzet. Etude asymptotique du systeme de Maxwell avec la condition aux limites absorbante de Silver-Müller II. *Comptes rendus de l'Académie des sciences. Série I, Mathématique*, 316(10) :1019–1024, 1993.

[5] A. Bendali and L. Halpern. Conditions aux limites absorbantes pour le système de maxwell dans le vide en dimension 3. 01 1988.

[6] J.-P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2) :185 – 200, 1994.

[7] F. Bourdel, P.-A. Mazet, and P. Helluy. Resolution of the non-stationary or harmonic maxwell equations by a discontinuous finite element method : Application to an e.m.i. (electromagnetic impulse) case. In *Proceedings of the 10th International Conference on Computing Methods in Applied Sciences and Engineering on Computing Methods in Applied Sciences and Engineering*, pages 405–422, Commack, NY, USA, 1991. Nova Science Publishers, Inc.

[8] H. Brézis. *Analyse fonctionelle :*. Collection Mathématiques appliquées pour la maîtrise. Masson, 1983.

[9] E. Burman, A. Ern, and M. Fernández. Explicit runge–kutta schemes and finite elements with symmetric stabilization for first-order linear

pde systems. *SIAM Journal on Numerical Analysis*, 48(6) :2019–2042, 2010.

[10] J.-P. Bérenger. Plaques minces aux différences finies. *6éme Colloque International et Exposition sur la Compatibilité Electromagnétique, CEM'92*, pages 298–303, Jun 1992.

[11] T. Cabel, J. Charles, and S. Lanteri. Multi-GPU acceleration of a DGTD method for modeling human exposure to electromagnetic waves. Research Report RR-7592, INRIA, April 2011.

[12] N. Castel, G. Cohen, and M. Duruflé. Application of Discontinuous Galerkin spectral method on hexahedral elements for aeroacoustic. *Journal of Computational Acoustics*, 17(2) :175–196, 2009.

[13] Young Wook Cheong, Yong Min Lee, Keuk Hwan Ra, Joon Gil Kang, and Chull Chae Shin. Wavelet-galerkin scheme of time-dependent inhomogeneous electromagnetic problems. *IEEE Microwave and Guided Wave Letters*, 9(8) :297–299, Aug 1999.

[14] Lyndon Clarke, Ian Glendinning, and Rolf Hempel. The mpi message passing interface standard. In Karsten M. Decker and René M. Rehmann, editors, *Programming Environments for Massively Parallel Distributed Systems*, pages 213–218, Basel, 1994. Birkhäuser Basel.

[15] G. Cohen, X Ferrieres, and S. Pernet. A spatial high-order hexahedral discontinuous galerkin method to solve maxwell's equations in time domain. *Journal of Computational Physics*, 217(2) :340–363, 2006.

[16] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1) :32–74, Dec 1928.

[17] R. Courant and K. O. Friedrichs. *Supersonic Flow and Shock Waves*, volume 21. Springer-Verlag, 1977.

[18] A. Crestetto and P. Helluy. Resolution of the Vlasov-Maxwell system by PIC Discontinuous Galerkin method on GPU with OpenCL. In *CEMRACS'11*, volume 38, pages 257–274, France, 2011. EDP Sciences.

[19] F. De Vuyst and F. Salvarani. GPU-accelerated numerical simulations of the Knudsen gas on time-dependent domains. working paper or preprint, September 2012.

[20] A. Ern and J.-L. Guermond. Discontinuous galerkin methods for friedrichs' systems. i. general theory. *SIAM Journal on Numerical Analysis*, 44(2) :753–778, 2006.

[21] J. Fang and Z. Wu. Generalized perfectly matched layer for the absorption of propagating and evanescent waves in lossless and lossy media.

IEEE Transactions on Microwave Theory and Techniques, 44(12) :2216–2222, Dec 1996.

[22] Loula Fezoui, Stéphane Lanteri, Stéphanie Lohrengel, and Serge Piperno. Convergence and stability of a discontinuous galerkin time-domain method for the 3d heterogeneous maxwell equations on unstructured meshes. *ESAIM : Mathematical Modelling and Numerical Analysis*, 39(6) :1149–1176, 2005.

[23] Bruno Fornet, Vincent Mouysset, and Ángel Rodríguez-Arós. Mathematical study of a hyperbolic regularization to ensure Gauss law conservation in Maxwell–Vlasov applications. *Mathematical Models and Methods in Applied Sciences*, 22(04) :1150020, 2012.

