Sur la propagation des ondes laser avec couplage à l'hydrodynamique pour l'interaction laser plasma

Sylvain Desroziers

Université Pierre et Marie Curie, Paris VI

École doctorale des sciences de Paris Centre
UFR 921

Sur la propagation des ondes laser avec couplage à l'hydrodynamique pour l'interaction laser-plasma

THÈSE

présentée et soutenue publiquement le 27/04/2006

pour l'obtention du

Doctorat de l'université Pierre et Marie Curie – Paris 6
(spécialité Mathématiques Appliquées)

par

Sylvain DESROZIERS

Composition du jury

Rapporteurs :	L. GIRAUD B. NKONGA
Examinateurs :	G. MEURANT O. PIRONNEAU
Invité :	S. HÜLLER
Directeurs de Thèse :	F. NATAF R. SENTIS

Laboratoire d'accueil : CEA - DAM / DCSA / SSEL

Mis en page avec la classe thloria.

Remerciements

Je tiens, tout d'abord, à remercier mes directeurs de thèse, Rémi Sentis et Frédéric Nataf, sans qui rien n'aurait été possible. La qualité de leur encadrement sans oublier leur gentillesse ont été pour moi source d'une grande motivation me permettant de m'accomplir pleinement dans le travail de cette thèse. Mais surtout, ils m'ont fait confiance durant ces trois années et ont su me faire partager leur passion pour le calcul scientifique. Pour cela, je les remercie à nouveau.

Dans un second temps, j'adresse mes remerciements à l'ensemble des membres du jury, en particulier les rapporteurs Luc Giraud et Boniface Nkonga pour leur lecture attentive permettant l'amélioration du manuscrit par leurs remarques avisées.

En outre, pendant ces années de recherche, j'ai intégré l'équipe de développement du code *HERA*. Il n'aurait pas été pensable de m'immerger dans ce code sans les conseils et la patience de Philippe Ballereau et David Dureau. Je les en remercie grandement.

J'aimerais aussi remercier Gérard Meurant pour l'intérêt qu'il a porté à mon travail. Ses passages dans mon bureau ont toujours été déclencheur de recherches fructeuses.

Durant cette thèse, j'ai eu le privilège de faire un grand nombre de rencontres plus instructives et enrichissantes les unes que les autres. Je pense notamment à mon ancien collègue de bureau Gwénaël Salin, sans cesse à l'écoute et d'un avis toujours pertinent. J'y joins également Stéphane Del Pino et Pascal Havé qui ont guidés mes premiers pas en programmation. Enfin, je remercie Philippe Hoch pour sa bonne humeur constante et sa joie de vivre. Les repas et les pauses n'auraient pas été aussi "déjantés" sans lui.

Je remercie également les anciens et nouveaux du couloir des thésards, Benjamin, Paul-Edouard, les deux Julien, Constant, Céline, Lisl, Hugues, Marc-Antoine, Olivier, les sportifs du foot et de la boxe ainsi que ceux que j'oublie.

Pour finir, je tiens à apporter une attention particulière aux personnes qui m'ont toujours soutenu, ma famille, ma fiancée Sabrina pour son soutien au quotidien ainsi que mes amis.

à ma famille,
à Sabrina.

Table des matières

Introduction

Le *Laser Mégajoule* (LMJ) en construction au CEA-Cesta de Bordeaux constitue un élément clé du *programme simulation* de la Direction des Applications Militaires.

Le dispositif expérimental de *Fusion contrôlée par Confinement Inertiel* vise à comprimer fortement un mélange fusible de deutérium et tritium contenu dans une petite capsule appelée *micro-ballon*. Expérimenté depuis les années 60, ce schéma de fusion est devenu réaliste depuis le développement de lasers de puissance tel que le LMJ.

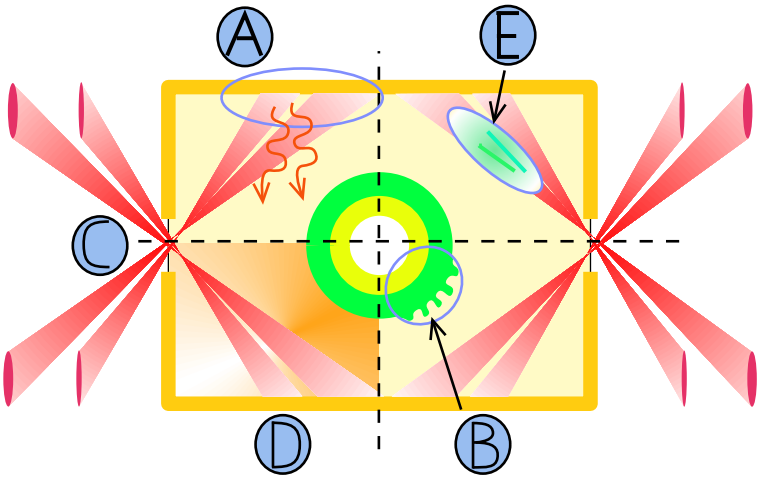


Fig. 1 – Lieux des principaux phénomènes physiques de la FCI

Le CEA-DAM a choisi la voie de l'attaque indirecte. Dans ce schéma, le micro-ballon est disposé au milieu d'une cavité cylindrique aux parois d'or que les faisceaux lasers éclairent. Sous l'impact des faisceaux lasers, les parois de la cavité explosent (D) et emmettent des rayons X (A). Ceci crée un plasma qui se détend fortement et vient impacter le micro-ballon dégradant ainsi son implosion. Pour pallier ces inconvénients, la cavité est remplie d'un gaz qui sert à contenir l'expansion des parois. Il faut donc fermer les trous d'entrée (C) avec un matériau transparent au rayonnement laser. Ce gaz étant très rapidement chauffé par les faisceaux lasers, un plasma se crée sur leurs trajectoires. Le contrôle de la propagation des faisceaux (E) lasers dans des plasmas de plusieurs centimètres est très difficile, et les phénomènes mis en jeu (*auto-focalisation*, *fi-*

lamentation, *déviation*, *instabilités paramétriques*, *rétro-diffusion*, etc.) sont très complexes. Le DT est ainsi porté à des conditions de température et de pression telles que la combustion thermonucléaire (B) devient significative et arrive à s'auto-entretenir : c'est *l'ignition*. Le temps de confinement est très bref, de l'ordre d'une centaine de picoseconde, et le dégagement de l'énergie est impulsionnel.

C'est dans le cadre de la propagation des faisceaux laser que s'inscrit ce travail. On s'intéresse ici à la simulation de la propagation de faisceaux lasers dans des plasmas. En particulier, la modélisation du faisceau laser grâce aux équations de Maxwell est impossible à réaliser numériquement sur des échelles macroscopiques raisonnables en raison des échelles caractéristiques du problème. De plus, les phénomènes de l'interaction se développent sur des temps beaucoup plus longs que la durée de traversée de la cible à la vitesse de la lumière.

Dans la littérature, il est d'usage d'approcher les équations de Maxwell par une équation de Schrödinger paraxiale [1], [2], [3], [4], [5], [6], [7] ou [8]. Elle est obtenue par une méthode de décomposition paraxiale (de type optique géométrique). Pourtant, on sait que la méthode paraxiale n'est valide que pour une propagation du faisceau laser dans une direction privilégiée et à condition que la densité électronique adimensionnée par la densité critique ne soit pas trop grande et ne varie pas trop. Dès que cette densité devient grande, il peut y avoir éclatement et réfraction à l'échelle macroscopique, ce qui peut conduire à un changement complet de la direction de propagation (dans la zone de la caustique) du faisceau laser. Cela nous pousse alors à considérer un couplage spatial entre deux modèles : d'une part un modèle paraxial, valide loin de la caustique et peu coûteux ; d'autre part, près de la caustique, on utilise un modèle basé sur les équations de Maxwell fréquentielles pour un champ électrique scalaire (i.e. polarisation linéaire). On appellera également ce dernier modèle Helmholtz par abus de langage. **Notre objectif est de mettre au point un solveur rapide pour les équations de Maxwell fréquentielles et d'opérer le couplage avec la zone paraxiale.** Par ailleurs, ces deux modèles sont aussi couplés aux équations de l'hydrodynamique par l'introduction d'une force supplémentaire dûe au gradient de l'intensité laser (bien sûr résolue dans tout le domaine).

Précisons qu'on ne peut pas comparer nos résultats avec ceux de la littérature. En effet, la plupart des résultats disponibles sont obtenus avec des codes basés sur le modèle paraxial donc ne pouvant traiter des faisceaux courbes. Il existe toutefois des codes de résolution du modèle Helmholtz écrits par des physiciens utilisant une transformée de Fourier dans une des directions ce qui permet des variations de densité jusqu'à la densité critique mais qui oblige à mettre de très larges zones d'absorption des ondes près des frontières perpendiculaires à cette direction. Toutefois, ces codes traitent généralement des faisceaux droits sur des domaines de petites tailles. Voir par exemple [9], [10] et [11]. On peut aussi noter l'article [12] mentionnant uniquement des résultats mais qui renvoie pour la description de la méthode numérique à un paragraphe de [13].

Cette thèse est composée de 7 chapitres. Elle est structurée comme suit :

- Dans le **premier chapitre**, les différents modèles des phénomènes physiques de l'interaction laser-plasma utilisés seront présentés. En particulier, pour la modélisation du plasma, la densité électronique adimensionnée $N(x, y)$ sera

obtenue à partir d'un modèle hydrodynamique type Euler simplifié. On pourra alors effectuer la décomposition de la densité électronique N selon ses échelles de variation

$$N(x,y) = N_0(x) + \delta N(x,y)$$

où N_0 est la densité homogénéisée dite moyenne, constante ou variable, et δN une perturbation ou fluctuation. Pour la modélisation de la propagation des ondes, on s'intéressera dans un premier temps à l'obtention du modèle Maxwell fréquentiel (qui est une équation de Schrödinger) vérifié par le champ laser ψ

$$\left[2i\frac{\epsilon}{c}\frac{\partial}{\partial t} + \epsilon^2\Delta + i\nu + (1 - N(x,y))\right]\psi(x,y,t) = 0$$

où ϵ est l'inverse du nombre d'onde dans le vide, c la vitesse de la lumière, ν un coefficient d'absorption.

Dans un second temps, en notant l'enveloppe laser u vérifiant l'approximation

$$\psi(x,y,t) \simeq u(x,y)\, e^{i\frac{\vec{K}.\vec{x}}{\epsilon}},$$

on définira le modèle paraxial suivant

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + iu\nabla\cdot\vec{K} + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u - \epsilon^{-1}\left(\delta N\right)u = 0$$

où $\vec{K}$ est le vecteur de propagation de l'onde et $\Delta_\perp$ le Laplacien dans la direction orthogonale à $\vec{K}$.

Notre but est d'effectuer une résolution sur un domaine d'une taille réaliste de plusieurs milliers de longueurs d'onde dans chaque direction. C'est un véritable challenge numérique nécessitant une stratégie adaptée. Les modèles hydrodynamique et paraxial étaient déjà implantés dans le code *HERA*. Les efforts de résolution sont donc centrés sur l'équation de Maxwell fréquentiel. Celle-ci se fait de façon totalement implicite en temps ; on est alors conduit à la résolution d'une équation de type Helmholtz. Or, on sait qu'une discrétisation précise nécessite au moins dix points par longueur d'onde entraînant la prise en compte de maillages de l'ordre de plusieurs dizaines à centaines de millions d'inconnues. A titre de comparaison, le maillage pour les modèles hydrodynamique et paraxial peut être dix fois plus grossier dans chaque direction. On comptera donc utiliser un solveur rapide avec la forte contrainte que l'équation d'Helmholtz est à coefficient variable. Les fluctuations de densité faisant perdre la séparabilité du problème, il conviendra d'utiliser une méthode itérative. Enfin, les conditions sortantes sont un aspect très important des problèmes de propagation d'ondes. La référence en la matière sont les couches absorbantes *PML* (*Perfectly Matched Layer*) [21]. Leur prise en compte accentuera encore les difficultés avec la perte locale de propriétés algébriques nécessaires à l'utilisation d'un solveur rapide de type Réduction Cyclique. Tout ceci nous poussera alors à utiliser une stratégie de décomposition de domaine type Schwarz associée à une méthode itérative de Krylov pour d'une part isoler les perturbations liées aux couches *PML* et d'autre part traiter les fluctuations de densité. La difficulté de cette thèse est de faire cohabiter toutes ces méthodes performantes et récentes.

• Après avoir introduit les différentes équations et motivations du problème, la discrétisation de ces équations sera présentée dans le **second chapitre**. Les échelles caractéristiques des grandeurs sont très différentes suivant les modèles. Typiquement, un pas de maillage de l'ordre de la longueur d'onde sera suffisant pour les modèles hydrodynamique et paraxial tandis que pour le modèle Helmholtz, il faudra considérer une petite fraction de la longueur d'onde. Ceci amènera alors à considérer deux maillages distincts se recouvrant ; un premier dit *grossier* pour les grandeurs fluide et du modèle paraxial et un second dit *fin* pour le modèle Helmholtz. Pour l'hydrodynamique, on utilise un schéma explicite de type Lagrange plus transport. Pour le modèle paraxial, on utilise une méthode à pas fractionnaires proposée par R. Sentis [3] consistant à considérer une étape d'advection puis une étape de diffraction. Pour le modèle Helmholtz, on utilisera une discrétisation différences finies classique. En particulier, on verra qu'avec les couches *PML*, l'opérateur Laplacien se discrétise comme

$$\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\delta x^2} + a_j \frac{a_{j+1/2}(\psi^n_{i,j+1} - \psi^n_{i,j}) - a_{j-1/2}(\psi^n_{i,j} - \psi^n_{i,j-1})}{\delta y^2}$$

où $a_{j+1/2} = \frac{a_{j+1}+a_j}{2}$ avec $a_j = \frac{\mathrm{i}\sqrt{1-N_j}}{\mathrm{i}\sqrt{1-N_j}+\sigma_j}$. L'impact des *PML* se traduit par la perte locale de la symétrie de l'opérateur. En posant $\delta N = 0$, la matrice de discrétisation associé au schéma s'écrira alors

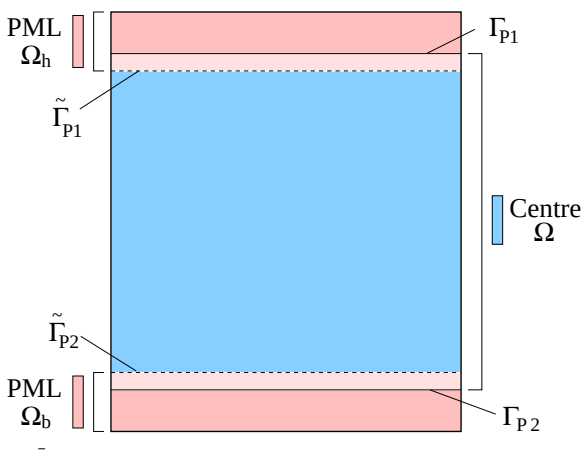
$$\begin{bmatrix} B_b & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & B_h \end{bmatrix}$$

où les matrices B_h et B_b contiennent les couches *PML* et ne sont pas symétriques. Notons n_x et n_y le nombre de points en x et y. Sans tenir compte des couches *PML*, on obtient une matrice carré symétrique (non hermitienne) de dimension $n_x \times n_y$. La matrice $T \in \mathbb{R}^{n_x \times n_x}$ est une constante fois l'identité et $A \in \mathbb{C}^{n_x \times n_x}$ est tridiagonale. Dans le cas sous-critique ($N < 1$ en sortie de boîte, à droite), il est intéressant d'améliorer la qualité de la condition classiques d'onde sortante

$$\left(\epsilon \frac{\partial}{\partial n} - i\sqrt{1-N}\right)\psi = g.$$

On présentera tout d'abord comment ces conditions s'obtiennent et comment être plus précis au niveau continu en utilisant la théorie des conditions absorbantes [18]. Au niveau discret, on étudiera en particulier une montée en ordre de la discrétisation de la dérivée normale. Une étude numérique sera effectuée pour comparer les différentes approches entreprises. Le gain de précision possible est remarquable. Enfin, on s'intéressera au traitement de la condition sur le bord entrant du domaine permettant le couplage entre les modèles paraxial et Helmholtz.

• Le **troisième chapitre** traite de la stratégie de décomposition de domaine utilisée et des manipulations algébriques engendrées. L'idée de la décomposition de domaine est ici d'isoler les couches *PML*.



Dans chacun des sous-domaines Ω_b, Ω_g et Ω_h, on résout respectivement

$$\begin{cases} \left(\epsilon^2\tilde{\Delta} + (1-N_0) + i\nu\right)\psi_{P1} &= 0 & \text{dans} \quad \Omega_b \\ \left(\epsilon^2\Delta + (1-N_0) + i\nu\right)\psi_G &= \delta N\psi_G & \text{dans} \quad \Omega_g \\ \left(\epsilon^2\tilde{\Delta} + (1-N_0) + i\nu\right)\psi_{P2} &= 0 & \text{dans} \quad \Omega_h \end{cases}$$

où on a noté l'opérateur

$$\tilde{\Delta} = \mu(y)\frac{\partial}{\partial y}\mu(y)\frac{\partial}{\partial y} + \frac{\partial^2}{\partial x^2}$$

avec $\mu(y)$ traduisant l'impact des couches PML.

Pour ψ_{P1}, une condition de raccord sur $\tilde{\Gamma}_{P1}$ doit être écrite, elle est du type

$$\mu(y)\frac{\partial\psi_{P1}}{\partial y} + \alpha\,\psi_{P1} = \frac{\partial\psi_G}{\partial y} + \alpha\,\psi_G \text{ sur } \tilde{\Gamma}_{P1}$$

où α est un coefficient complexe à déterminer et de même pour les autres conditions sur ψ_G et ψ_{P2}.

L'écriture matricielle d'une telle décomposition sans et avec recouvrement ainsi que sa résolution par une stratégie de préconditionnement sera la motivation de ce chapitre. Dans une première partie, on montrera formellement l'équivalence entre la solution du problème initial et la solution du problème décomposé au niveau algébrique. Après un rappel sur les techniques itératives de Krylov, on décrira la méthode itérative associée à la résolution du système linéaire de la décomposition de domaine. En particulier, il sera possible d'avoir une approche de préconditionnement particulièrement adaptée. Ainsi, on aura la structure par bloc suivante

$$A_D = \begin{pmatrix} A_H & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_B \end{pmatrix} \qquad \text{et} \qquad A_E = \begin{pmatrix} 0 & C_1 & 0 \\ C_2 & 0 & C_3 \\ 0 & C_4 & 0 \end{pmatrix}$$

où A_D est la matrice composée des discrétisations de chaque domaine et A_E les connections entre sous-domaines. Notons δ_N la matrice diagonale associée à

δN. On résoudra alors le système linéaire

$$(A_D - A_E + \delta_N)X = b.$$

Donc en utilisant A_D^{-1} comme matrice de préconditionnement, cela revient à

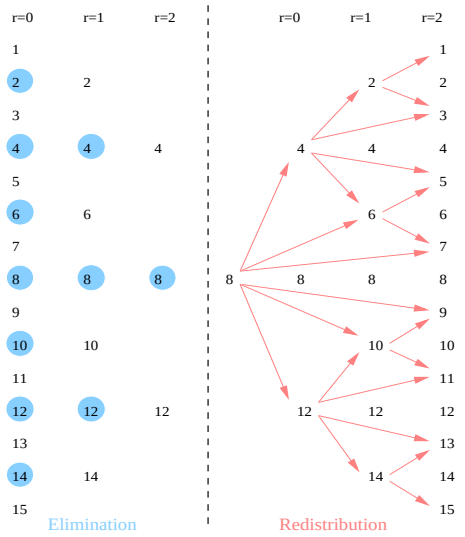
$$(I - A_D^{-1}(A_E - \delta_N))X = A_D^{-1}b.$$

Le bloc A_I sera inversé par un solveur rapide. La méthode de décomposition de domaine utilisera les méthodes de Krylov dont *GMRES*.

- Dans le **quatrième chapitre**, on présentera le solveur rapide utilisé, la Réduction Cyclique dont une multitude de variantes existe. On peut citer par exemple la méthode très connue *FACR* [34]. Toutefois, l'utilisation de méthodes rapides type transformée de Fourier ne sera pas applicable à notre problème, excluant ainsi ce type de méthode. La discrétisation de notre problème amène à considérer un système linéaire tridiagonal par bloc d'une structure très simple.

$$\begin{pmatrix} B & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & B \end{pmatrix}$$

La Réduction Cyclique consistera à éliminer les inconnues par combinaison linéaire de manière récursive. On se restreindra dans cette partie à deux grandes



approches a priori adaptées ; une première dite *standard* basée sur une analyse spectrale et une seconde appelée *PSCR* [22] basée sur une technique de solutions partielles [26]. L'idée sera de comparer les méthodes et surtout de justifier le choix d'utiliser la méthode *standard* et non la méthode *PSCR* récente et très utilisée pour les problèmes d'Helmholtz à coefficient constant. En particulier, l'avantage indéniable de la méthode *standard* sur la méthode *PSCR* sera sa

simplicité algorithmique et de mise en oeuvre. En effet, la méthode s'écrit simplement et le noeud de la méthode sera une séquence de produit d'une matrice dense par un ensemble de vecteur. Notons que d'un point de vue historique, la Réduction Cyclique fut éprouvée sur des problèmes elliptiques simples. Un point crucial de cette méthode sera la capacité à calculer très précisément et rapidement le spectre complet de la matrice A tridiagonale symétrique (non hermitiennne).

- Basé sur des séquences d'opérations algébriques sur les valeurs et vecteurs propres, la performance du solveur rapide dépendra directement de la précision de la décomposition spectrale. Le **cinquième chapitre** présente une méthode ultra rapide et performante pour obtenir les valeurs et vecteurs propres d'une matrice notée A tridiagonale complexe symétrique non-hermitienne. Autrement dit, on cherchera les matrices Λ et Q vérifiant

$$A = Q\Lambda Q^T.$$

Il existe de nombreuses méthodes permettant d'obtenir les valeurs propres dont le fameux algorithme QR. Par contre, aucune de ces méthodes ne tient compte du caractère tridiagonal de la matrice. On s'intéressera à l'algorithme LR récemment revu par B. Parlett [14]. D'un point de vue historique, cette méthode précède l'algorithme QR et est basée sur le même procédé itératif de multiplication de matrices issues ici de décompositions LU. De plus, les mêmes techniques d'accélération de la méthode par décalage seront utilisées. Bien qu'applicable en théorie dans le cas de matrice générale, cette méthode est sans doute la plus performante pour le cas des matrices tridiagonales. En effet, elle repose sur des décompositions LU, très bien adaptées pour ce type de matrice.

$A_1 = A,$
pour $k = 1, 2, \ldots :$
$\quad A_k = LU$
$\quad A_{k+1} = UL$
fin

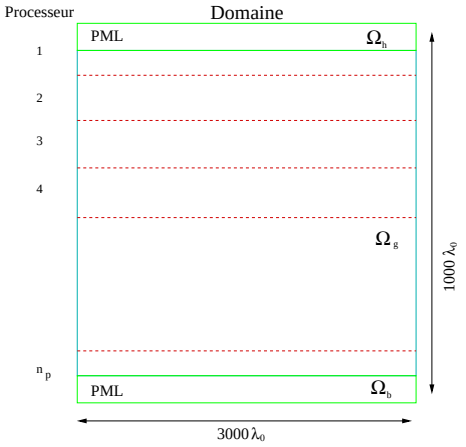
Pour les vecteurs propres, on cherchera à déterminer le noyau des applications $A - \lambda_i I$ où $\lambda_i \in \mathbb{C}$. Cela exigera donc de résoudre les systèmes suivants :

$$\begin{bmatrix} a_{1,1} - \lambda_i & a_{1,2} & & \\ a_{1,2} & a_{2,2} - \lambda_i & \ddots & \\ & \ddots & \ddots & a_{n-1,n} \\ & & a_{n-1,n} & a_{n,n} - \lambda_i \end{bmatrix} \begin{pmatrix} v_i^1 \\ v_i^2 \\ \vdots \\ v_i^n \end{pmatrix} = 0.$$

Une manière naturelle pour trouver v_i est de choisir un indice pivot k et de poser $v_i^k = 1$. S.I. Dhillon [15] a récemment développé une méthode très performante donnant des vecteurs propres directement orthogonaux évitant ainsi les réorthogonalisations des vecteurs propres. Elle est basée sur un choix astucieux du pivot pour chaque vecteur propre dépendant de la factorisation particulière de la matrice du noyau. Une technique de raffinage des valeurs propres sera également employée. Initialement prévue pour les cas réels, on intéressera ici à

l'implémentation dans le cas complexe ce qui est nouveau à notre connaissance. La comparaison avec la méthode classique QR dans la bibliothèque $LAPACK$ montrera des gains d'un facteur au minimum 40 en temps calcul et une excellente précision.

- La parallélisation de toutes ces techniques sera alors abordée dans un **sixième chapitre**. On intégrera un module dans le code parallèle *HERA* calculant l'hydrodynamique et le paraxial. Notre but sera de paralléliser et d'optimiser nos méthodes afin d'effectuer un calcul massivement parallèle sur un domaine réaliste avec couplage entre les modèles paraxial, Helmholtz et hydrodynamique. Pour obtenir de bonnes performances, l'aspect informatique scientifique aura une place cruciale. En effet, on cherchera à exploiter au mieux l'architecture des machines disponibles. La décomposition de domaine, la répartition des charges ainsi que les protocoles de communication seront les points fondamentaux de notre étude. La distribution des données sera dictée par celle effectuée dans le code *HERA*. Le maillage *grossier* fluide sera ainsi distribué en bandes horizontales de manière équilibrée entre chaque processeur. On choisira d'utiliser une approche



basée sur *MPI* pour les communications internoeuds et sur le *multithreading* pour gérer au mieux les données et éviter les redondances de données. Les opérations coûteuses de la résolution sont les produits matrices pleines Q et Q^T par la matrice issue du second membre dans le solveur rapide. Le second membre sera distribué naturellement suivant le maillage. D'un point de vue stratégique, il sera primordial que chaque processus dispose de la matrice Q. Les séquences de produit matrice-matrice n'auront alors pas besoin de synchronisation. C'est dans ce contexte que la stratégie hybride *MPI-multithreading* prendra toute son importance. Dans la mesure où l'on souhaite plutôt bénéficier de la localité des données, on se devra alors d'avoir une grande efficacité dans la parallélisation des opérations algébriques. Une bonne gestion du cache devient alors primordiale mais difficilement contrôlable. En particulier, le caractère très mobile des threads sur les processeurs du noeud ne permet souvent pas une optimisation de gestion du cache. Pour pallier ces effets et ceux des temps de gestion des threads par le système, il faudra avoir des tâches particulièrment conséquentes à paralléliser. Dès lors, le *multithreading* ne s'appliquera pas dans toutes les opérations,

notamment celles de petites tailles. L'opération du produit matrice-matrice sera suffisamment coûteuse pour que la programmation par *multithreading* donne de très bons résultats. Au cours de la Réduction Cyclique, les processus éliminent au fur et à mesure les inconnues qu'ils contiennent jusqu'à ce qu'il ne reste plus qu'un seul processus contenant le problème réduit. On voit ici le paradoxe de la parallélisation de méthodes de réduction. Dès lors, on jouera sur deux tableaux en parallélisant à la fois les étapes de la méthode de Réduction Cyclique et les opérations effectuées durant ces étapes. La très grande taille du domaine associée au fait d'utiliser une méthode itérative où l'on doit stocker les directions de descente seront des contraintes très fortes à gérer simultanément. Du fait de la répartition en bande des données, augmenter le nombre de processeurs reviendra à diminuer le nombre de vecteurs à multiplier par base des vecteurs propres. Il est donc naturel que le temps calcul soit divisé par deux si on multiplie par deux le nombre de processeurs. On constatera des très bons résultats de scalabilté pour un problème fixe bien qu'au cours de la Réduction Cyclique, le nombre de processeurs calculant diminue du fait précisement de la réduction.

- Le **dernier chapitre** présente les résultats obtenus avec le code *HERA* (et le module Helmholtz) par des simulations parallèles de cas réalistes de propagation d'un faisceau laser. Le faisceau laser va se propager et créer de fortes perturbations en creusant au fil du temps le plasma. L'interaction du plasma sur le faisceau va induire des phénomènes tels que la focalisation ou la filamentation du laser. Le comportement de la méthode itérative sera directement lié aux fluctuations de densité. On présentera tout d'abord un cas de validation d'une propagation sans incidence dans un plasma sous-critique. L'idée ici sera de comparer une simulation par le modèle paraxial à une simulation par le modèle Helmholtz. On constatera alors la cohérence et la qualité des résultats obtenus par le modèle Helmholtz. On présentera ensuite pour le même cas une simulation avec couplage entre les modèles paraxial et Helmholtz. Notons que ce problème de couplage entre deux modèles différents dans le cadre général d'un couplage avec un modèle hydrodynamique semble nouveau. Nous avons exhibé une méthode robuste et précise pour la résolution du problème couplé. Les simulations suivantes effectuées permettront l'étude de la propagation de faisceaux composés d'un à plusieurs points chauds (*speckle*) dans un plasma où la densité variera jusqu'à la densité critique. On observera alors une forte variation de la direction de propagation du faisceau en plus des autres phénomènes du cas sous-critique. On validera le couplage des modèles paraxial et Helmholtz en présentant une comparaison avec le modèle Helmholtz seul. Précisons que les problèmes réalistes traités sont résolus typiquement sur des maillages de 80 millions et nécessitent 128 processeurs. On présente également un cas d'étude de deflection du laser de 200 millions de points résolus sur 256 et 512 processeurs.

Table des figures

Première partie

Présentation globale et discrétisation

1

Présentation générale des équations et problématique

Sommaire

1.1 Notations générales

Pour une présentation physique des phénomènes, voir [4]. Pour une approche plus mathématique, voir [3] par exemple.

On s'intéresse à l'interaction d'un laser de forte intensité et d'un plasma homogène à l'échelle mésoscopique (de l'ordre du micron). On pose le problème sur un domaine $\mathcal{D}$ de $\mathbb{R}^2$, typiquement un rectangle d'une longueur et d'une largeur de plusieurs milliers de longueurs d'onde. On note $\partial\mathcal{D}$ son bord. On introduit les notations suivantes

- q_e, m_e, la charge et la masse des électrons ;
- λ_0, la longueur d'onde du laser ;
- c, la vitesse de la lumière ;
- $k_0 = \frac{2\pi}{\lambda_0}$, le nombre d'onde du laser ;
- $\omega_0 = k_0\, c$, la pulsation du laser ;
- ε_0, la perméabilité du milieu ;
- N_e, la densité électronique ;
- N_i, la densité ionique ;
- Z, m_e, le niveau de ionisation et la masse ionique ;
- N_c, la densité critique définie par $\omega_0^2 = N_c q_e^2 (\epsilon_0 m_e)^{-1}$;
- ν_{ei}, la fréquence de collision électron-ion.

Remarque la fréquence ν_{ei} est proportionnelle à la densité ionique et dépend de la température des électrons. On la considère généralement égale à la densité ionique multipliée par une constante (ou une fonction qui est constante dans le domaine de simulation que nous considérons).

Nous nous limitons dans cette étude au cas où le laser est monochromatique, c'est-à-dire à λ_0 fixé.

1.1.1 Des équations de Maxwell à l'équation d'Helmholtz

Le faisceau laser est caractérisé par les champs électromagnétiques que l'on note $\mathcal{E}$ et $\mathcal{B}$. On note également le courant électrique $\mathcal{J}$. Ces champs peuvent être décomposés en un champ électrostatique et une composante oscillante rapide. Les champs électromagnétiques rapidement oscillants satisfont les équations de Maxwell

$$\begin{array}{llll} i) & \dfrac{\partial}{\partial t}\mathcal{E} - c^2 \mathrm{rot}\mathcal{B} + \varepsilon_0^{-1} J = 0, & ii) & \dfrac{\partial}{\partial t}\mathcal{B} + \mathrm{rot}\mathcal{E} = 0, \\ iii) & \mathrm{div}\mathcal{E} = \dfrac{q_e}{\varepsilon_0}(N_i - N_e), & iv) & \mathrm{div}\mathcal{B} = 0. \end{array}$$

Alors, en utilisant la dérivée temporelle des deux premières équations, on obtient l'équation des ondes classique

$$\frac{\partial^2}{\partial t^2}\mathcal{E} - c^2 \Delta \mathcal{E} = -\varepsilon_0^{-1} \frac{\partial}{\partial t}\mathcal{J}. \tag{1.1}$$

On doit coupler cela à une équation de relation de fermeture sur le courant $\mathcal{J}$

$$\frac{\partial}{\partial t}\mathcal{J} + \nu_{ei}\mathcal{J} = \frac{q_e^2}{m_e} N_e \mathcal{E}. \tag{1.2}$$

Si ν_{ei} est nul, on aurait une expression simple pour le second membre. Si il est non nul, on doit trouver une relation de fermeture pour le terme $\frac{\partial \mathcal{J}}{\partial t}$. Pour cela, on remarque que $\nu_{ei} << \omega_0$ et que le champ $\mathcal{E}$ est rapidement oscillant pour la pulsation ω_0. En utilisant (1.2), une approximation grossière de $\mathcal{J}$ est

$$\mathcal{J} \simeq -\frac{1}{\omega_0^2}\frac{\partial^2}{\partial t^2}\mathcal{J} \simeq -\frac{1}{\omega_0^2}\frac{q_e^2}{m_e} N_e \frac{\partial}{\partial t}\mathcal{E}.$$

Donc,

$$\frac{\partial}{\partial t}\mathcal{J} \simeq \frac{q_e^2}{m_e} N_e \left(\mathcal{E} + \frac{\nu_{ei}}{\omega_0^2} \frac{\partial}{\partial t} \mathcal{E} \right) \tag{1.3}$$

et l'equation des ondes (1.1) devient

$$\frac{\partial^2}{\partial t^2}\mathcal{E} - c^2 \Delta \mathcal{E} + \frac{q_e^2}{\varepsilon_0 m_e} N_e \mathcal{E} + \frac{q_e^2}{\varepsilon_0 m_e} N_e \frac{\nu_{ei}}{\omega_0^2} \frac{\partial}{\partial t} \mathcal{E} = 0$$

ou encore

$$\frac{\partial^2}{\partial t^2}\mathcal{E} - c^2 \Delta \mathcal{E} + \omega_0^2 \frac{N_e}{N_c} \mathcal{E} + c\, \nu_0 \frac{\partial}{\partial t} \mathcal{E} = 0 \quad \text{avec} \quad \nu_0 = \nu_{ei} \frac{N_e}{c N_c}. \tag{1.4}$$

Cette équation est habituellement appelée « équation de Klein-Gordon amortie ».

On introduit maintenant l'enveloppe temporelle des quantités oscillant rapidement. En notant *c.c.* le complexe conjugué d'une quantité, on peut écrire

$$\mathcal{E} = i\psi e^{-i\omega_0 t} + c.c. \tag{1.5}$$

où ψ est lentement variable en temps.

En utilisant la fermeture précédente (1.3), on a

$$\mathcal{J} = -\left(1 - i\frac{\nu_{ei}}{\omega_0^2}\right) \frac{q_e^2}{\omega_0 m_e} N_e \psi e^{-i\omega_0 t} + c.c.$$

De l'expression (1.5), on déduit les formules de dérivation en temps suivantes :

$$\begin{cases} \dfrac{\partial}{\partial t}\mathcal{E}(t,x,y) = \left(\dfrac{\partial}{\partial t} - i\omega_0\right) \psi(t,x,y) e^{-i\omega_0 t}, \\ \dfrac{\partial^2}{\partial t^2}\mathcal{E}(t,x,y) = \left(\dfrac{\partial^2}{\partial t^2} - \omega_0^2 - 2i\omega_0 \dfrac{\partial}{\partial t}\right) \psi(t,x,y) e^{-i\omega_0 t}. \end{cases}$$

En insérant ces développements dans (1.4), on obtient

$$\left(\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - k_0^2 - 2i\frac{k_0}{c}\frac{\partial}{\partial t}\right)\psi - \Delta\psi + k_0^2 \frac{N_e}{N_c}\psi + \nu_0 \left(\frac{1}{c}\frac{\partial}{\partial t} - ik_0\right)\psi = 0.$$

Maintenant, comme ψ est lentement variable en temps, on peut négliger $\frac{\partial^2}{\partial t^2}\psi$ par rapport à $\frac{\partial}{\partial t}\psi$. De plus, le terme $\frac{1}{c}\frac{\partial}{\partial t}\psi$ est également négligeable comparé à $ik_0\psi$. A partir de (1.4), on retrouve alors l'équation de Schrödinger linéaire

$$\boxed{\left[2i\frac{k_0}{c}\frac{\partial}{\partial t} + \Delta + ik_0\nu_0 + k_0^2(1-N)\right]\psi(x,y,t) = 0.} \tag{1.6}$$

où on a posé $N = \frac{N_e}{N_c}$.

Remarque Si ψ est indépendante du temps, on retrouve l'équation des ondes fréquentielle (ou Helmholtz) dont l'enveloppe ψ est solution :

$$\boxed{\left[\Delta + ik_0\nu_0 + k_0^2(1-N)\right]\psi(x,y) = 0.} \tag{1.7}$$

Remarque Dans le cas où ψ est lentement variable en temps, on doit garder la dérivée temporelle $\frac{\partial \psi}{\partial t}$. D'un point de vue numérique, on est alors conduit après une discrétisation implicite en temps à résoudre à chaque pas de temps δt

$$\boxed{\left[\Delta + 2ik_0\nu_0 + k_0^2(1-N)\right]\psi + \frac{2ik_0}{c\delta t}\psi = \frac{2ik_0}{c\delta t}\psi^{ini}} \tag{1.8}$$

où ψ^{ini} est la solution au pas de temps précédent. Par rapport à (1.7), un second membre est ajouté à l'équation.

Dans toute la suite, on fera référence pour (1.8) au modèle Schrödinger-Helmholtz (ou par abus d'écriture Helmholtz).

1.1.2 Conditions limites

Pour les équations (1.6) et (1.7), en notant e_b le vecteur unitaire caractérisant la direction de propagation du faisceau laser, on doit dans un premier temps considérer la partie éclairée du bord Γ^{in} défini par

$$\Gamma^{in} = \{x \in \partial\mathcal{D}, \text{ tel que } e_b.n < 0\}, \quad n, \text{ la normale sortante.}$$

On suppose l'onde incidente ψ^{in} de la forme $\alpha^{in}e^{ik_0\sqrt{1-N}e_b.x}$, sachant que $\alpha^{in} = \alpha^{in}(x)$ est la restriction sur Γ^{in} d'une fonction régulière. La condition entrante sur le bord Γ^{in} est

$$\left(n.\nabla + ik_0\sqrt{1-N}e_b.n\right)\left(\psi - \psi^{in}\right) = 0. \tag{1.9}$$

Dans un second temps, si on note la frontière $\Gamma^{out} = \partial\mathcal{D} - \{\Gamma^{in}\}$ (où $e_b.n \geq 0$), la condition limite sur le bord Γ^{out} s'écrit

$$\left(n.\nabla - ik_0\sqrt{1-N}\right)\psi = 0. \tag{1.10}$$

Remarque Dans la suite, on considère aussi sur Γ^{out} des couches *PML* (perfectly matched layers) du type décrit dans [21]. Ces conditions s'inscrivent dans le cadre des couches absorbantes.