[24] K. O. Friedrichs. Symmetric hyperbolic linear differential equations. *Communications on Pure and Applied Mathematics*, 7(2) :345–392, 1954.

[25] C Gabriel, A Peyman, and E H Grant. Electrical conductivity of tissue at frequencies below 1 mhz. *Physics in Medicine and Biology*, 54(16) :4863, 2009.

[26] Camelia Gabriel. Compilation of the dielectric properties of body tissues at rf and microwave frequencies. page 272, 02 1996.

[27] S Gabriel, R W Lau, and C Gabriel. The dielectric properties of biological tissues : Ii. measurements in the frequency range 10 hz to 20 ghz. *Physics in Medicine and Biology*, 41(11) :2251, 1996.

[28] S. D. Gedney. An anisotropic perfectly matched layer-absorbing medium for the truncation of fdtd lattices. *IEEE Transactions on Antennas and Propagation*, 44(12) :1630–1639, Dec 1996.

[29] The Khronos Group. The open standard for parallel programming of heterogeneous systems.

[30] C. Guiffaut. *Contribution a la méthode FDTD pour l'étude d'antennes et de la diffraction d'objets enfouis*. 2000.

[31] Christophe Guiffaut, Alain Reineix, and B. Pecqueux. New Oblique Thin Wire Formalism in the FDTD Method. In *IEEE AP-S International Symposium on Antennas and Propagation and 2010*, page inconnu, Toronto, Canada, November 2010.

[32] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*, 14(3) :1052–1061, Jul 1999.

[33] A. Guéna, S. Lamesch, V. Forte, D. Halley, P. Muris, M. Viguier, N. Muot, T. Strub, B. Weber, and C. Girard. Electromagnetic analysis : Radiated emission of iot applications close to an anthropomor-

phic mannequin. In *2017 International Symposium on Electromagnetic Compatibility - EMC EUROPE*, pages 1–5, Sept 2017.

[34] M. Han, R. W. Dutton, and S. Fan. Model dispersive media in finite-difference time-domain method with complex-conjugate pole-residue pairs. *IEEE Microwave and Wireless Components Letters*, 16(3) :119–121, 2006.

[35] P. Helluy and J. Jung. Two-fluid compressible simulations on GPU cluster. *ESAIM : Proceedings and Surveys*, pages 349 – 358, November 2014.

[36] Philippe Helluy, Thomas Strub, Michel Massaro, and Malcolm Roberts. Asynchronous opencl/mpi numerical simulations of conservation laws. In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 547–565. Springer, 2016.

[37] J. S. Hesthaven and T. Warburton. High–order nodal discontinuous galerkin methods for the maxwell eigenvalue problem. *Philosophical Transactions of the Royal Society of London A : Mathematical, Physical and Engineering Sciences*, 362(1816) :493–524, 2004.

[38] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods : Algorithms, Analysis, and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[39] R. Holland and L. Simpson. Finite-difference analysis of emp coupling to thin struts and wires. *IEEE Transactions on Electromagnetic Compatibility*, EMC-23(2) :88–97, May 1981.

[40] Intel. *Intel SDK for OpenCL Applications XE 2013 Beta. Optimization Guide. Document number : 326542-001US*, 2012.

[41] J. JAFFRE, C. JOHNSON, and A. SZEPESSY. Convergence of the discontinuous galerkin finite element method for hyperbolic conservation laws. *Mathematical Models and Methods in Applied Sciences*, 05(03) :367–386, 1995.

[42] C. Johnson and J. Pitkäranta. An analysis of the discontinuous galerkin method for a scalar hyperbolic equation. *Math. Comput.*, 46(173) :1–26, January 1986.

[43] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1) :359–392, December 1998.

[44] T. Kato. *Perturbation Theory for Linear Operators*. Classics in Mathematics. Springer Berlin Heidelberg, 1995.

[45] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven. Nodal discontinuous galerkin methods on graphics processors. *J. Comput. Phys.*, 228(21) :7863–7882, November 2009.

[46] D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *International Journal for Numerical Methods in Engineering*, 53(1) :105–122.