1.1.3 Problématique

Précisons que les échelles caractéristiques pour l'équation (1.4) sont de l'ordre de ω_0^{-1} c'est-à-dire d'environ 10^{-16} secondes et de quelques 10^{-5} centimètres pour la longueur d'onde du laser.

En considérant les échelles caractéristiques, on constate que la discrétisation de l'équation des ondes (1.4) à l'échelle mésoscopique est numériquement trop contraignante pour envisager la prise en compte de grands domaines tel que $\mathcal{D}$ (ou un passage en trois dimensions). Or, une discrétisation donnant des résultats précis pour nos équations nécessite au moins dix mailles par longueur d'onde. Typiquement, pour un domaine 2D réaliste de dimensions $[5000\lambda_0 \times 3000\lambda_0]$, le maillage considéré est composé d'environ 1,5 milliards d'inconnus. Il n'est

donc pas raisonnable d'envisager un tel maillage. Pour s'affranchir (du moins partiellement) de ces problèmes, il est courant dans les études d'interaction laser plasma d'utiliser une approche paraxiale (ou type W.K.B.) où l'on suppose connue la direction principale du laser. Par contre, on montre que cette approximation n'est valide que sous des hypothèses très restrictives, notamment pour des densités ne dépassant pas 25% de la densité critique. Notre but est de faire cohabiter le modèle paraxial là où il est valide avec le modèle défini par l'équation (1.8) ailleurs. La situation standard est la suivante : le modèle paraxial est valide sur environ 80% du domaine. Ailleurs, la densité électronique varie fortement induisant ainsi le changement de direction de propagation du laser. On ne résout l'équation d'Helmholtz (1.7) que dans cette zone.

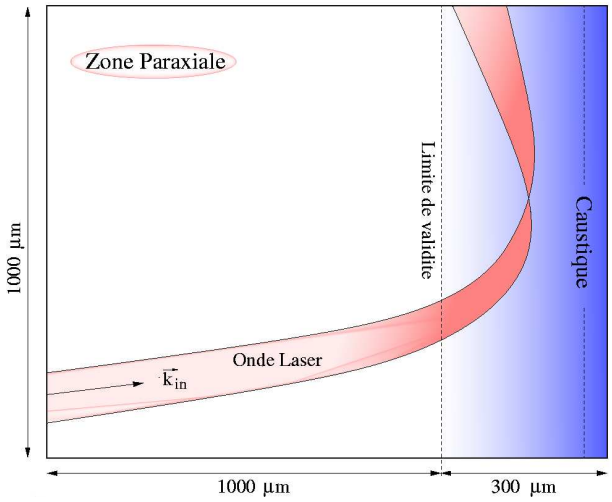


FIG. 1.1 – Domaine global

Rappelons que la densité critique est la densité à partir de laquelle aucune onde électromagnétique ne peut se propager. Il est intéressant de noter que pour les équations (1.6) et (1.8) si on a $N_e \geq N_c$ (ou $N \geq 1$), alors ces équations ne traduisent plus un phénomène propagatif mais un phénomène d'absorption (l'équation des ondes devient une équation elliptique). D'un point de vue physique, aucune onde ne se propage. D'un point de vue numérique, on doit vérifier que si la densité est supérieure à la densité critique, l'intensité laser devient quasiment nulle.

On s'intéresse maintenant à l'obtention du modèle paraxial.

1.2 Approximation optique géométrique

On donne ici quelques résultats de l'optique géométrique afin de bien comprendre la méthode paraxiale qu'on va mettre en œuvre. On se place maintenant dans la zone où il n'y a pas de forte variation de densité électronique.

L'approximation W.K.B. consiste à écrire pour l'équation (1.7)

$$\psi(t,x,y) \simeq u(x,y)\, e^{i\, k_0 \phi(x,y)} \tag{1.11}$$

et à identifier, pour k_0 tendant vers $+\infty$, les fonctions u et ϕ supposées être lentement variables par rapport à l'espace. On vérifie que :

$$\begin{cases} \nabla\psi = [\nabla u + ik_0 u\nabla\phi]\, e^{ik_0\phi} \\ \Delta\psi = \left[\Delta u + 2ik_0\nabla\phi\cdot\nabla u + ik_0 u\,(\nabla\cdot\nabla\phi) - uk_0^2\,|\nabla\phi|^2\right] e^{ik_0\phi} \end{cases}$$

et en développant dans (1.7), on obtient alors :

$$\Delta u + ik_0\,[\,2\nabla\phi\cdot\nabla u + u\,(\nabla\cdot\nabla\phi) + 2\nu_0 u\,] - k_0^2 u\left[\,|\nabla\phi|^2 - (1-N)\,\right] = 0.$$

Cette expression est un polynôme en k_0 à identifier avec le polynôme nul. Ses différents coefficients sont donc nécessairement nuls. On voit ainsi que la phase ϕ est solution de l'eikonale $|\nabla\phi|^2 = 1 - N$. On impose sur Γ^{in} la direction de propagation $\nabla\phi$ (qui est alors celle du laser incident). D'autre part, en écrivant $\vec{K} = \nabla\phi$, u vérifie l'équation d'advection

$$\vec{K}\nabla u + \frac{u}{2}\nabla\cdot\vec{K} + \nu_0 u = 0$$

avec $u = u^{in}$ comme condition sur le bord Γ^{in} représentant l'intensité du laser. On remarque que l'énergie $|u|^2$ satisfait

$$2\nu_0\,|u|^2 + \nabla\cdot\left(\vec{K}\,|u|^2\right) = 0.$$

Par cette méthode d'optique géométrique, on modélise donc les phénomènes de réfraction et d'absorption mais pas la diffraction. Dans toute la littérature traitant de l'interaction laser plasma, les effets de diffraction étant principalement transverses au laser ([1], [2], [3], [4], [5], [6], [7] ou [8]), on utilise une modélisation de ces phénomènes grâce à un laplacien transverse en supposant que la direction de propagation du faisceau est fixée. C'est ce qu'on appelle l'approximation paraxiale dans laquelle on décompose la densité N en une densité moyenne et une perturbation.

1.3 Modèle paraxial

1.3.1 Décomposition de la densité

On suppose la densité du plasma N connue. Dans notre cas, on a défini N comme étant le rapport entre la densité électronique N_e et la densité de coupure N_c du plasma. En pratique, la densité N_e est obtenue par la résolution de l'hydrodynamique du plasma dans le domaine $\mathcal{D}$. La densité de coupure, quant à elle, ne dépend que de la fréquence laser ω_0. Le laser traverse une grande partie du domaine de densité presque constante puis entre dans la zone du plasma où la densité varie fortement pour approcher la densité de coupure N_c. On observe alors un changement de direction du laser due à une caustique de type pli. Dans un premier temps, remarquons que cette caustique est induite par les variations de la densité N_e traduisant les changements de pression et densité dans le plasma sur le domaine. Il est ensuite utile de préciser que N_e est presque

constante "loin" à l'entrée du domaine mais peut varier à l'intérieur d'une part du fait de fluctuations dues à l'interaction laser-plasma et d'autre part à l'échelle macroscopique (à cause de l'évolution du plasma à l'échelle hydrodynamique). En utilisant ces remarques, on décompose alors N en fonction de ses variations par la prise en compte d'un développement asymptotique classique. Pour l'instant, on écrit simplement :

$$N(x,y) = N_0(x) + \delta N(x,y) \tag{1.12}$$

où N_0 est la densité homogénéisée dite moyenne, constante ou presque, et δN une perturbation ou fluctuation. On fait bien sûr l'hypothèse que la densité moyenne est inférieure à la densité de coupure : $N_0 < 1$.

Par la suite, on étudiera l'approximation paraxiale pour deux comportements distincts de la densité moyenne : le cas où N_0 est constante puis celui lentement variable. Notons que dans ce cas l'approximation paraxiale fait intervenir entre autres un gradient transverse à l'onde dont la définition s'avère délicate lorsque la direction du vecteur de propagation de l'onde varie.

On se place maintenant dans le cas particulier où la densité N_0 est constante.

1.3.2 Equation paraxiale

On appelle $\vec{K}$ le vecteur de propagation de l'onde dans le domaine $\mathcal{D}$. L'équation eikonale de l'optique géométrique se résume alors à

$$\left|\vec{K}\right|^2 = 1 - N_0$$

où on a noté $\vec{K} = \nabla\phi$ avec $\phi = \sqrt{1-N_0}\,\vec{\alpha}_{in}\,.\,\vec{x}$ où $\vec{\alpha}_{in}$ est le vecteur unitaire de propagation de l'onde laser.

On a alors $\vec{K} = \sqrt{1-N_0}\,\vec{\alpha}_{in}$.

On utilise alors la décomposition (1.12) dans l'équation d'Helmholtz (1.6) et on obtient

$$2i\frac{k_0}{c}\frac{\partial\psi}{\partial t} + \Delta\psi + 2ik_0\nu_0\psi + k_0^2(1 - N_0 - \delta N)\psi = 0.$$

On définit alors $\epsilon = k_0^{-1}$ qui est petit par rapport aux dimensions de variation de la densité électronique et par rapport à la taille des *speckles* (granularités lumineuses d'une taille de l'ordre de 8 λ_0). En multipliant par ϵ, on obtient alors l'équation

$$\boxed{\frac{2i}{c}\frac{\partial\psi}{\partial t} + \epsilon\Delta\psi + 2i\nu_0\,\psi + \epsilon^{-1}\,(1 - N_0 - \delta N)\,\psi = 0.} \tag{1.13}$$

à laquelle on associe la condition limite issue de (1.9) sur le bord Γ^{in} :

$$(\epsilon\frac{\partial}{\partial n} + i\,\vec{K}\cdot n)(\psi - \psi^{in}) = 0. \tag{1.14}$$

Reconsidérons l'approximation W.K.B. (1.11) :

$$\psi(t,x,y) \simeq u(x,y)\,e^{i\,k_0\phi(x,y)}.$$

On rappelle les formules de dérivation :

$$\begin{cases} \dfrac{\partial \psi}{\partial x} = \left[\dfrac{\partial u}{\partial x} + ik_0 u \dfrac{\partial \phi}{\partial x}\right] e^{ik_0\phi}, \\ \dfrac{\partial^2 \psi}{\partial x^2} = \left[\dfrac{\partial^2 u}{\partial x^2} + 2ik_0 \dfrac{\partial \phi}{\partial x}\dfrac{\partial u}{\partial x} + ik_0 u \dfrac{\partial^2 \phi}{\partial x^2} - uk_0^2 \left(\dfrac{\partial \phi}{\partial x}\right)^2\right] e^{ik_0\phi}. \end{cases}$$

On peut ainsi écrire en tenant compte de l'équation eikonale :

$$\Delta\psi = \left[\Delta u + 2ik_0 \vec{K}\cdot\nabla u + ik_0 u \nabla\cdot\vec{K} - uk_0^2(1-N_0)\right] e^{ik_0\phi}$$

ou encore

$$\Delta\psi = \left[\Delta u + \frac{2i\vec{K}}{\epsilon}\nabla u + \frac{iu}{\epsilon}\nabla\cdot\vec{K} - \frac{u}{\epsilon^2}(1-N_0)\right] e^{i\frac{\phi}{\epsilon}}.$$

En reportant cette formule dans l'équation (1.7), on obtient :

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta u + iu\nabla\cdot\vec{K} + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u + \epsilon^{-1}\delta N u = 0.$$

1.3.3 Cas N_0 constant

Le vecteur $\vec{K}$ est constant et tel que $|\vec{K}|^2 = 1 - N_0$. On peut donc écrire :

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta u + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u - \epsilon^{-1}\delta N u = 0. \tag{1.15}$$

On modélise les effets de diffraction par la prise en compte d'un laplacien transverse. En effet, le laplacien (terme de diffusion) de l'équation (1.15) est en facteur du petit paramètre ϵ. Comme dans le cas d'une équation de transport diffusion-convection classique, la convection est donc dominante dans le sens de l'écoulement. Ici, l'écoulement est la propagation du laser dans la direction $\vec{K}$. Ce qui signifie que l'on peut négliger la diffusion dans la direction $\vec{K}$. En d'autres termes, $\vec{K}\cdot\nabla u = 0$. En écrivant le gradient transverse :

$$\nabla_\perp = \nabla - \frac{\vec{K}(\vec{K}\cdot\nabla)}{|\vec{K}|^2}$$

et en le complétant par le gradient classique, on obtient l'équation suivante :

$$\boxed{\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + 2i\vec{K}\cdot\nabla u + 2i\nu u_0 - \epsilon^{-1}\delta N u = 0.} \tag{1.16}$$

Pour la compatibilité du modèle, on choisit l'onde incidente de la forme

$$\psi^{in} = u^{in}\, e^{i\frac{\vec{K}\cdot\vec{x}}{\epsilon}}$$

où u^{in} est l'intensité du laser définie sur le bord.

En insérant le développement (1.11) dans la condition (1.9) et en négligeant $\frac{\vec{K}(\vec{K}\cdot\nabla)}{|\vec{K}|^2}$, on obtient la condition limite pour l'équation (1.16) :

$$(\epsilon\vec{n}\cdot\nabla_\perp + 2i\,\vec{K}\cdot n)(u - u^{in}) = 0. \tag{1.17}$$

Remarque Dans les modèles paraxiaux (1.16) et (1.18), l'onde ne se propage que dans la direction du vecteur $\vec{K}$. Le couplage avec le modèle plus général (1.13) ne se fait donc que par la prise en compte de la condition limite (1.9) dans ce dernier avec

$$\psi^{in} = u^{in}\, e^{i\frac{\vec{K}\cdot\vec{x}}{\epsilon}}$$

où u^{in} est la restriction de la solution u sur le bord entrant.

1.3.4 Cas général

On s'intéresse maintenant au cas où la densité moyenne N_0 est lentement variable. De même que précédemment , on peut négliger $\frac{\vec{K}(\vec{K}\cdot\nabla)}{|\vec{K}|^2}$ devant $\nabla_\perp$ et on peut alors écrire :

$$\boxed{\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + iu\nabla\cdot\vec{K} + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u - \epsilon^{-1}\delta N u = 0.} \tag{1.18}$$

On associe à l'équation (1.18) la condition aux limites (1.17). On rappelle que la direction de propagation du laser est supposée connue. Ainsi, l'application de la méthode d'approximation paraxiale nécessite la résolution de l'équation eikonale afin d'obtenir l'expression du vecteur $\vec{K}$. Si on ne faisait pas d'hypothèse, c'est un problème difficile. Dans notre cas, on peut faire deux types d'hypothèses pour le comportement de la densité. Premièrement, on suppose que $\vec{\alpha}^{in}$ est parallèle à x et que N_0 est seulement fonction de la variable x qui est une direction du parallélépipède. On choisit alors N_0 comme la moyenne de N_e dans la direction transverse à la propagation :

$$N_0(z) = \int \frac{N_e(x, x_\perp)}{L}\, dz_\perp$$

où L est la longueur du laser sur laquelle on fait l'approximation.

Dès lors, l'équation (1.18) se formalise comme suit :

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + iu\frac{\partial K}{\partial x} + 2iK\frac{\partial u}{\partial x} + 2i\nu_0 u - \epsilon^{-1}\delta N u = 0.$$

C'est l'équation classique (voir Berger (1998)).

Un autre cas particulier consiste à considérer seulement que N_0 ne varie que suivant une direction, par exemple x. On montre que la solution de l'équation eikonale est :

$$\begin{cases} K_1(\vec{x}) = (1 - N_0(\vec{x}) - |K_2^{in}|^2)^{1/2}, \\ K_2(\vec{x}) = K_2^{in}. \end{cases}$$

où on a noté le vecteur $\vec{K}^{in} = k_0\sqrt{1-N}e_b$. Comme $\nabla\phi = (1-N_0)\vec{\alpha}^{in}$ sur $\partial\mathcal{D}$, on a donc $K_2^{in} = (1-N_0)\vec{\alpha}_2^{in}$. C'est l'équation dite paraxiale oblique [37].

1.4 Couplage avec le modèle Hydrodynamique

Dans tout ce qui suit, on suppose être dans un plasma quasi-neutre, ce qui se traduit par l'approximation suivante

$$N_e = ZN_I.$$

On redéfinit ainsi la densité électronique adimensionnée

$$N = \frac{Z}{N_e} N_I.$$

On peut obtenir différents modèles en couplant les équations (1.7), (1.6), (1.16) ou encore (1.18) avec un système Euler classique suivant le domaine de validité des équations. Les inconnues sont alors : le champ laser ψ ou u, la densité N et la vitesse ionique U. Ces inconnues satisfont

$$\frac{\partial}{\partial t} N + \nabla(NU) = 0, \tag{1.19}$$

$$\frac{\partial}{\partial t}(NU) + \nabla(NUU) + c_s^2 \nabla N = \mathcal{F}_p \tag{1.20}$$

où on a noté $\mathcal{F}_p$ la force de couplage dite *pondéromotrice*. On prend pour notre modèle

$$\mathcal{F}_p = -N\gamma_p \nabla |\psi|^2$$

où le coefficient $\gamma_p = \frac{Z q_e^2}{4 m_e m_i \omega_0^2}$ est constant et la vitesse du son c_s est également constante ou fonction de N.

1.5 Remarque préliminaire sur la taille mémoire

On s'intéresse à une modélisation en 2D. Notre domaine d'étude réaliste est une boîte d'une longueur de $1500\mu m$ pour une hauteur de $1000\mu m$ avec un profil de densité comprenant un plateau où N est pratiquement constant puis N croît jusqu'à la valeur égale à un près du bord de sortie. On met ici en évidence les contraintes numériques d'une résolution sur un tel domaine.

1.5.1 Zone paraxiale

Le domaine est restreint à un rectangle d'une hauteur de 1000 μm pour une largeur de 1200 μm. Le plasma est sous-dense dans ce domaine. On résout sur ce domaine l'équation Schrödinger paraxiale. On choisit d'appliquer une propagation oblique de l'onde en entrée du domaine avec un angle de faible incidence. On choisit alors un pas régulier de discrétisation $h = \lambda_0$. A titre indicatif, on obtient ainsi un maillage d'environ dix millions de noeuds. En considérant un codage numérique sur huit octets, le vecteur solution obtenu sur ce maillage occuperait environ 160 Mo.

1.5.2 Zone Helmholtz

Le domaine considéré est un rectangle d'une hauteur de 1000 μm pour une largeur de 300 μm contenant la caustique. L'équation paraxiale n'est plus valide. En effet, la densité du plasma varie fortement induisant des changements de direction du faisceau laser. On résout alors l'équation d'Helmholtz. D'une manière classique, on choisit un pas de discrétisation de 10 noeuds par longueur d'onde. On obtient donc un maillage composé d'environ 250 millions de noeuds. Cela représente un encombrement mémoire d'environ 4 Go.

1.6 Stratégie de résolution

Nous centrons par la suite notre travail sur la résolution de l'équation d'Helmholtz (1.7). En effet, les efforts nécessaires à la résolution de ce problème sont énormes en comparaison de ceux requis pour la partie sous-dense. On envisage donc l'utilisation d'un solveur rapide type Réduction Cyclique pour la partie Helmholtz. Evidemment, la rapidité d'un tel solveur nécessite de fortes propriétés algébriques. Ici, la propriété à obtenir est la symétrie de l'opérateur. Les autres propriétés fondamentales sont garanties par le schéma de discrétisation. On présente ici les différents problèmes et la manière de s'adapter.

1.6.1 Couches absorbantes

L'idée générale d'utilisation des couches absorbantes classiques est d'accoler au domaine de propagation une fine couche dans laquelle on génère un mécanisme d'absorption. De manière naturelle, il est possible que ce mécanisme respecte la physique du problème. Par exemple, on rajoute un domaine absorbant par la prise en compte d'un coefficient d'absorption dans le problème étant nul dans le domaine et de profil exponentiel dans les couches d'absorption. L'idée introduite par Berenger [21] pour les couches *PML* est de générer de l'absorption par un phénomène non physique.

Le modèle *PML* peut être obtenu à partir du modèle général

$$-\Delta u - \omega^2 u = f \tag{1.21}$$

en faisant le changement de variable

$$y \to y + \frac{1}{\mathrm{i}\omega}\int_0^y \sigma(\xi) \quad \text{d'où} \quad \frac{\partial}{\partial y} \to \left(1 + \frac{\sigma}{\mathrm{i}\omega}\right)^{-1}\frac{\partial}{\partial y}$$

où σ est une fonction d'amortissement à profil exponentiel traduisant l'effet des couches *PML*. Ceci entraîne le changement dans l'opérateur Laplacien

$$\Delta \to \left(1 + \frac{\sigma}{\mathrm{i}\omega}\right)^{-1}\frac{\partial}{\partial y}\left(1 + \frac{\sigma}{\mathrm{i}\omega}\right)^{-1}\frac{\partial}{\partial y} + \frac{\partial^2}{\partial x^2}.$$

Précisons que σ est nulle hors des couches *PML*. On résout donc le problème modèle hors des couches *PML* et un problème modifié dans les couches *PML*.

Les avantages de cette méthode sont la simplicité d'implémentation et l'absence de réflexion aux interfaces. Toutefois, la perte de propriétés de l'opérateur différentiel empêche l'utilisation de solveur rapide. En effet, en différences finies, les couches *PML* font perdre la séparabilité du problème ou sa symétrie. La décomposition de domaine est donc une alternative pour isoler les perturbations dues aux couches *PML* et pour utiliser en même temps un solveur rapide sur la très grande partie du domaine. Notons $\Omega \subset \mathcal{D}$ le domaine rectangulaire dans lequel l'approximation paraxiale n'est plus valide. Dans notre cas, le faisceau se propage jusqu'à la zone de caustique. Dès lors, le faisceau dévie de sa trajectoire et ressort par le haut ou le bas du domaine. Un fort dépôt d'énergie se fait dans la courbure de la trajectoire. Dans ces conditions, on place les couches *PML* en haut et en bas du domaine. Pour confiner les problèmes engendrés par les couches, une décomposition de domaine est l'outil idéal.

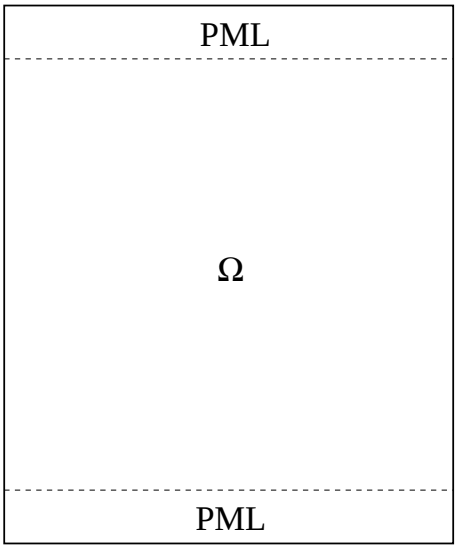


FIG. 1.2 – Domaine global

1.6.2 Solveur Rapide

Il est crucial de remarquer que la décomposition

$$N(x,y) = N_0(x) + \delta N(x,y)$$

permet d'isoler le traitement du système linéaire associé à la discrétisation du problème

$$[\frac{2ik_0}{c\Delta t} + \Delta + 2ik_0\nu_0 + k_0^2(1-N_0)]\psi = f \quad \text{dans} \quad \Omega. \tag{1.22}$$

Or, la matrice associée à ce problème est séparable, c'est-à-dire que l'on peut l'écrire sous la forme

$$A = \begin{pmatrix} B & I & & \\ I & B & I & \\ & I & B & \ddots \\ & & \ddots & \ddots \end{pmatrix}.$$

La matrice B de taille $(n_x \times n_x)$ est à un coefficient additif près associée à la discrétisation de

$$\frac{\partial}{\partial x^2} + 2ik_0\nu_0 + k_0^2(1-N_0).$$

La résolution du système linéaire peut se faire grâce à une technique de réduction cyclique. Notons que l'on ne peut pas intégrer les couches PML ni le terme δN car cela détruirait le caractère séparable du problème. On va utiliser une technique de décomposition de domaine de type Schwarz (ici avec recouvrement) pour résoudre le problème global et au cours de la méthode itérative, on imbrique le traitement du δN. Schématiquement, sans tenir compte des problèmes liés au recouvrement, en notant A_H et A_B les matrices de discrétisation dans les

couches *PML* ; A_I la matrice de discrétisation de la zone interne ; C_1, C_2, C_3 et C_4 les matrices de couplage entre les couches *PML* et la zone interne ; δ la matrice diagonale correspondant au terme δN, on doit résoudre le système du type suivant

$$\begin{pmatrix} A_H & C_1 & 0 \\ C_2 & A_I + \delta & C_3 \\ 0 & C_4 & A_B \end{pmatrix}.$$

La méthode itérative choisie sera une méthode de Krylov type *GMRES*. En effet, ces méthodes sont très bien adaptés au procédé de décomposition de domaine. On obtient notamment de bien meilleurs résultats qu'avec des méthodes classiques type Gauss-Seidel. Ces méthodes sont fortement conditionnées par le spectre de la matrice à inverser. L'expérience montre leur efficacité avec une stratégie de préconditionnement, qui pour nous sera associée à

$$\begin{pmatrix} A_H & & \\ & A_I & \\ & & A_B \end{pmatrix}.$$

1.7 Résumé

Pour résumer, le problème d'interaction laser-plasma revient à faire cohabiter un modèle hydrodynamique avec un ou plusieurs modèles de propagation. Dans un premier temps, on recherche la densité N et la vitesse ionique U en résolvant le système (1.19), (1.20) défini par

$$\frac{\partial}{\partial t} N + \nabla(NU) = 0$$

et

$$\frac{\partial}{\partial t}(NU) + \nabla(NUU) + c_s^2 \nabla N = \mathcal{F}_p.$$

On décompose la densité en posant (1.12)

$$N(x, y) = N_0(x, y) + \delta N(x, y).$$

On peut ensuite coupler plusieurs modèles de propagation laser en fonction de leur domaine de validité. Dans le cadre général, on utilise le modèle (1.13) basé sur l'équation de Schrödinger

$$\frac{2i}{c}\frac{\partial \psi}{\partial t} + \epsilon \Delta \psi + 2i\nu_0 \, \psi + \epsilon^{-1} \left(1 - N_0 - \delta N\right) \psi = 0.$$

On y associe la condition limite entrante (1.9)

$$\left(n.\nabla + ik_0\sqrt{1 - N} e_b.n\right)\left(\psi - \psi^{in}\right) = 0.$$

En sortie de boîte (à droite), on distingue deux cas. Premièrement, si $N < 1$ (cas sous-critique), on associe la condition sortante (1.10)

$$\left(n.\nabla - ik_0\sqrt{1 - N}\right)\psi = 0.$$

Deuxièment, si $N \geq 1$ (cas sur-critique), l'équation devient localement elliptique et une condition de Neuman $\frac{\partial \psi}{\partial n} = 0$ est suffisante.

Dans le cadre d'une propagation paraxiale, on utilise l'approximation (1.11)

$$\psi(\vec{x}) \simeq u(\vec{x})\, e^{i\frac{\vec{K}\cdot\vec{x}}{\epsilon}}.$$

On impose la direction de propagation suivant le vecteur $\vec{K}$. Dans ce cas, la densité doit être constante ou quasi-constante. Dans le cas N_0 constant, on calcule l'enveloppe laser u en utilisant le modèle (1.16)

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + 2i\vec{K}\cdot\nabla u + 2i\nu u_0 - \epsilon^{-1}\delta N u = 0.$$

Dans le cas N_0 quasi-constant, on utilise le modèle paraxial (1.18)

$$\frac{2i}{c}\frac{\partial u}{\partial t} + \epsilon\Delta_\perp u + iu\nabla\cdot\vec{K} + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u - \epsilon^{-1}\delta N u = 0.$$

On y associe la condition limite (1.17)

$$(\epsilon\vec{n}\cdot\nabla_\perp + 2i\,\vec{K}\cdot n)(u - u^{in}) = 0$$

en entrée de boîte (à gauche).

En haut et en bas du domaine, on mettra des couches absorbantes *PML* pour le modèle paraxial [20], [8] et le modèle Helmholtz [21].

2

Discrétisation des équations

Sommaire

On s'intéresse dans cette partie à la discrétisation des différents modèles de propagation et hydrodynamique. A chaque pas de temps $[t^n, t^n + \delta t]$, on doit résoudre successivement le système hydrodynamique (1.19), (1.20), l'équation paraxiale (1.16) ou (1.18) puis l'équation Schrödinger linéaire (ou Helmholtz) (1.13).

Les échelles caractéristiques sont très différentes suivant les modèles. Ceci nous amènera alors à considérer dans une première partie une composition de maillages de pas de discrétisation différents adaptés aux grandeurs. Les schémas utilisés dans le code *HERA* [19] seront ensuite présentés. Dans une dernière partie, la discrétisation des conditions limites pour l'équation (1.13) sera abordée. En particulier, nous mettrons en évidence la nécessité d'une grande précision dans ces conditions afin d'éviter la réflexion des ondes sortantes aux bords. Une étude de la condition de couplage Paraxial-Helmholtz sera également effectuée.

Enfin, nous étudierons d'un point de vue algébrique une condition d'interface type Robin dans le cadre d'une décomposition de domaine.

2.1 Maillages et interpolation

Les longueurs caractéristiques des grandeurs considérées sont très différentes suivant les modèles. Typiquement, un pas de discrétisation suffisant du domaine $\mathcal{D}$ pour le modèle paraxial (1.16) ou (1.18) est de l'ordre de la longueur d'onde car on ne considère que l'enveloppe laser. Par contre, pour le modèle (1.13), on modélise la solution oscillante du phénomène de propagation dans le domaine $\Omega \subset \mathcal{D}$. Un pas de discrétisation satisfaisant est alors une petite fraction longueur d'onde. On considère ainsi deux maillages cartésiens de type différences finies pour les différents modèles. Le premier dit grossier d'un pas d'espace typiquement de l'ordre de la longueur d'onde pour les modèles hydrodynamique et paraxial et un second dit fin d'un pas de l'ordre du dixième de la longueur d'onde pour le modèle Schrödinger linéaire.

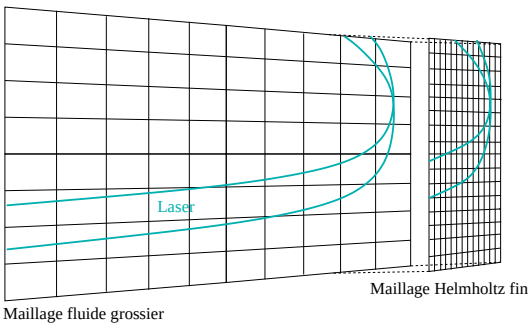


FIG. 2.1 – Maillages fin et grossier

La densité et le champ laser sont définis sur le maillage grossier au centre des mailles. En particulier, la force pondéromotrice du système (1.19) est définie au centre des mailles. Le champ laser sur le maillage fin est défini quant à lui aux nœuds sur le maillage fin. On passe le champ laser du maillage fin au maillage grossier en moyennant. On passe la densité du maillage grossier au maillage fin par interpolation linéaire classique.

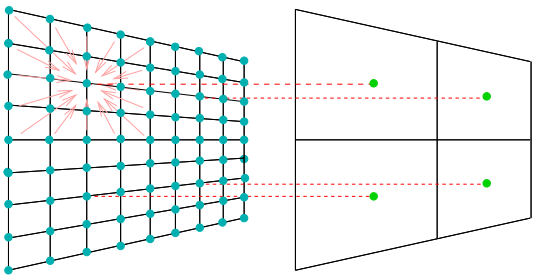


FIG. 2.2 – Maillage fin Helmholtz vers maillage grossier fluide

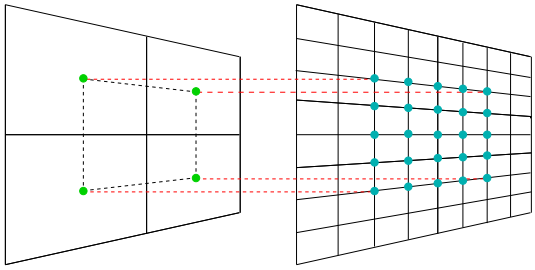


FIG. 2.3 – Maillage grossier fluide vers maillage fin Helmholtz

D'un point de vue pratique, la densité définie sur $\mathcal{D}$ est obtenu par le modèle hydrodynamique sur le maillage grossier. Le champ laser est obtenu par le modèle paraxial sur le domaine $\mathcal{D} \setminus \Omega$ et le modèle Schrödinger-Helmholtz sur le domaine Ω. Pour ce dernier point, il faut passer la densité du maillage grossier au maillage fin. Après le calcul, le champ laser doit être passé sur le maillage grossier fluide pour permettre le calcul de la force pondéromotrice.

Par la suite, on note Δx le pas de discrétisation dans la direction x du maillage grossier et δx la pas du maillage fin. Ces pas sont choisis comme une fraction de la longueur d'onde et au niveau de l'interface, on les fait coïncider. C'est un critère arbitraire qui rend le code plus souple.

2.2 Modèle hydrodynamique

Pour le système hydrodynamique, un schéma explicite de type Euler-Lagrange peut par exemple être utilisé. Dans l'étape Lagrange, on peut utiliser un schéma classique de Godunov pour évaluer les valeurs intermédiaires de la densité et la vitesse. La force pondéromotrice est prise en compte par une discrétisation centrée standard. Après cela, une technique de projection du second ordre avec limiteur de VanLeer pour la partie advective (voir [19]).

2.3 Modèle paraxial

On s'intéresse maintenant à la résolution de l'équation paraxiale (1.18). On choisit y la direction transverse et on note $u_{i,j}^n$ l'évolution de u au temps $t^n = n\delta t$ et aux positions $x_i = i\Delta x$ et $y_j = j\Delta x$. On s'intéresse à une méthode de pas fractionnaires proposée par R. Sentis [3]. S'il y avait uniquement un phénomène propagatif, c'est-à-dire si on avait simplement à traiter l'opérateur d'advection $(\frac{1}{c}\frac{\partial}{\partial t} + K\frac{\partial}{\partial x} + \nu_0)$, on utiliserait un schéma en temps implicite et on aurait les valeurs $u_{i,j}^n$ au temps t^n pas à pas à partir de l'origine $x = 0$. Ici, on effectue un splitting de l'équation (1.18) en fonction de la variable x. On passe de $u_{i,j}^n$ à $u_{i,j}^{n+1}$ en passant par l'intermédiaire $\tilde{u}_{i,j}^n$.

Première étape : l'advection

On traite dans cette étape les termes d'advection, de modification du vecteur de propagation, ainsi que ceux représentant l'absorption du faisceau par le plasma. On suppose connaître les valeurs $u_{i,j}^n$ et on a à résoudre entre x_i et x_{i+1} l'équation d'advection

$$\frac{1}{c}\frac{\partial u}{\partial t} + \vec{K}\cdot\nabla u + \left(\frac{1}{2}\nabla\cdot\vec{K} + \nu_0\right)u = 0.$$

Notons $\mu_{i+1/2} = \frac{1}{2}(\nu_{0,i} + \nu_{0,i+1}) + (K_{i+1} + K_i)/2\Delta x$. On peut calculer $\tilde{u}_{i,j}^n$ par un schéma classique upwind en la variable x

$$\frac{1}{c}\frac{u_{i,j}^n - u_{i,j}^{n-1}}{\delta t} + \vec{K}_i\frac{\tilde{u}_{i+1,j}^n - u_{i,j}^n}{\Delta x} + \frac{\mu_{i+1/2}}{4}(u_{i,j}^n + \tilde{u}_{i+1,j}^n) = 0.$$

Toutefois, l'expérience numérique montre que cette méthode entraîne une perte d'énergie laser. On s'intéresse alors à une méthode indirecte posée sur le module et l'argument de u. On pose $u = |u|.Z$ avec $Z = e^{i\theta}$. On peut alors écrire

$$Z\left[\frac{1}{c}\frac{\partial|u|}{\partial t} + \vec{K}\cdot\nabla|u| + \left(\frac{1}{2}\nabla\cdot\vec{K} + \nu_0\right)|u|\right] + |u|\left[\frac{\partial Z}{\partial t} + \vec{K}\cdot\nabla Z\right] = 0.$$

Puis, on scinde en deux l'équation

$$\begin{cases} \dfrac{1}{c}\dfrac{\partial|u|}{\partial t} + \vec{K}\cdot\nabla|u| + \left(\dfrac{1}{2}\nabla\cdot\vec{K} + \nu_0\right)|u| = 0, \\ \dfrac{\partial Z}{\partial t} + \vec{K}\cdot\nabla Z = 0. \end{cases}$$

Ceci peut aussi se mettre sous la forme

$$\begin{cases} \dfrac{1}{c}\dfrac{\partial|u|^2}{\partial t} + \vec{K}\cdot\nabla|u|^2 + \left(\nabla\cdot\vec{K} + 2\nu_0\right)|u|^2 = 0, \\ \dfrac{\partial Z}{\partial t} + \vec{K}\cdot\nabla Z = 0. \end{cases}$$

Pour la résolution de ce système, on utilise un schéma upwind par rapport à la variable x. On obtient alors

$$\begin{cases} \eta(|u_{i,j}^n|^2 - |u_{i,j}^{n-1}|^2) + (|\tilde{u}_{i+1,j}^n|^2 - |u_{i,j}^n|^2) + \frac{\mu_{i+1/2}}{2K_{i+1/2}}(|\tilde{u}_{i+1,j}^n|^2 - |u_{i,j}^n|^2) = 0 \\ \eta(Z_{i,j}^n - Z_{i,j}^{n-1}) + (\tilde{Z}_{i+1,j}^n - Z_{i,j}^n) = 0 \end{cases}$$

avec $\eta = \frac{\Delta x}{cK\delta t}$ et $\mu_i = \frac{1}{2}\nabla\cdot K_i + \nu_0$. Finalement, on obtient

$$|\tilde{u}_{i+1,j}^n|^2 = \left(|u_{i,j}^n|^2(1 - \eta - \frac{\mu_{i+1/2}\Delta x}{2K_i} + \eta|u_{i,j}^{n-1}|^2\right)\left(1 + \frac{\mu_{i+1/2}\Delta x}{2K_i}\right)^{-1}$$

et pour la partie sur l'argument

$$\tilde{Z}_{i+1,j}^n = (1-\eta)Z_{i,j}^{n+1} + \eta Z_{i,j}^n.$$

Deuxième étape : la diffraction (Cas de l'incidence normale)

On traite ensuite le phénomène de diffraction via l'opérateur Laplacien transverse et le terme de réfraction non-linéaire permettant la prise en compte de l'évolution de la densité. Maintenant que $\tilde{u}^n_{i+1,*}$ est obtenue, il y deux manières d'obtenir $u^n_{i+1,*}$ en résolvant

$$\epsilon\Delta_\perp u + 2i\vec{K}\cdot\nabla u + 2i\nu_0 u - \epsilon^{-1}\delta N u = 0.$$

Une première possibilité est la discrétisation de l'opérateur Laplacien transverse par une technique de Crank-Nicholson

$$\begin{aligned} K\tfrac{u^n_{i+1,j}-\tilde{u}^n_{i+1,j}}{\Delta x} = & \quad \tfrac{i\epsilon}{4}\left((\Delta_\perp \tilde{u}^n_{i+1})_j + (\Delta_\perp u^n_{i+1})_j\right) \\ & - \; c\tfrac{i\epsilon^{-1}}{4}(\delta N)_{i+1/2}\left(\tilde{u}^n_{i+1,j} + u^n_{i+1,j}\right), \end{aligned}$$

où $(\Delta_\perp u^n_{i+1})_j$ est ici la discrétisation de l'opérateur Laplacien transverse

$$\frac{u^n_{j+1} - 2u^n_j + u^n_{j-1}}{\Delta y^2}.$$

On peut d'ailleurs utiliser des approches type couches *PML* sur les bords transverses (voir [20]).