[47] Ethan J Kubatko, Clint Dawson, and Joannes J Westerink. Time step restrictions for runge–kutta discontinuous galerkin methods on triangular grids. *Journal of Computational Physics*, 227(23) :9697–9710, 2008.

[48] M. Lassas, J. Liukkonen, and E. Somersalo. Complex riemannian metric and absorbing boundary conditions. *Journal de Mathématiques Pures et Appliquées*, 80(7) :739 – 768, 2001.

[49] S. Laurens. *High accuracy approximation for diffraction problems.* Theses, Université Paul Sabatier - Toulouse III, March 2010.

[50] Sophie Laurens. A general family of Perfectly Matched Layers for non necessarily convex domains. Research report, June 2010.

[51] P. D. Lax and R. S. Phillips. Local boundary conditions for dissipative symmetric linear differential operators. *Communications on Pure and Applied Mathematics*, 13 :427–455, 1960.

[52] P. Lesaint and P.-A. Raviart. On a finite element method for solving the neutron transport equation. In Carl de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 89–123. Academic Press, 1974.

[53] S. Lohrengel, S. Piperno, L. F. Fezoui, and S. Lanteri. Convergence and Stability of a Discontinuous Galerkin Time-Domain method for the 3D heterogeneous Maxwell equations on unstructured meshes. *Modelisation Mathématique et Analyse Numérique*, 39(6) :1149–1176, 2005.

[54] Tiao Lu, Pingwen Zhang, and Wei Cai. Discontinuous galerkin methods for dispersive and lossy maxwell's equations and pml boundary conditions. *Journal of Computational Physics*, 200(2) :549 – 580, 2004.

[55] T. Lähivaara and T. Huttunen. A non-uniform basis order for the discontinuous galerkin method of the 3d dissipative wave equation with perfectly matched layer. *Journal of Computational Physics*, 229(13) :5144 – 5160, 2010.

[56] P.-A. Mazet, S. Paintandre, and A. Rahmouni. Interprétation dispersive du milieu pml de bérenger. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 327(1) :59 – 64, 1998.

[57] P.-A. Mazet, L. Ségui, and B. Dah. Sur l'existence et l'unicité des solutions pour le système de maxwell harmonique en présence de couches de bérenger. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 333(6) :599 – 604, 2001.

[58] N. Muot, B. Weber, T. Strub, A. Guéna, and S. Lamesch. Implémentation massivement parallèle d'une méthode galerkin discontinue appliquée à l'électromagnétisme en domaine temporel – application aux interactions onde / corps humain. In *EMC EUROPE - 2018 International Symposium on Electromagnetic Compatibility*, 2018.

[59] O. Ramadan. Unified frequency-dependent split-step fdtd formulations based on the complex-conjugate pole-residue pairs model. *IEEE Microwave and Wireless Components Letters*, 24(8) :509–511, 2014.

[60] J. Rauch. Symmetric positive systems with boundary characteristic of constant multiplicity. *Transactions of the American Mathematical Society*, 291(1) :167–187, 1985.

[61] P.-A. Raviart and J.-M. Thomas. Primal hybrid finite element methods for 2nd order elliptic equations. 31, 05 1977.

[62] Kai Schneider, Dmitry Kolomenskiy, and Erwan Deriaz. Is the cfl condition sufficient ? some remarks. In *The Courant–Friedrichs–Lewy (CFL) Condition*, pages 139–146. Springer, 2013.

[63] L. Schwartz. *Méthodes mathématiques pour les sciences physiques*. Hermann, 1956.

[64] T Schwartzkopff, Claus-Dieter Munz, and Eleuterio F Toro. Ader : A high-order approach for linear hyperbolic systems in 2d. *Journal of Scientific Computing*, 17(1-4) :231–240, 2002.

[65] Thomas Schwartzkopff, Michael Dumbser, and Claus-Dieter Munz. Fast high order ader schemes for linear hyperbolic equations. *Journal of Computational Physics*, 197(2) :532–539, 2004.

[66] T. B. A. Senior. Impedance boundary conditions for imperfectly conducting surfaces. *Applied Scientific Research, Section B*, 8(1) :418, Dec 1960.

[67] T. G. Spence and D. H. Werner. A novel miniature broadband/multiband antenna based on an end-loaded planar open-sleeve dipole. *IEEE Transactions on Antennas and Propagation*, 54(12) :3614–3620, Dec 2006.

[68] Erich Strohmaier. Top500 supercomputer. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[69] T. Strub. *Résolution des équations de Maxwell tridimensionnelles instationnaires sur architecture massivement multicœur*. Theses, Université de Strasbourg, March 2015.

[70] D. M. Sullivan. An unsplit step 3-d pml for use with the fdtd method. *IEEE Microwave and Guided Wave Letters*, 7(7) :184–186, Jul 1997.

[71] A. Taflove and S.C. Hagness. *Computational Electrodynamics : The Finite-difference Time-domain Method*. Artech House Antennas and Prop. Artech House, 2005.

[72] Ubaid Ullah, Nor Mahyuddin, Zainal Ahmad, Mohd Zaid Abdullah, and Arjuna Marzuki. Antenna in ltcc technologies : A review and the current state of the art. 57 :241–260, 04 2015.

[73] B. Weber, C. Girard, B. Cirou, and V. Cameo ponz. Shape project axessim - cines partnership : Hpc for connected objects. Technical report, 2018.

[74] Bruno Weber, Philippe Helluy, and Thomas Strub. Optimization of a Discontinuous Galerkin algorithm in OpenCL applied to electromagnetism simulation. In *NUMELEC 2017 - 9th European Conference on Numerical Methods in Electromagnetics*, pages 1–2, Paris, France, November 2017.

[75] J. P. Wrenger. Numerical reflection from fdtd-pmls : a comparison of the split pml with the unsplit and cfs pmls. *IEEE Transactions on Antennas and Propagation*, 50(3) :258–265, Mar 2002.

Bruno WEBER

Optimisation de code Galerkin Discontinu sur ordinateur hybride. Application à la simulation numérique en électromagnétisme.

Résumé :
Nous présentons dans cette thèse les évolutions apportées au solveur Galerkin Discontinu `teta-clac`, issu de la collaboration IRMA-AxesSim, au cours du projet HOROCH (2015-2018). Ce solveur permet de résoudre les équations de Maxwell en 3D, en parallèle sur un grand nombre d'accélérateurs OpenCL.
L'objectif du projet HOROCH était d'effectuer des simulations de grande envergure sur un modèle numérique complet de corps humain. Ce modèle comporte 24 millions de mailles hexaédriques pour des calculs dans la bande de fréquences des objets connectés allant de 1 à 3 GHz (Bluetooth). Les applications sont nombreuses : téléphonie et accessoires, sport (maillots connectés), médecine (sondes : gélules, patchs), *etc.*
Les évolutions ainsi apportées comprennent, entre autres : l'optimisation des kernels OpenCL à destination des CPU dans le but d'utiliser au mieux les architectures hybrides ; l'expérimentation du runtime StarPU ; le design d'un schéma d'intégration à pas de temps local ; et bon nombre d'optimisations permettant au solveur de traiter des simulations de plusieurs millions de mailles.

Mots-clés :
solveur, Maxwell, électromagnétisme, système hyperbolique, Galerkin Discontinu, GD, GDTD, maillage, hexaèdres, GPU, CPU, OpenCL, MPI, StarPU, pas de temps local, ordre spatial adaptatif, modèle de corps humain complet, objets connectés, Bluetooth.

Abstract:
In this thesis, we present the evolutions made to the Discontinuous Galerkin solver `teta-clac` — resulting from the IRMA-AxesSim collaboration — during the HOROCH project (2015-2018). This solver allows to solve the Maxwell equations in 3D and in parallel on a large amount of OpenCL accelerators.
The goal of the HOROCH project was to perform large-scale simulations on a complete digital human body model. This model is composed of 24 million hexahedral cells in order to perform calculations in the frequency band of connected objects going from 1 to 3 GHz (Bluetooth). The applications are numerous: telephony and accessories, sport (connected shirts), medicine (probes: capsules, patches), *etc.*
The changes thus made include, among others: optimization of OpenCL kernels for CPUs in order to make the best use of hybrid architectures; StarPU runtime experimentation; the design of an integration scheme using local time steps; and many optimizations allowing the solver to process simulations of several millions of cells.

Keywords:
solver, Maxwell, electromagnetism, hyperbolic system, Discontinuous Galerkin, DG, DGTD, mesh, hexahedrons, GPU, CPU, OpenCL, MPI, StarPU, local time step, adaptive spatial order, complete human body model, connected objects, Bluetooth.