Une deuxième approche consiste à utiliser une formulation analytique pour traiter le terme $\frac{i}{2\epsilon}\delta N$ et une méthode spectrale pour l'opérateur Laplacien transverse. Plus précisément, si on note $\mathcal{T}_\mathcal{F}(u_i)(\xi)$ la transformée de Fourier dans la direction transverse de la fonction $u_i(y)$ et ξ la variable de Fourier correspondant, on pose simplement

$$\hat{u}(\xi) = \exp\left(-i\frac{\xi^2\epsilon}{2K}\Delta x \mathcal{T}_\mathcal{F}(\tilde{u}^n_{i+1})\right)(\xi)$$

et

$$u^n_{i+1} = \exp\left(-i\frac{\epsilon^{-1}}{2K}(\delta N + 2i\nu_0)_{i+1/2}\Delta x\right)\mathcal{T}_\mathcal{F}^{-1}(\hat{u}).$$

D'un point de vue numérique, on utilise la transformée de Fourier rapide (FFT) et inverse pour $\mathcal{T}_\mathcal{F}$ et$\mathcal{T}_\mathcal{F}^{-1}$.

Remarque Dans le cas de l'incidence oblique, ce traitement en Fourier se généralise [36].

2.4 Modèle Helmholtz

On présente maintenant le schéma numérique utilisé pour la résolution de l'équation (1.13). Comme précédemment, l'évolution de ψ au temps $t^n = n\delta t$ et aux positions $x_i = i\delta x$ est notée $\psi^n_{i,j}$ pour $1 \le i \le n_x$ et $y_j = j\delta x$ pour $1 \le j \le n_y$. En utilisant le schéma classique à cinq points, l'équation (1.13) se discrétise comme

$$\begin{aligned} \tfrac{2i}{c}\frac{\psi^n_{i,j}}{\delta t} \; + \; & \epsilon\left[\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\delta x^2} + \frac{\psi^n_{i,j+1} - 2\psi^n_{i,j} + \psi^n_{i,j-1}}{\delta y^2}\right] \\ + \; & 2i\nu_0\,\psi^n_{i,j} + \epsilon^{-1}\left(1 - N_0 - \delta N\right)\psi^n_{i,j} \\ = \; & \tfrac{2i}{c}\frac{\psi^{n-1}_{i,j}}{\delta t}. \end{aligned} \tag{2.1}$$

On connaît l'importance des conditions de sortie dans les problèmes de propagation d'ondes. En effet, les problèmes de réflexion d'ondes en sortie de domaine peuvent dégrader la solution sur tout le domaine. A l'heure actuelle, la référence en la matière est la technique des couches absorbantes *PML* [21]. En pratique, on place ces couches d'une épaisseur de quelques longueurs d'onde dans la trajectoire du faisceau laser pour borner artificiellement le domaine. Dans les couches *PML*, l'opérateur Laplacien se discrétise comme

$$\frac{\psi^n_{i+1,j} - 2\psi^n_{i,j} + \psi^n_{i-1,j}}{\delta x^2} + a_j \frac{a_{j+1/2}(\psi^n_{i,j+1} - \psi^n_{i,j}) - a_{j-1/2}(\psi^n_{i,j} - \psi^n_{i,j-1})}{\delta y^2}$$

où $a_{j+1/2} = \frac{a_{j+1}+a_j}{2}$ avec $a_j = \frac{\mathrm{i}\omega}{\mathrm{i}\omega+\sigma_j}$.

L'impact des couches se traduit par la perte locale de la symétrie de l'opérateur. Dans notre cas, les variations des fluctuations δN font également perdre la séparabilité du problème.

Dans le cas sous-critique ($N < 1$ en sortie de boîte, à droite), il est intéressant d'améliorer la qualité de la condition classiques d'onde sortante (1.10)

$$\left(\epsilon \frac{\partial}{\partial n} - i\sqrt{1-N}\right)\psi = 0.$$

On montrera également l'intérêt d'améliorer la condition entrante (1.9)

$$\left(\epsilon \frac{\partial}{\partial n} + i\sqrt{1-N} e_b.n\right)\left(\psi - \psi^{in}\right) = 0.$$

On présentera tout d'abord comment ces conditions s'obtiennent et comment être plus précis au niveau continu en utilisant la théorie des conditions absorbantes [18]. Au niveau discret, on étudiera en particulier une montée en ordre de la discrétisation de la dérivée normale. Une étude numérique sera effectuée pour comparer les différentes approches entreprises.

Remarque On parle de condition de sortie pour les conditions limites en sortie de boîte à droite. De plus, on place des conditions absorbantes *PML* en sortie de boîte en haut et en bas.

2.4.1 Préliminaires algébriques

On considère le maillage fin Helmholtz régulier du domaine Ω. On numérote les points de discrétisation de manière naturelle qui consiste à numéroter de gauche à droite et de haut en bas.

En posant $\delta N = 0$, la matrice de discrétisation associé au schéma (2.1) s'écrit alors

$$\begin{bmatrix} B_b & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & B_h \end{bmatrix} \tag{2.2}$$

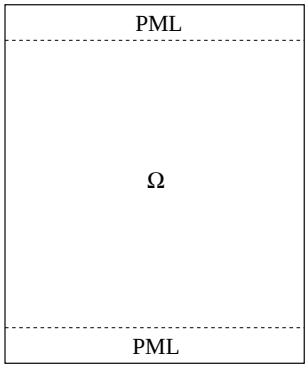


FIG. 2.4 – Domaine

où les matrices B_h et B_b contiennent les couches PML et ne sont pas symétriques.

Sans tenir compte des couches PML, on obtient une matrice carré symétrique (non hermitienne) de dimension $n_x \times n_y$. La matrice $T \in \mathbb{R}^{n_x \times n_x}$ est diagonale et $A \in \mathbb{C}^{n_x \times n_x}$ est tridiagonale et s'écrit

$$A = -\epsilon^2(M_\Delta + M_{CL}) - M_{1-N}$$

où en notant $h = \frac{\delta x}{\delta y}$

$$M_\Delta = \frac{1}{\delta x^2}\begin{bmatrix} -1-2h^2 & 1 & & & & \\ 1 & -2(1+h^2) & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2(1+h^2) & 1 & \\ & & & 1 & -1-2h^2 \end{bmatrix}$$

et

$$M_{1-N} = \begin{bmatrix} 1-N_1 & & & \\ & 1-N_2 & & \\ & & \ddots & \\ & & & 1-N_{n_x} \end{bmatrix}.$$

On a noté M_{CL} la matrice des conditions limites que l'on explicitera par la suite ainsi que la matrice T.

2.4.2 Cas test : la fonction d'Airy

Le cas test fondamental est d'obtenir la fonction connue d'Airy dans le cas où la densité varie linéairement jusqu'à la densité critique. En effet, cette fonction est solution du problème d'Helmholtz en une dimension complétée par une condition d'onde entrante. Dans ce cas, sur la frontière du coté où la densité N est supérieure à 1, on peut prendre la condition au bord $\frac{\partial \psi}{\partial n} = 0$.

Pour mémoire, la fonction d'Airy au carré est

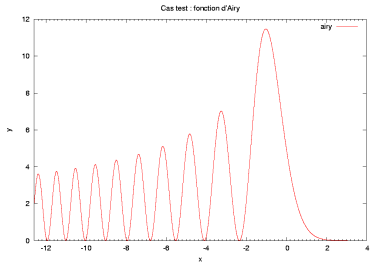


FIG. 2.5 – Fonction d'Airy au carré

L'onde se propage jusqu'à la caustique. A partir de cette frontière, aucune onde électromagnétique ne peut se propager. L'onde se réfléchit donc et se propage dans le sens inverse en déphasée. Une forte énergie apparaît sur la caustique. Une comparaison est effectuée ici entre un cas 1D et une coupe d'un cas 2D. Pour cela, on résout l'équation (1.13) avec la condition entrante (1.9) pour un laser d'une longueur d'onde $\lambda_0 = 0.351 \mu m$ sur un domaine d'une dizaine de longueurs d'onde dans chaque direction avec un critère de discrétisation de 20 points par longueur d'onde.

Nos résultats numériques sont les suivants

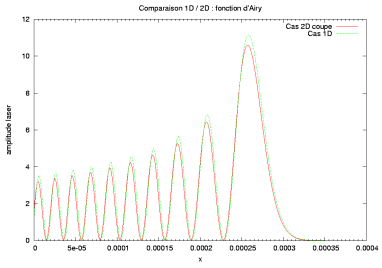


FIG. 2.6 – Coupe d'un cas 2D comparée à la solution 1D

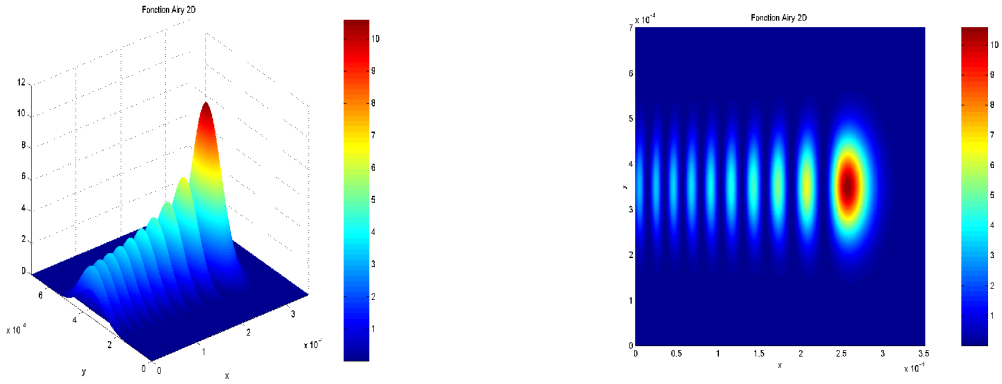


FIG. 2.7 – Cas test 2D

2.5 Condition limite absorbante

On se replace dans le cas où la densité N est toujours strictement inférieure à la densité critique. Un bref rappel est fait sur la théorie des conditions absorbantes [18]. On se place dans le cadre théorique d'étude de la propagation d'une onde dans un domaine qui est tronqué à partir d'une frontière verticale Γ fixée en $x = 0$. On veut alors trouver l'expression d'une condition limite sur Γ qui soit la plus transparente possible pour l'onde traversant la frontière.

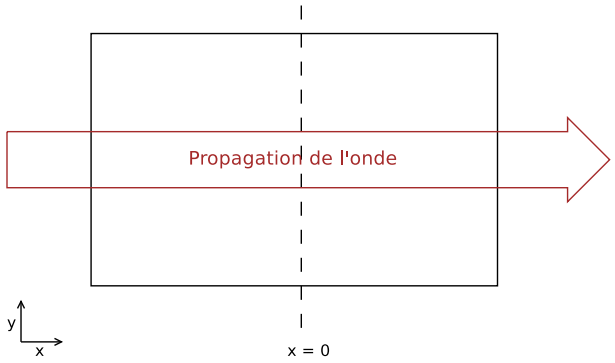


FIG. 2.8 – Limite transparente

On considère l'opérateur d'Helmholtz $\mathcal{L}$ général dans le cas stationnaire et sans absorption défini par

$$\mathcal{L}(u) := \left(-\epsilon^2\Delta - (1-N)\right)(u) = f \quad \text{dans} \quad \Omega. \tag{2.3}$$

On s'intéresse au cas $\mathcal{L}(u) = f$ où le support de f est inclus dans le demi plan $\{x < 0\}$.

On cherche à écrire l'opérateur $\mathcal{L}$ comme une composition d'opérateurs propageant les ondes dans des sens opposés. On écrit alors

$$\mathcal{L} = -\left(\epsilon\frac{\partial}{\partial x} + \Lambda\right)\left(\epsilon\frac{\partial}{\partial x} - \Lambda\right) = -\epsilon^2\frac{\partial^2}{\partial x^2} + \Lambda^2$$

ce qui implique que

$$\Lambda^2 = -\epsilon^2\frac{\partial^2}{\partial y^2} - (1-N).$$

En appliquant une transformée de Fourier en y de variable ξ, on a

$$\mathcal{T}_{\mathcal{F}}(\Lambda^2) = \hat{\Lambda}^2 = -(1-N) + (\epsilon\xi)^2$$

et donc

$$\hat{\Lambda} = \pm\sqrt{-(1-N) + (\epsilon\xi)^2} = \pm \mathrm{i}\sqrt{1-N}\sqrt{1 - \frac{(\epsilon\xi)^2}{(1-N)}}.$$

On choisit

$$\hat{\Lambda} = \mathrm{i}\sqrt{1-N}\sqrt{1 - \frac{(\epsilon\xi)^2}{(1-N)}}.$$

Précisons que $\epsilon\frac{\partial}{\partial x} - \Lambda$ (resp. $\epsilon\frac{\partial}{\partial x} + \Lambda$) est l'opérateur de propagation vers la gauche (resp. droite). Ainsi, on résout

$$-\left(\epsilon\frac{\partial}{\partial x} - \mathcal{T}_{\mathcal{F}}^{-1}\left(\hat{\Lambda}\right)\right)\left(\epsilon\frac{\partial}{\partial x} + \mathcal{T}_{\mathcal{F}}^{-1}\left(\hat{\Lambda}\right)\right)u = 0 \quad \text{dans} \quad \{x \geq 0\}.$$

Si on s'intéresse à la partie $\{x \geq 0\}$, aucune onde ne provient de l'infini. Soit

$$w = \left(\epsilon\frac{\partial}{\partial x} + \mathcal{T}_{\mathcal{F}}^{-1}\left(\hat{\Lambda}\right)\right)u$$

on vérifie donc

$$\begin{cases} \left(\epsilon\frac{\partial}{\partial x} - \mathcal{T}_{\mathcal{F}}^{-1}\left(\hat{\Lambda}\right)\right)w = 0 & \text{dans} \quad \{x \geq 0\} \\ w = 0 & \text{en} \quad +\infty \end{cases}$$

ce qui donne $w = 0$.

La condition limite adaptée est donc

$$\left(\epsilon\frac{\partial}{\partial x} + \mathcal{T}_{\mathcal{F}}^{-1}\left(\mathrm{i}\sqrt{1-N}\sqrt{1 - \frac{(\epsilon\xi)^2}{(1-N)}}\right)\right)u = 0 \quad \text{sur} \quad \Gamma.$$

On l'appelle condition limite absorbante exacte.

Par contre, on remarque que $\hat{\Lambda}$ n'est pas un polynôme en ξ à cause de la racine carré. Ce n'est donc pas le symbole d'un opérateur différentiel local dans le domaine réel. On construit alors différentes approximations du terme $\sqrt{1-(\epsilon\xi)^2/(1-N)}$. En effectuant un développement à l'ordre zéro, on a

$$\sqrt{1 - \frac{(\epsilon\xi)^2}{(1-N)}} \simeq 1.$$

On trouve alors la condition limite classique suivante

$$\left(\epsilon\frac{\partial}{\partial x} + \mathrm{i}\sqrt{1-N}\right)u = 0 \quad \text{sur} \quad \Gamma. \tag{2.4}$$

Avec un développement d'ordre deux, on obtient

$$\sqrt{1 - \frac{(\epsilon\xi)^2}{(1-N)}} \simeq 1 - \frac{(\epsilon\xi)^2}{2(1-N)}.$$

On obtient alors une deuxième condition limite plus précise en continu

$$\left(\epsilon\frac{\partial}{\partial x} + \mathrm{i}\sqrt{1-N} - \frac{\mathrm{i}\epsilon^2}{2\sqrt{1-N}}\frac{\partial^2}{\partial y^2}\right)u = 0. \quad \text{sur} \quad \Gamma \tag{2.5}$$

On dispose maintenant de deux sortes de conditions limites à comparer. Précisons que ces conditions sont adaptées aux ondes planes traversant suivant la normale.

2.6 Discrétisation de la condition limite classique

On s'intéresse ici et dans la suite à la discrétisation de la condition suivante. Que cela soit pour la condition entrante dansle cas où $g \neq 0$ ou sortante dans le cas où $g = 0$ (condition (2.4))

$$\left(\epsilon\frac{\partial}{\partial x} + \mathrm{i}\sqrt{1-N}\right)\psi = g$$

sur une frontière verticale. On se propose ici d'augmenter l'ordre de discrétisation de la dérivée normale pour obtenir un gain de précision. On souhaite ainsi limiter la réflexion des ondes aux bords, y compris pour les ondes arrivant avec un angle d'incidence. Ceci s'applique aux bords sortant mais aussi entrant.

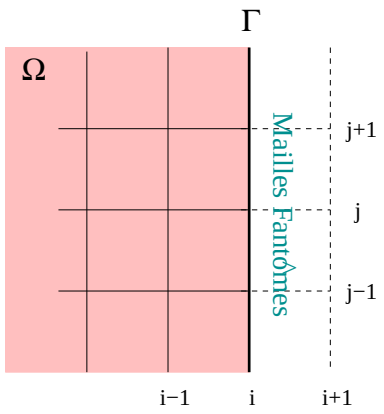


FIG. 2.9 – Bord

On indice par $i+1$ les mailles fantômes du domaine. On rappelle l'écriture du schéma (2.1)

$$-\epsilon^2\left[\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\delta y^2}\right] - (1-N_i)\psi_{i,j} = f_{i,j}.$$

2.6.1 Résultats numériques 1D préliminaires

On présente dans un premier temps un cas 1D très simple permettant de se donner une idée claire du gain engendré par une montée en ordre sur la dérivée normale. Le cas considéré est la propagation d'un faisceau d'un longueur d'onde $\lambda_0 = 0.351\mu m$ dans un plasma sous-dense de densité constante $N = 0.1$ avec condition de reflexion en $x = x_f$.

$$\begin{cases} \epsilon^2\dfrac{\partial^2\psi}{\partial x^2} + (1-N)\psi = 0 \quad \text{sur} \quad [0, x_f], \\ \epsilon\dfrac{\partial\psi}{\partial x} + \mathrm{i}\sqrt{1-N}\psi = g \quad \text{en} \quad x = 0, \\ \dfrac{\partial\psi}{\partial x} = 0 \quad \text{en} \quad x = x_f, \end{cases} \tag{2.6}$$

où $x_f = 10\lambda_0$. On a noté le second membre $g = \epsilon\frac{\partial\psi^{in}}{\Delta x} + \mathrm{i}\sqrt{1-N}\psi^{in}$ où la donnée au bord $\psi^{in} = a^{in}e^{\mathrm{i}\sqrt{1-N}x/\epsilon}$ et $a^{in} = 1$.

La solution attendue est un faisceau dont l'intensité (qui est le carré de la norme de la solution) est constante et égale à la valeur 1. On présente dans les sections suivantes les résultats obtenus pour un cas avec la dérivée normale discrétisée à l'ordre un et un cas avec l'ordre deux. On effectue une première comparaison à dix points par longueur d'onde puis à cinquante points par longueur d'onde.

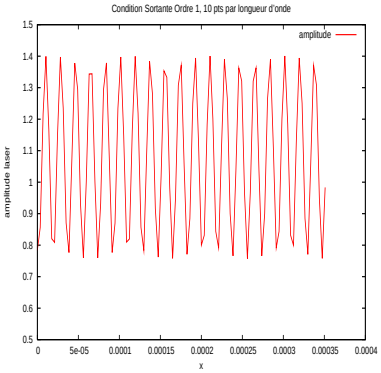


FIG. 2.10 – Ordre un, $\lambda_0/10$

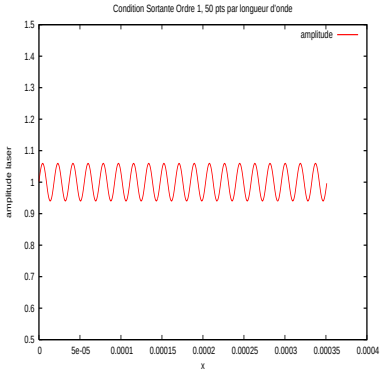


FIG. 2.11 – Ordre un, $\lambda_0/50$

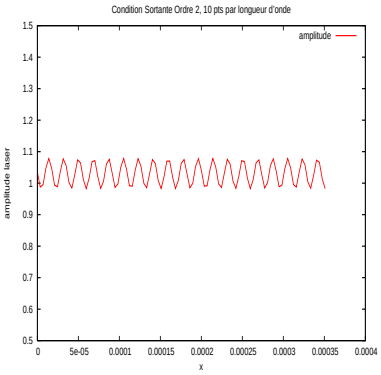


FIG. 2.12 – Ordre deux, $\lambda_0/10$

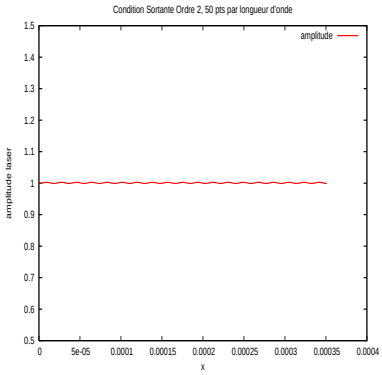


FIG. 2.13 – Ordre deux, $\lambda_0/50$

Toutes les solutions obtenues sont oscillantes. Cela est dû d'une part au caractère oscillant intrinsèque à l'onde et d'autre part à la réflexion de l'onde sur le bord sortant. Plus le nombre de point par longueur d'onde est important, meilleure est l'approximation de l'oscillation mais aussi la qualité de la discrétisation de la dérivée normale.

Le gain de la montée en ordre est flagrant. Avec une discrétisation de la dérivée normale à l'ordre deux et à dix points par longueur d'onde, la précision est meilleure qu'avec une discrétisation à l'ordre un et cinquante point par longueur d'onde. Un résultat intéressant est la réduction significative de l'intensité maximale pour chacun des différents cas en fonction du nombre de points par longueur d'onde et de la montée en ordre.

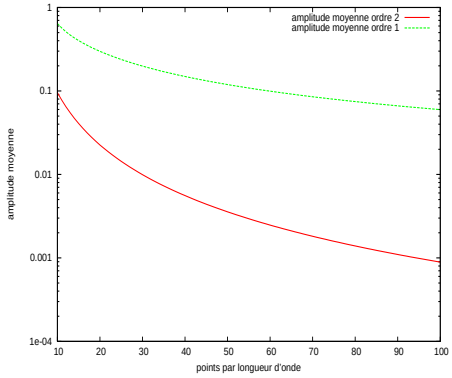


FIG. 2.14 – Amplitude en fonction de la discrétisation

2.6.2 Dérivée normale à l'ordre un

En utilisant l'approximation à l'ordre un (A.5), on discrétise la condition (2.4) comme suit

$$\epsilon \left[\frac{\psi_{i+1,j} - \psi_{i,j}}{\delta x} \right] + \mathrm{i}\sqrt{1 - N_i}\psi_{i,j} = g_{i,j}.$$

On isole alors les points fantômes du maillage

$$\psi_{i+1,j} = \psi_{i,j} + \frac{\delta x}{\epsilon} \left[g_{i,j} - \mathrm{i}\sqrt{1 - N_i}\psi_{i,j} \right]$$

afin de les éliminer dans le schéma (2.1) posé à la frontière, ce qui donne

$$\begin{aligned} - \quad & \epsilon^2 \left[\frac{\psi_{i-1,j} - \psi_{i,j}}{\delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\delta y^2} \right] \\ - \quad & (1 - N_i)\psi_{i,j} + \frac{\mathrm{i}\epsilon}{\delta x}\sqrt{1 - N_i}\psi_{i,j} = f_{i,j} + \frac{\epsilon}{\delta x} g_{i,j} \end{aligned} \tag{2.7}$$

pour $1 \leq j \leq n_y$.

Cette condition, classique pour la résolution de l'équation d'Helmholtz, peut être utilisée en entrée et sortie du domaine. Dans ce cas, on a

$$M_{CL} = -\frac{\mathrm{i}}{\epsilon \delta x} \left(\sqrt{1 - N_1}\delta_1 + \sqrt{1 - N_{n_x}}\delta_{n_x} \right)$$

où on a noté δ_i la matrice de taille $n_x \times n_x$ dont le $i$ème coefficient de la diagonale vaut un et zéro partout ailleurs. On a également

$$T = -\frac{\epsilon^2}{\delta y^2} I$$

où I est la matrice identité de taille $n_x \times n_x$.

2.6.3 Dérivée normale à l'ordre deux

On utilise ici la discrétisation de la dérivée normale à l'ordre supérieur (A.8). La condition limite (2.4) s'écrit

$$\epsilon \left[\frac{\psi_{i+1,j} - \psi_{i,j}}{\delta x} + \frac{\delta x}{2} \left(f_{i,j} + \left. \frac{\partial^2 \psi}{\partial y^2} \right|_{i,j} + \frac{(1 - N_i)}{\epsilon^2} \right) \right] + \mathrm{i}\sqrt{1 - N_i}\psi_{i,j} = g_{i,j}$$

et donc

$$\psi_{i+1,j} = \psi_{i,j} - \frac{\delta x^2}{2}\left(f_{i,j} + \left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} + \frac{(1-N_i)}{\epsilon^2}\right) + \frac{\delta x}{\epsilon}\left[g_{i,j} - \mathrm{i}\sqrt{1-N_i}\psi_{i,j}\right].$$

Après insertion dans (2.1), on a

$$\begin{array}{rcl} - & \epsilon^2\left[\dfrac{\psi_{i-1,j}-\psi_{i,j}}{\delta x^2} + \dfrac{1}{2}\dfrac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{\delta y^2}\right] & \\ - & \dfrac{(1-N_i)}{2}\psi_{i,j} + \dfrac{\mathrm{i}\epsilon}{\delta x}\sqrt{(1-N_i)}\psi_{i,j} = \dfrac{f_{i,j}}{2} + \dfrac{\epsilon}{\delta x}g_{i,j}. & \end{array} \tag{2.8}$$

Remarquons que si on discrétise la condition limite au bord sans l'insérer dans le schéma (2.1), on a alors

$$\left.\frac{\partial\psi}{\partial x}\right|_{i,j} = \frac{\psi_{i,j}-\psi_{i-1,j}}{\delta x} - \frac{\delta x}{2}\left(f_{i,j} + \left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} + \frac{(1-N_i)}{\epsilon^2}\right)$$

et la condition limite se discrétise comme

$$\begin{array}{rcl} \epsilon\dfrac{\psi_{i,j}-\psi_{i-1,j}}{\delta x} & - & \dfrac{\epsilon\delta x}{2}\left[f_{i,j} + \dfrac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{\delta y^2} + \dfrac{(1-N_i)}{\epsilon^2}\psi_{i,j}\right] \\ & + & \dfrac{\mathrm{i}\epsilon}{\delta x}\sqrt{1-N_i}\psi_{i,j} = g_{i,j}. \end{array}$$

En multipliant par $\frac{\epsilon}{\delta x}$, on retrouve (2.8). En effet, la condition limite est d'ordre deux (comme le schéma) et tient compte de la dérivée transverse.

Si on utilise cette condition limite en entrée et en sortie, on a alors

$$M_{CL} = -\frac{\mathrm{i}}{\epsilon\delta x}\left(\sqrt{1-N_1}\delta_1 + \sqrt{1-N_{n_x}}\delta_{n_x}\right) + \frac{1}{\delta y^2}(\delta_1 + \delta_{n_x}) - \frac{1}{2\epsilon^2}M_{1-N}$$

et

$$T = \frac{\epsilon^2}{\delta y^2}\left(I - \frac{\delta_1}{2} - \frac{\delta_{n_x}}{2}\right).$$

2.7 Condition limite grand angle (G.A.)

Au niveau continu, on a vu que la condition limite (2.5) est plus précise que la condition (2.4). En introduisant un second membre, elle s'écrit

$$\left(\epsilon^2\frac{\partial^2}{\partial y^2} - 2(1-N) + 2\mathrm{i}\epsilon\sqrt{1-N}\frac{\partial}{\partial x}\right)\psi = 2\mathrm{i}\sqrt{1-N}g. \tag{2.9}$$

2.7.1 Condition G.A. avec dérivée normale à l'ordre un

Le principe est le même que pour la partie précédente. On discrétise la condition (2.9) en utilisant (A.5).

$$\left.\epsilon^2\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} - 2(1-N_i)\psi_{i,j} + 2\mathrm{i}\epsilon\sqrt{1-N_i}\frac{\psi_{i+1,j}-\psi_{i,j}}{\delta x} = 2\mathrm{i}\sqrt{1-N}g_{i,j}$$

ce qui permet d'écrire

$$\psi_{i+1,j} = \psi_{i,j} + \frac{\delta x}{\epsilon}\left[g_{i,j} - \mathrm{i}\sqrt{1-N_i}\psi_{i,j}\right] - \frac{\mathrm{i}\epsilon\delta x}{2\sqrt{1-N_i}}\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j}.$$

On remarque que c'est équivalent à la condition (2.4) plus un terme de correction en la dérivée seconde de la direction transverse. En insérant ceci dans (2.1), on trouve

$$\begin{aligned} - \quad & \epsilon^2\left[\frac{\psi_{i-1,j}-\psi_{i,j}}{\delta x^2}\right. \\ & \left. + \left(1 - \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_i}}\right)\frac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{\delta y^2}\right] \\ - \quad & (1-N_i)\psi_{i,j} + \frac{\mathrm{i}\epsilon}{\delta x}\sqrt{1-N_i}\psi_{i,j} = f_{i,j} + \frac{\epsilon}{\delta x}g_{i,j}. \end{aligned} \tag{2.10}$$

Si on utilise cette condition limite en entrée et en sortie, on a alors

$$\begin{aligned} M_{CL} = \quad & - \quad \frac{\mathrm{i}}{\epsilon\delta x}\left(\sqrt{1-N_1}\delta_1 + \sqrt{1-N_{n_x}}\delta_{n_x}\right) \\ & - \quad \frac{\mathrm{i}}{2\delta x\delta y^2\epsilon}\left(\frac{\delta_1}{\sqrt{1-N_1}} + \frac{\delta_{n_x}}{\sqrt{1-N_{n_x}}}\right) \end{aligned}$$

et

$$T = -\frac{\epsilon^2}{\delta y^2}\left(I - \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_1}}\delta_1 - \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_{n_x}}}\delta_{n_x}\right).$$

2.7.2 Condition G.A. avec dérivée normale à l'ordre deux

On discrétise maintenant la condition (2.9) en utilisant (A.8).

$$\begin{aligned} & \epsilon^2\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} - 2(1-N_i)\psi_{i,j} \\ + \quad & 2\mathrm{i}\epsilon\sqrt{1-N_i}\left(\frac{\psi_{i+1,j}-\psi_{i,j}}{\delta x} + \frac{\delta x}{2}\left(\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} + \frac{(1-N_i)}{\epsilon^2}\right)\right) = 0 \end{aligned}$$

ce qui permet d'écrire

$$\begin{aligned} \psi_{i+1,j} = \psi_{i,j} \quad & - \quad \frac{\delta x^2}{2}\left(\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j} + \frac{(1-N_i)}{\epsilon^2}\right) - \frac{\delta x}{\epsilon}\mathrm{i}\sqrt{1-N_i}\psi_{i,j} \\ & - \quad \frac{\mathrm{i}\epsilon\delta x}{2\sqrt{1-N_i}}\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,j}. \end{aligned}$$

On remarque que c'est encore équivalent à la condition (2.4) à l'ordre supérieur plus un terme de correction en la dérivée seconde de la direction transverse. En insérant ceci dans (2.1), on trouve

$$\begin{aligned} \epsilon^2\left[\frac{\psi_{i-1,j}-\psi_{i,j}}{\delta x^2} \quad & + \quad \left(\frac{1}{2} - \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_i}}\right)\frac{\psi_{i,j+1}-2\psi_{i,j}+\psi_{i,j-1}}{\delta y^2}\right] \\ & + \quad \frac{(1-N_i)}{2}\psi_{i,j} - \frac{\mathrm{i}\epsilon}{\delta x}\sqrt{1-N_i}\psi_{i,j} = 0. \end{aligned} \tag{2.11}$$

Remarquons qu'en ne discrétisant que la condition limite sans insérer dans le schéma (2.1), on retrouve l'expression (2.11).

Si on utilise cette condition limite en entrée et en sortie, on a alors

$$\begin{aligned} M_{CL} = & - \frac{\mathrm{i}}{\epsilon\delta x}\left(\sqrt{1-N_1}\delta_1 + \sqrt{1-N_{n_x}}\delta_{n_x}\right) \\ & - \frac{\mathrm{i}}{2\delta x\delta y^2\epsilon}\left(\frac{\delta_1}{\sqrt{1-N_1}} + \frac{\delta_{n_x}}{\sqrt{1-N_{n_x}}}\right) + \frac{1}{\delta y^2}(\delta_1 + \delta_{n_x}) \end{aligned}$$

et

$$T = -\frac{\epsilon^2}{\delta y^2}\left(I - \left(\frac{1}{2} + \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_1}}\right)\delta_1 - \left(\frac{1}{2} + \frac{\mathrm{i}\epsilon}{2\delta x\sqrt{1-N_{n_x}}}\right)\delta_{n_x}\right).$$

2.8 Résultats numériques

Une comparaison des différentes discrétisations est ici présentée. On s'intéresse à un cas de propagation dans un domaine rectangulaire $[0, 10\lambda_0] \times [0, 20\lambda_0]$ pour un laser de longueur d'onde $\lambda_0 = 0.351\mu m$ avec le critère de discrétisation de 20 points par longueur d'onde. On se place dans le cas sous-critique et la densité N est choisie constante et égale à 10% de la densité critique. On résout l'équation (1.6) avec les conditions limites classique (2.4) et grand angle (2.5).

Dans un premier temps, on s'intéresse à un cas où la direction de propagation est suivant l'axe x. Dans ce cas, la montée en ordre sur la dérivée normale s'avère être la manipulation permettant un gain flagrant de précision. En effet, on sait que cette condition est transparente pour les ondes se propageant suivant la normale. On s'aperçoit également qu'il n'y a pas de différence notable entre la condition limite classique et la condition limite grand angle.

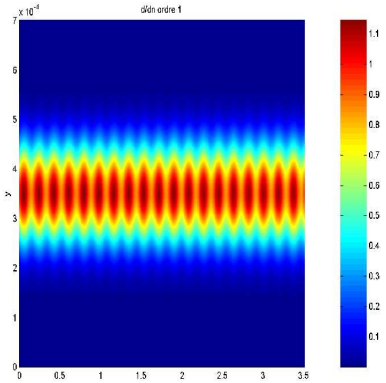


FIG. 2.15 – Condition de sortie classique avec $\frac{\partial}{\partial n}$ à l'ordre 1

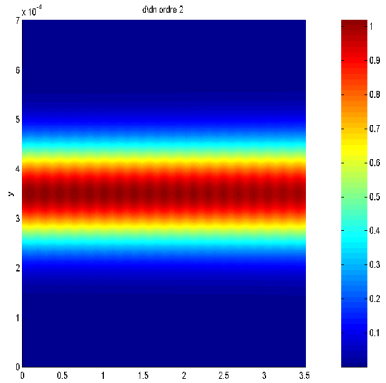


FIG. 2.16 – Condition de sortie classique avec $\frac{\partial}{\partial n}$ à l'ordre 2

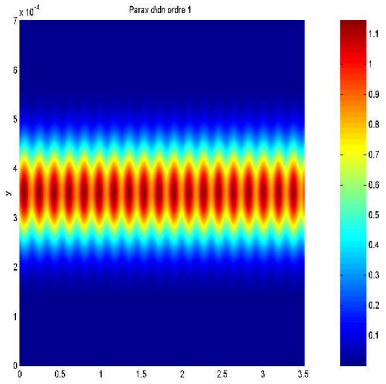


FIG. 2.17 – Condition de sortie grand angle avec $\frac{\partial}{\partial n}$ à l'ordre 1

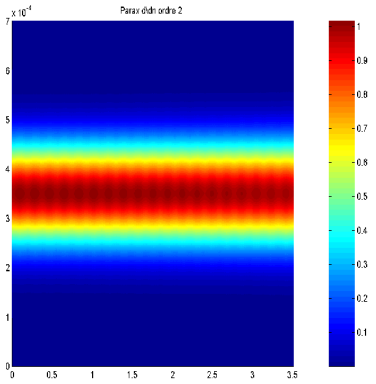


FIG. 2.18 – Condition de sortie grand angle avec $\frac{\partial}{\partial n}$ à l'ordre 2

Dans un second temps, on s'intéresse à un cas de propagation avec un angle d'incidence de 30 degrés.

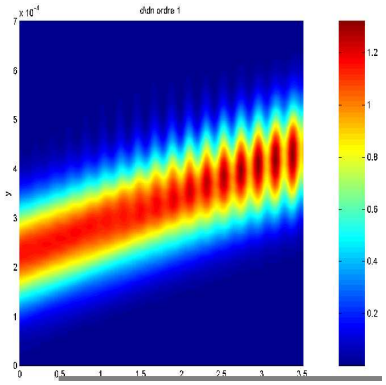


FIG. 2.19 – Condition de sortie classique avec $\frac{\partial}{\partial n}$ à l'ordre 1

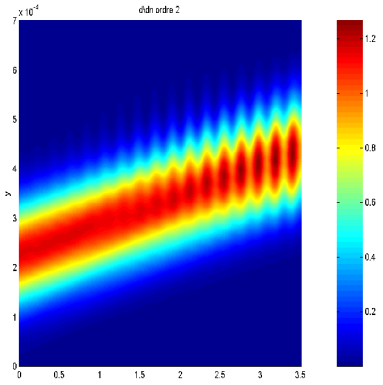


FIG. 2.20 – Condition de sortie classique avec $\frac{\partial}{\partial n}$ à l'ordre 2

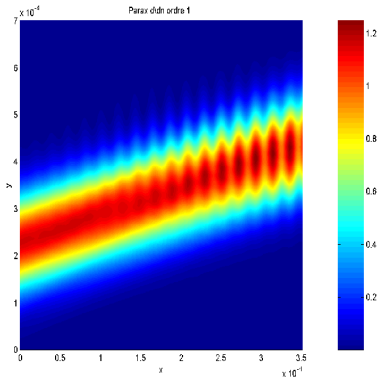


FIG. 2.21 – Condition de sortie grand angle avec $\frac{\partial}{\partial n}$ à l'ordre 1

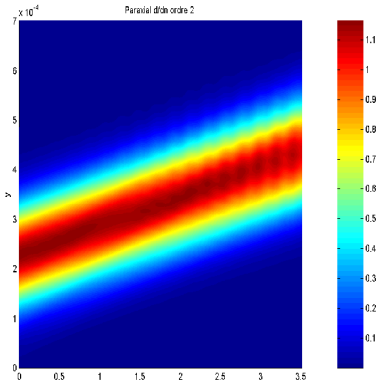


FIG. 2.22 – Condition de sortie grand angle avec $\frac{\partial}{\partial n}$ à l'ordre 2

La condition classique s'avère alors non suffisante et génère beaucoup de réflexions au bord. La différence entre la condition limite classique et la condition

grand angle est par contre significative. Cela est clairement dû à la précision continue supérieure de la condition grand angle engendrée par le terme correctif dans la direction y. Notons également la précision obtenue pour la condition sortante par l'augmentation du nombre de points de discrétisation par longueur d'onde.

2.9 Couplage Paraxial / Helmholtz

On considère la condition entrante (1.9) sur le bord Γ^{in}

$$\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\left(\psi - \psi^{in}\right) = 0$$

où n est la normale sortante. Il est intéressant de noter que la condition (1.9) est une condition sortante du même type que (1.10), à ceci près que l'onde sortante est $\psi - \psi^{in}$. Dès lors, ψ^{in} devient une onde entrante. Nous venons de voir que ces conditions nécessitent une discrétisation particulièrement fine pour être précise et ne pas engendrer de réflexion aux bords. On note

$$\psi^{in} = u^{in} e^{\mathrm{i}\frac{\vec{k}\cdot\vec{x}}{\epsilon}}$$

où on a noté $\vec{k} = \sqrt{1-N}(\cos\alpha, \sin\alpha)^T$ et $\vec{x} = (x, y)^T$. De plus, u^{in} est donnée par le modèle paraxial au bord Γ^{in}. On s'intéresse à la manière de discrétiser le second membre $\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\psi^{in}$ et notamment la dérivée normale de ψ^{in}.

Une première idée est d'utiliser l'écriture analytique de ψ^{in}, ce qui donne

$$\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\psi^{in} = (1+\cos\alpha)\mathrm{i}\sqrt{1-N}\psi^{in}.$$

Une seconde approche consiste à discrétiser la dérivée normale de ψ^{in} par un schéma décentré classique au bord

$$\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\psi^{in} = \left[\xi|_0 + i\sqrt{1-N}\right]\psi^{in}$$

où on a noté

$$\xi|_0 = \frac{e^{\mathrm{i}\sqrt{1-N}\frac{\delta x}{\epsilon}} - 1}{\delta x}.$$

Dans le cas test 1D (2.6), la solution attendue est la valeur 1. Une comparaison significative des deux types de conditions entrantes est de calculer la valeur absolue de la moyenne en fonction du nombre de points par longueur d'onde, c'est-à-dire que l'on évalue

$$a = \frac{1}{x_f}\int_{[0,x_f]} |\psi(x)|^2 dx - 1.$$

On se place dans le cas où on utilise des discrétisations d'ordre deux pour les conditions aux bords.

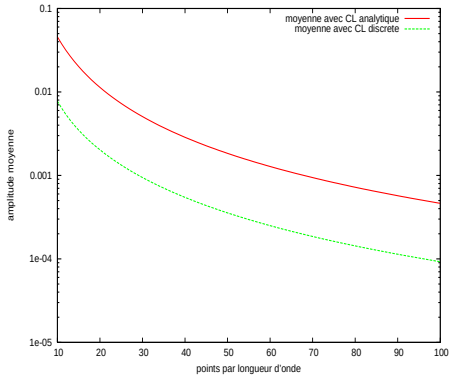


FIG. 2.23 – Moyenne en fonction de la discrétisation

On voit immédiatement le gain apporté par l'utilisation d'une discrétisation pour la dérivée normale de la donnée initiale. Le fait de discrétiser plutôt que d'utiliser la fonction analytique permet une certaine cohérence non intuitive dans le schéma.

2.10 Remarque sur une condition d'interface pour les *PML*

Dans le cadre d'une décomposition de domaine, les conditions d'interface de Robin sont issues de la théorie des conditions absorbantes. On considère une décomposition en deux sous-domaines. L'idée d'une condition d'interface pour les problèmes d'onde est d'être transparente au bord par rapport à la solution du domaine voisin. Une telle condition s'écrit sur l'interface

$$(\frac{\partial}{\partial n} + \alpha)(\psi^1) = (-\frac{\partial}{\partial n} + \alpha)(\psi^2)$$

avec α le paramètre de Robin que l'on peut choisir de différentes manières (par exemple à différents ordres comme dans les sections précédentes).

On recherche ici une écriture de la discrétisation de la condition d'interface compatible avec le schéma utilisé. Or, le schéma (2.1) est d'ordre deux en pas d'espace. Il semble donc naturel d'utiliser une discrétisation également d'ordre deux pour la dérivée normale dans les conditions de Robin. On a d'ailleurs vu le gain de précision engendré par une telle discrétisation. On sait que les conditions de Robin garantissent l'égalité des solutions et des flux à l'interface. On s'intéresse à une discrétisation des dérivées normales des conditions aux interfaces compatible avec le schéma (2.1) notamment pour $\psi^1 = \psi^2$ sur l'interface.

On suppose dans cette partie que la ligne i représente une interface entre deux sous-domaines. On note les solutions ψ^1 et ψ^2 sur les sous-domaines respectifs. On note également n_1 et n_2 les normales à la frontière.

Notons que pour ce cas particulier,

$$\frac{\partial}{\partial n_1} = -\frac{\partial}{\partial n_2} = \frac{\partial}{\partial x}.$$

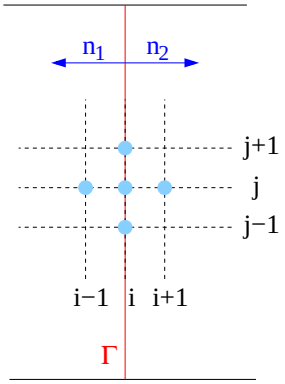


FIG. 2.24 – Schéma et interface

On obtient les approximations suivantes

$$\left.\frac{\partial u}{\partial n_1}\right|_{i,j} = \left.\frac{\partial u}{\partial x}\right|_{i,j} \simeq \frac{u_{i,j} - u_{i-1,j}}{\delta y} + \frac{\delta x}{2} D(u)|_{i,j} \tag{2.12}$$

$$\left.\frac{\partial u}{\partial n_2}\right|_{i,j} = -\left.\frac{\partial u}{\partial x}\right|_{i,j} \simeq -\frac{u_{i+1,j} - u_{i,j}}{\delta x} + \frac{\delta x}{2} D(u)|_{i,j} \tag{2.13}$$

où on a posé

$$D(u)|_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta y^2} + (1 - N_i)u_{i,j} + f_{i,j}.$$

Dès lors, la condition d'interface

$$(\frac{\partial}{\partial n_1} + \alpha)\psi^1 = (-\frac{\partial}{\partial n_2} + \alpha)\psi^2 \tag{2.14}$$

s'écrit

$$\frac{\psi^1_{i,j} - \psi^1_{i-1,j}}{\delta y} + \frac{\delta x}{2} D(\psi^1)\big|_{i,j} + \alpha\psi^1_{i,j} = \frac{\psi^2_{i+1,j} - \psi^2_{i,j}}{\delta y} + \frac{\delta x}{2} D(\psi^2)\big|_{i,j} + \alpha\psi^2_{i,j}.$$

Enfin, pour $\psi^1\big|_{i,j} = \psi^2\big|_{i,j} = \psi|_{i,j}$ pour tout i, on a

$$-\frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\delta y^2} - \frac{\psi^2_{i+1,j} - 2\psi_{i,j} + \psi^1_{i-1,j}}{\delta x^2} - (1 - N_i)\psi_{i,j} = f_{i,j},$$

ce qui coïncide bien avec le schéma (2.1). On a donc une discrétisation de la condition d'interface qui est compatible avec le schéma. Précisons que ce n'est pas la seule. En particulier, en discrétisant la dérivée normale aux ordres supérieurs, on obtient aussi le schéma (2.1) plus un petit terme de correction dû à cette montée en ordre. Par contre, discrétiser la dérivée normale en utilisant un schéma d'ordre un n'amène pas à une bonne compatibilité. Nous n'avons d'ailleurs même pas l'assurance de capter la bonne solution à l'interface.

Par la suite, on considère la décomposition du domaine Ω en trois sous-domaines se recouvrant afin d'isoler les couches absorbantes *PML*.

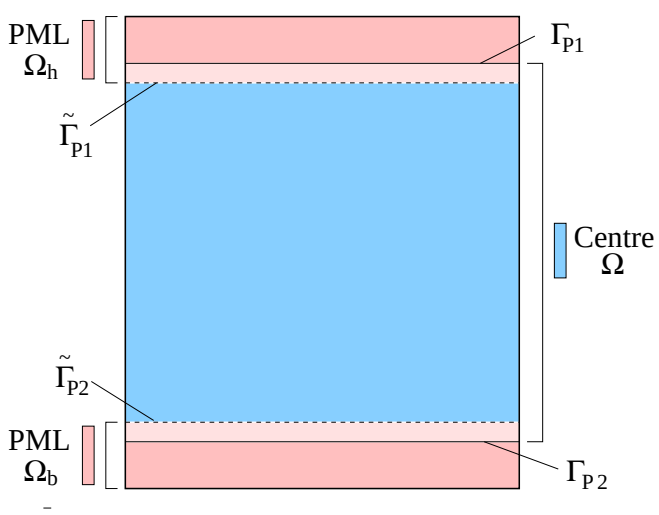


FIG. 2.25 – Décomposition de domaine

Deuxième partie

Résolution des systèmes

3

Algèbre linéaire

Sommaire

Par soucis de performance, nous avons choisi d'utiliser en sortie de domaine les couches absorbantes *PML*. Toutefois, la perte de propriétés intéressantes de l'opérateur localement nous conduit à utiliser une décomposition de domaine. On confine alors les perturbations sur les quelques lignes de maillage des couches *PML*. On rappelle que la déviation macroscopique du laser entraîne le placement de ces couches en haut et en bas du domaine $\Omega \subset \mathcal{D}$ proche de la caustique.

On étudie le problème (2.3) où l'on a découpé Ω en trois bandes horizontales Ω_b, Ω_g et Ω_h. Les domaines Ω_b et Ω_h contiennent les couches *PML*, ce qui représente quelques lignes de maillage. Le domaine Ω_g contient le reste du domaine qui est de très grande taille. On note Γ_{P1} et $\tilde{\Gamma}_{P1}$ (resp. Γ_{P2} et $\tilde{\Gamma}_{P2}$) les interfaces de la zone de recouvrement pour la couche *PML* contenue dans Ω_h (resp. Ω_b). On note ψ_G, ψ_{P1} et ψ_{P2} les solutions dans chacun des 3 sous-domaines Ω_G, Ω_b et Ω_h respectifs.

On impose sur les interfaces une condition de Robin dont l'opérateur est défini comme

$$\mathcal{B}(\psi) = \frac{\partial \psi}{\partial n} + \alpha\psi, \quad \alpha \in \mathbb{C}. \tag{3.1}$$

On résout alors

$$\left\{\begin{array}{rcll}\left(\epsilon^2\tilde{\Delta}+(1-N_0)+i\nu\right)\psi_{P1} & = & 0 & \text{dans } \Omega_b,\\ \epsilon\frac{\partial\psi_{P1}}{\partial n}+i\sqrt{1-N_0}\psi_{P1} & = & 0 & \text{sur } \Gamma^{in}\cap\partial\Omega_b,\\ \frac{\partial\psi_{P1}}{\partial n} & = & 0 & \text{sur } (\partial\Omega\cap\partial\Omega_b)\setminus\Gamma^{in},\\ \mu(y)\frac{\partial\psi_{P1}}{\partial y}+\alpha\,\psi_{P1} & = & \frac{\partial\psi_G}{\partial y}+\alpha\,\psi_G & \text{sur } \tilde{\Gamma}_{P1},\end{array}\right.$$

$$\left\{\begin{array}{rcll}\left(\epsilon^2\Delta+(1-N_0)+i\nu\right)\psi_G & = & \delta N\psi_G & \text{dans } \Omega_g,\\ \epsilon\frac{\partial\psi_G}{\partial n}+i\sqrt{1-N_0}\psi_G & = & g & \text{sur } \Gamma^{in}\cap\partial\Omega_g,\\ \frac{\partial\psi_G}{\partial n} & = & 0 & \text{sur } (\partial\Omega\cap\partial\Omega_g)\setminus\Gamma^{in},\\ \frac{\partial\psi_G}{\partial y}+\alpha\,\psi_G & = & \mu(y)\frac{\partial\psi_{P1}}{\partial y}+\alpha\,\psi_{P1} & \text{sur } \Gamma_{P1},\\ \frac{\partial\psi_G}{\partial y}+\alpha\,\psi_G & = & \mu(y)\frac{\partial\psi_{P2}}{\partial y}+\alpha\,\psi_{P2} & \text{sur } \Gamma_{P2},\end{array}\right. \tag{3.2}$$

$$\left\{\begin{array}{rcll}\left(\epsilon^2\tilde{\Delta}+(1-N_0)+i\nu\right)\psi_{P2} & = & 0 & \text{dans } \Omega_h,\\ \epsilon\frac{\partial\psi_{P2}}{\partial n}+i\sqrt{1-N_0}\psi_{P2} & = & 0 & \text{sur } \Gamma^{in}\cap\partial\Omega_h,\\ \frac{\partial\psi_{P2}}{\partial n} & = & 0 & \text{sur } (\partial\Omega\cap\partial\Omega_h)\setminus\Gamma^{in},\\ \mu(y)\frac{\partial\psi_{P2}}{\partial y}+\alpha\,\psi_{P2} & = & \frac{\partial\psi_G}{\partial y}+\alpha\,\psi_G & \text{sur } \tilde{\Gamma}_{P2},\end{array}\right.$$

où la fonction $\mu(y)$ traduit l'impact des couches *PML* dans l'opérateur

$$\tilde{\Delta}=\mu(y)\frac{\partial}{\partial y}\mu(y)\frac{\partial}{\partial y}+\frac{\partial^2}{\partial x^2}.$$

On a noté α le paramètre de Robin de la condition d'interface et

$$g=\left(\epsilon\frac{\partial}{\partial n}+i\sqrt{1-N_0}\right)\psi_{in}\quad\text{sur}\quad\Gamma^{in}\cap\partial\Omega_g.$$

Remarque B. Després a montré que choisir α comme le paramètre $i\omega$ de l'équation d'Helmholtz amène à de bons résultats de convergence [38]. Pour plus de performances, on peut aussi optimiser α à des ordres élevés [39]. Par contre, un mauvais choix de paramètre, par exemple pour $\alpha=0$ dans le cas sans recouvrement, entraîne la divergence de la méthode de décomposition. Le choix du paramètre de Robin s'avère donc être un élément important pour la convergence de la méthode. On rappelle que dans notre cas, $\omega=\sqrt{1-N_0}$ varie suivant la variable x.

Précisons aussi que la prise en compte d'un recouvrement dans la décompositon de domaine limite largement la perte qualitative engendrée par l'utilisation du paramètre α non optimal. De plus, la condition de Robin aux interfaces est une condition de transmission. Dans notre cas, on peut voir cela comme une condition sortante pour l'onde arrivant en bord du domaine afin d'entrer dans une couche *PML*. Intuitivement, il paraît raisonnable de choisir α constant égal à la valeur de $i\omega$ à l'endroit de sortie du laser sur l'interface. Si N_0 est presque constant, on prend $\alpha=i\sqrt{1-N_{moy}}$, sinon on prend $\alpha=i\sqrt{1-N_\star}$ où $N_\star$ est la densité près de la zone où le laser tourne.

Enfin, rappelons que la décomposition de domaine est un préconditionnement in fine.

Cette décomposition de domaine avec recouvrement s'écrit de manière générale

$$\left\{\begin{array}{rcll} \mathcal{L}(\psi^1) & = & f_1 & \text{dans } \Omega_b, \\ (\frac{\partial}{\partial n_1}+\alpha)(\psi^1) & = & (-\frac{\partial}{\partial n_g}+\alpha)(\psi^g) & \text{sur } \tilde{\Gamma}_{P1}, \end{array}\right.$$

$$\left\{\begin{array}{rcll} \mathcal{L}(\psi^g) & = & f_g & \text{dans } \Omega_g, \\ (\frac{\partial}{\partial n_g}+\alpha)(\psi^g) & = & (-\frac{\partial}{\partial n_1}+\alpha)(\psi^1) & \text{sur } \Gamma_{P1}, \\ (\frac{\partial}{\partial n_g}+\alpha)(\psi^g) & = & (-\frac{\partial}{\partial n_2}+\alpha)(\psi^2) & \text{sur } \Gamma_{P2}, \end{array}\right. \tag{3.3}$$

$$\left\{\begin{array}{rcll} \mathcal{L}(\psi^2) & = & f_2 & \text{dans } \Omega_h, \\ (\frac{\partial}{\partial n_2}+\alpha)(\psi^2) & = & (-\frac{\partial}{\partial n_g}+\alpha)(\psi^g) & \text{sur } \tilde{\Gamma}_{P2}, \end{array}\right.$$

où on a noté $f_\bullet = f|_{\Omega_\bullet}$.

L'écriture matricielle d'une telle décomposition sans et avec recouvrement ainsi que sa résolution par une stratégie de préconditionnement est la motivation de ce chapitre. Dans une première partie, on montrera formellement l'équivalence entre la solution du problème initial et la solution du problème décomposé. Après un rappel sur les techniques itératives de Krylov, on décrira la méthode itérative associée à la résolution du système linéaire de la décomposition de domaine. En particulier, il est possible d'avoir une approche de préconditionnement. Enfin, on présentera une application directe au cas de propagation de faisceau avec couplage à l'hydrodynamique.

Remarque On considèrera dans ce qui suit un problème général stationnaire. L'extension au cas instationnaire est très simple : il suffit de rajouter un terme au second membre du système linéaire à résoudre.

3.1 Ecriture matricielle

On regarde maintenant le problème sous son aspect matriciel. En numérotant les inconnues de gauche à droite et de bas en haut, la matrice sous-jacente à la discrétisation du problème (3.3) en différences finies est symétrique tridiagonale par blocs. On montre ici que la compatibilité de la condition d'interface a aussi un impact matriciel permettant un formalisme très naturel.

3.1.1 Cas sans recouvrement

On note A_{Pi} et A_G les matrices de discrétisation des sous-domaines respectifs Ω_i^p et Ω_G. Les matrices des interfaces sont notées $A_{\Gamma i}$. Enfin, les matrices issues de la connexion entre les sous-domaines et interfaces sont notées $C_{\Gamma i}$ et $C_{i\Gamma}$.

Le problème général peut être alors formalisé de la manière suivante :

$$\begin{pmatrix} A_{P1} & C_{1\Gamma} & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{\Gamma 2} & 0 & 0 \\ 0 & C_{2\Gamma} & A_G & C_{3\Gamma} & 0 \\ 0 & 0 & C_{\Gamma 3} & A_{\Gamma 2} & C_{\Gamma 4} \\ 0 & 0 & 0 & C_{4\Gamma} & A_{P2} \end{pmatrix} \begin{pmatrix} U_{P1} \\ U_{\Gamma 1} \\ U_G \\ U_{\Gamma 2} \\ U_{P2} \end{pmatrix} = \begin{pmatrix} F_{P1} \\ F_{\Gamma 1} \\ F_G \\ F_{\Gamma 2} \\ F_{P2} \end{pmatrix}. \tag{3.4}$$

Le principe matriciel de la décomposition de domaine repose sur le fait de dupliquer les inconnues aux interfaces. On décompose ainsi le vecteur inconnu $U_{\Gamma i}$ en $U^1_{\Gamma i}$ et $U^2_{\Gamma i}$ pour $i = 1, 2$. Pour écrire le système linéaire, on doit également "splitter" les matrices $A_{\Gamma i}$ comme la somme de deux matrices notées $A^1_{\Gamma i}$ et $A^2_{\Gamma i}$. On a vu ci-dessus que l'on peut écrire $A_{\Gamma i} = A^1_{\Gamma i} + A^2_{\Gamma i}$ et que la condition d'interface (2.14) peut s'écrire $(A^1_{\Gamma 1} + \alpha)U^1 = (-A^2_{\Gamma 1} + \alpha)U^2$.

Une forme équivalente à (3.4) s'écrit donc :

$$\begin{pmatrix} A_{P1} & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{\Gamma 2} & 0 & 0 & 0 \\ 0 & 0 & C_{2\Gamma} & C_G & C_{3\Gamma} & 0 & 0 \\ 0 & 0 & 0 & C_{\Gamma 3} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{\Gamma 4} \\ 0 & 0 & 0 & 0 & 0 & C_{4\Gamma} & A_{P2} \\ 0 & -I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -I & I & 0 \end{pmatrix} \begin{pmatrix} U_{P1} \\ U^1_{\Gamma 1} \\ U^2_{\Gamma 1} \\ U_G \\ U^1_{\Gamma 2} \\ U^2_{\Gamma 2} \\ U_{P2} \end{pmatrix} = \begin{pmatrix} F_{P1} \\ F_{\Gamma 1} \\ F_G \\ F_{\Gamma 2} \\ F_{P2} \\ 0 \\ 0 \end{pmatrix}.$$

L'étape suivante consiste à remplacer la relation de continuité $[U_{\Gamma i}] = 0$, $i = 1, 2$ au travers des interfaces par la relation $\alpha[U_{\Gamma i}] = 0$ pour la prise en compte de la condition de Robin. Dès lors, on écrit :

$$\begin{pmatrix} A_{P1} & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{\Gamma 2} & 0 & 0 & 0 \\ 0 & 0 & C_{2\Gamma} & A_G & C_{3\Gamma} & 0 & 0 \\ 0 & 0 & 0 & C_{\Gamma 3} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{\Gamma 4} \\ 0 & 0 & 0 & 0 & 0 & C_{4\Gamma} & A_{P2} \\ 0 & -\alpha & \alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\alpha & \alpha & 0 \end{pmatrix} \begin{pmatrix} U_{P1} \\ U^1_{\Gamma 1} \\ U^2_{\Gamma 1} \\ U_G \\ U^1_{\Gamma 2} \\ U^2_{\Gamma 2} \\ U_{P2} \end{pmatrix} = \begin{pmatrix} F_{P1} \\ F_{\Gamma 1} \\ F_G \\ F_{\Gamma 2} \\ F_{P2} \\ 0 \\ 0 \end{pmatrix}.$$

La dernière étape consiste à faire des combinaisons linéaires pour faire apparaître explicitement la condition de Robin aux interfaces. On additionne la deuxième ligne à la sixième et on soustrait la sixième ligne à la deuxième. De même, on additionne la quatrième ligne à la septième puis on soustraie la quatrième ligne à la septième. On réordonne ensuite pour plus de clarté. On résout alors le système linéaire $A'X' = b'$ où :

$$A' = \left(\begin{array}{cc|ccc|cc} A_{P1} & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} + \alpha & A^2_{\Gamma 1} - \alpha & C_{\Gamma 2} & 0 & 0 & 0 \\ \hline C_{\Gamma 1} & A^1_{\Gamma 1} - \alpha & A^2_{\Gamma 1} + \alpha & C_{\Gamma 2} & 0 & 0 & 0 \\ 0 & 0 & C_{2\Gamma} & A_G & C_{3\Gamma} & 0 & 0 \\ 0 & 0 & 0 & C_{\Gamma 3} & A^1_{\Gamma 2} + \alpha & A^2_{\Gamma 2} - \alpha & C_{\Gamma 4} \\ \hline 0 & 0 & 0 & C_{\Gamma 3} & A^1_{\Gamma 2} - \alpha & A^2_{\Gamma 2} + \alpha & C_{\Gamma 4} \\ 0 & 0 & 0 & 0 & 0 & C_{4\Gamma} & A_{P2} \end{array}\right)$$

$$X' = \begin{pmatrix} U_{P1} & U^1_{\Gamma 1} & U^2_{\Gamma 1} & U_G & U^1_{\Gamma 2} & U^2_{\Gamma 2} & U_{P2} \end{pmatrix}^T$$

et

$$b' = \begin{pmatrix} F_{P1} & F_{\Gamma 1} & F_{\Gamma 1} & F_G & F_{\Gamma 2} & F_{\Gamma 2} & F_{P2} \end{pmatrix}^T$$

3.1.2 Cas avec recouvrement

Le principe de l'écriture du cas avec recouvrement est globalement similaire à celui du cas sans recouvrement. Par contre, on duplique un plus grand nombre d'inconnues, ce qui rend la lecture moins aisée. Par conséquent, on choisit de décrire une décomposition de domaine ne traitant que deux sous-domaines.

Dans le même esprit que précédemment, on appelle les sous-domaines Ω^p (associé à la matrice A_P) et Ω_g. On définit donc le domaine de recouvrement Ω_r et Γ_1, Γ_2 les interfaces entre les différents domaines. Pour ne pas introduire de nouvelles notations, on redéfinit maintenant Ω^p et Ω_g comme étant les domaines non recouverts.

La système linéaire de la discrétisation de l'opérateur $\mathcal{L}$ sur le domaine Ω peut s'écrire ainsi :

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{1R} & 0 & 0 \\ 0 & C_{R1} & A_R & C_{R2} & 0 \\ 0 & 0 & C_{2R} & A_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & C_{\Gamma 2} & A_G \end{pmatrix} \begin{pmatrix} U_P \\ U_{\Gamma 1} \\ U_R \\ U_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ F_{\Gamma 2} \\ F_G \end{pmatrix}.$$

Comme dans le cas sans recouvrement, on va dupliquer les inconnues des interfaces mais aussi du domaine de recouvrement. Pour plus de clarté, on va présenter ces répartitions en deux étapes. La première est de dupliquer les inconnues du domaine de recouvrement. On décompose alors le vecteur U_R en U_R^1 et U_R^2 et on résout alors :

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{1R} & 0 & 0 & 0 \\ 0 & C_{R1} & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & -I & I & 0 & 0 \\ 0 & 0 & 0 & C_{2R} & A_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \end{pmatrix} \begin{pmatrix} U_P \\ U_{\Gamma 1} \\ U_R^1 \\ U_R^2 \\ U_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ 0 \\ F_{\Gamma 2} \\ F_G \end{pmatrix}$$

ou encore

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{1R} & 0 & 0 & 0 \\ 0 & C_{R1} & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & -A_R & A_R & 0 & 0 \\ 0 & 0 & 0 & C_{2R} & A_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \end{pmatrix} \begin{pmatrix} U_P \\ U_{\Gamma 1} \\ U_R^1 \\ U_R^2 \\ U_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ 0 \\ F_{\Gamma 2} \\ F_G \end{pmatrix}$$

et donc

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{1R} & 0 & 0 & 0 \\ 0 & C_{R1} & A_R & 0 & C_{R2} & 0 \\ 0 & C_{R1} & 0 & A_R & C_{R2} & 0 \\ 0 & 0 & 0 & C_{2R} & A_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \end{pmatrix} \begin{pmatrix} U_P \\ U_{\Gamma 1} \\ U_R^1 \\ U_R^2 \\ U_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ F_R \\ F_{\Gamma 2} \\ F_G \end{pmatrix}.$$

On duplique maintenant les inconnues aux interfaces. On décompose $U_{\Gamma i}$ en $U^1_{\Gamma i}$ et $U^2_{\Gamma i}$ pour $i = 1, 2$. Comme précédemment, on effectue un splitting des matrices $A_{\Gamma i}$ comme la somme de deux matrices notées $A^1_{\Gamma i}$ et $A^2_{\Gamma i}$ où $A^1_{\Gamma i} = A^2_{\Gamma i} = \dfrac{A_{\Gamma i}}{2}$. On répartit les doublons d'inconnues à ceux des domaines de recouvrement et on obtient :

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 & & \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{1R} & 0 & 0 & 0 & 0 \\ 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 & 0 \\ 0 & 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & 0 & 0 & C_{2R} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \\ 0 & -I & I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & -I & 0 \end{pmatrix} \begin{pmatrix} U_P \\ U^1_{\Gamma 1} \\ U^2_{\Gamma 1} \\ U^1_R \\ U^2_R \\ U^1_{\Gamma 2} \\ U^2_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ F_R \\ F_{\Gamma 2} \\ F_G \\ 0 \\ 0 \end{pmatrix}.$$

En considérant le terme α de la condition de Robin, on a :

$$\begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{1R} & 0 & 0 & 0 & 0 \\ 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 & 0 \\ 0 & 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & 0 & 0 & C_{2R} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \\ 0 & -\alpha & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha & -\alpha & 0 \end{pmatrix} \begin{pmatrix} U_P \\ U^1_{\Gamma 1} \\ U^2_{\Gamma 1} \\ U^1_R \\ U^2_R \\ U^1_{\Gamma 2} \\ U^2_{\Gamma 2} \\ U_G \end{pmatrix} = \begin{pmatrix} F_P \\ F_{\Gamma 1} \\ F_R \\ F_R \\ F_{\Gamma 2} \\ F_G \\ 0 \\ 0 \end{pmatrix}.$$

On additionne ensuite la deuxième ligne à la septième et la cinquième à la huitième. On obtient la matrice suivante du système linéaire :

$$A' = \begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{1R} & 0 & 0 & 0 & 0 \\ 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 & 0 \\ 0 & 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & 0 & 0 & C_{2R} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \\ C_{\Gamma 1} & A^1_{\Gamma 1} - \alpha & A^2_{\Gamma 1} + \alpha & C_{1R} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{2R} & A^1_{\Gamma 2} + \alpha & A^2_{\Gamma 2} - \alpha & C_{2\Gamma} \end{pmatrix}$$

ainsi que les vecteurs inconnues et second membre :

$$X' = \begin{pmatrix} U_P & U^1_{\Gamma 1} & U^2_{\Gamma 1} & U^1_R & U^2_R & U^1_{\Gamma 2} & U^2_{\Gamma 2} & U_G \end{pmatrix},$$

$$b' = \begin{pmatrix} F_P & F_{\Gamma 1} & F_R & F_R & F_{\Gamma 2} & F_G & F_{\Gamma 1} & F_{\Gamma 2} \end{pmatrix}.$$

D'après les quatrième et cinquième colonnes de la matrice A', on peut voir que chaque paire d'inconnues aux interfaces n'est rattachée qu'à un seul domaine de recouvrement. Or, comme $U^1_R = U^2_R$, on peut choisir d'avoir une contribution

des deux interfaces à chaque domaine de recouvrement. On écrit donc :

$$A'' = \begin{pmatrix} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1} & A^2_{\Gamma 1} & C_{1R} & 0 & 0 & 0 & 0 \\ 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 & 0 \\ 0 & 0 & C_{R1} & 0 & A_R & 0 & C_{R2} & 0 \\ 0 & 0 & 0 & 0 & C_{2R} & A^1_{\Gamma 2} & A^2_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \\ C_{\Gamma 1} & A^1_{\Gamma 1} - \alpha & A^2_{\Gamma 1} + \alpha & 0 & C_{1R} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{2R} & 0 & A^1_{\Gamma 2} + \alpha & A^2_{\Gamma 2} - \alpha & C_{2\Gamma} \end{pmatrix}.$$

On a de plus l'égalité $U^1_{\Gamma i} = U^2_{\Gamma i}$ pour $i = 1, 2$. On ramène alors les contributions des interfaces sans condition de Robin dans chacun des domaines de la décomposition de domaine. En remaniant les inconnues, on obtient finalement l'expression du système linéaire $A'X' = b'$ à inverser dans le cas d'une méthode de décomposition de domaine avec recouvrement pour seulement deux domaines :

$$A''' = \left(\begin{array}{cccc|cccc} A_P & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A_{\Gamma 1} & C_{1R} & 0 & 0 & 0 & 0 & 0 \\ 0 & C_{R1} & A_R & C_{R2} & 0 & 0 & 0 & 0 \\ 0 & 0 & C_{2R} & A^1_{\Gamma 2} + \alpha & 0 & 0 & A^2_{\Gamma 2} - \alpha & C_{2\Gamma} \\ \hline C_{\Gamma 1} & A^1_{\Gamma 1} - \alpha & 0 & 0 & A^2_{\Gamma 1} + \alpha & C_{1R} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{R1} & A_R & C_{R2} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{2R} & A_{\Gamma 2} & C_{2\Gamma} \\ 0 & 0 & 0 & 0 & 0 & 0 & C_{\Gamma 2} & A_G \end{array}\right)$$

et

$$X''' = \begin{pmatrix} U_P & U^1_{\Gamma 1} & U^1_R & U^2_{\Gamma 1} & U^1_{\Gamma 2} & U^2_R & U^2_{\Gamma 2} & U_G \end{pmatrix},$$
$$b''' = \begin{pmatrix} F_P & F_{\Gamma 1} & F_R & F_{\Gamma 2} & F_{\Gamma 1} & F_R & F_{\Gamma 2} & F_G \end{pmatrix}.$$

On a ensuite la possibilité d'utiliser les mêmes méthodes de résolution que pour le cas sans recouvrement.

3.2 Résolution du système linéaire

On peut donc interpréter une décomposition de domaine comme une suite de manipulations algébriques afin de dédoubler des inconnues aux interfaces. On impose également la condition d'interface de la décomposition au moment du split de la matrice des inconnues à l'interface ainsi qu'au moment du choix du paramètre α. Alors, sous réserve d'obtenir un système inversible, on a donc équivalence entre le problème initial et le problème décomposé. On s'intéresse maintenant à la manière de résoudre ces systèmes linéaires engendrés. On présente notamment ici une manière d'utiliser la décomposition de domaine comme préconditionnement.

Pour simplifier les écritures, on se replace dans le cadre d'une décomposition sans recouvrement. C'est exactement pareil pour le cas avec recouvrement. On

note :

$$A_D = \left(\begin{array}{cc|ccc|cc} A_{P1} & C_{1\Gamma} & 0 & 0 & 0 & 0 & 0 \\ C_{\Gamma 1} & A^1_{\Gamma 1}+\alpha & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & A^2_{\Gamma 1}+\alpha & C_{2\Gamma} & 0 & 0 & 0 \\ 0 & 0 & C_{2\Gamma} & A_G & C_{3\Gamma} & 0 & 0 \\ 0 & 0 & 0 & C_{\Gamma 3} & A^1_{\Gamma 2}+\alpha & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & A^2_{\Gamma 2}+\alpha & C_{\Gamma 4} \\ 0 & 0 & 0 & 0 & 0 & C_{4\Gamma} & A_{P2} \end{array}\right)$$

et

$$A_E = \left(\begin{array}{cc|ccc|cc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -A^2_{\Gamma 1}+\alpha & -C_{\Gamma 2} & 0 & 0 & 0 \\ \hline -C_{\Gamma 1} & -A^1_{\Gamma 1}+\alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -A^2_{\Gamma 2}+\alpha & -C_{\Gamma 4} \\ \hline 0 & 0 & 0 & -C_{\Gamma 3} & -A^1_{\Gamma 2}+\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

On a la structure par bloc suivante

$$A_D = \begin{pmatrix} A_H & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_B \end{pmatrix} \quad \text{et} \quad A_E = \begin{pmatrix} 0 & C_1 & 0 \\ C_2 & 0 & C_3 \\ 0 & C_4 & 0 \end{pmatrix} \tag{3.5}$$

où A_D est la matrice composée des discrétisations de chaque domaine et A_E les connections entre sous-domaines.

On cherche alors à résoudre :

$$A_D X' = A_E X' + b'. \tag{3.6}$$

Un vaste choix se présente alors pour la méthode de résolution. On peut par exemple utiliser l'algorithme de Jacobi par blocs et résoudre :

$$A_D (X')^{n+1} = A_E (X')^n + b'.$$

Dans un cadre tout aussi naturel, on peut choisir des variantes de cet algorithme de type Gauss-Seidel ou encore insérer de la sous-relaxation.

On peut aussi choisir de résoudre le problème équivalent :

$$(I - A_D^{-1} A_E) X' = A_D^{-1} b'. \tag{3.7}$$

On remarque que sous cette forme, on résout le problème (3.6) préconditionné par la matrice A_D. Pour la résolution, on utilise généralement des méthodes itératives de Krylov. Elles sont très performantes et adaptées aux problèmes préconditionnés de ce type.

Pour résoudre (3.6) ou (3.7) par un processus itératif, il est donc nécessaire de résoudre un système linéaire correspondant à A_D à chaque itération. En

examinant sa structure, on constate que A_D est composée de trois blocs indépendants caractérisant les trois domaines de la décomposition. L'inversion de A_D à chaque itération revient à inverser ces trois blocs indépendemment.

On peut aussi choisir d'utiliser un solveur approché pour l'inversion de A_D. En particulier, on utilise un solveur rapide pour l'inversion du bloc central A_I. Dans ce cas, on résout alors

$$(I - \tilde{A}_D^{-1} A_E) X' = \tilde{A}_D^{-1} b',$$

où on a noté

$$\tilde{A}_D = \begin{pmatrix} A_H & 0 & 0 \\ 0 & \tilde{A}_I & 0 \\ 0 & 0 & A_B \end{pmatrix}.$$

Ici, $\tilde{A}_I$ est la matrice approchée de A_I.

3.3 Résolution par méthode de Krylov

Soient $\mathcal{P}_\mathcal{K}$ le projecteur orthogonal sur $\mathcal{K}$ et $\mathcal{Q}_\mathcal{K}^\mathcal{L}$ le projecteur (oblique) sur $\mathcal{K}$ orthogonalement à $\mathcal{L}$. Illustrés sur la figure (3.1), on rappelle que ces projecteurs sont définis par

$$\begin{aligned} \mathcal{P}_\mathcal{K} x \in \mathcal{K}, &\quad x - \mathcal{P}_\mathcal{K} x \perp \mathcal{K}, \\ \mathcal{Q}_\mathcal{K}^\mathcal{L} x \in \mathcal{K}, &\quad x - \mathcal{Q}_\mathcal{K}^\mathcal{L} x \perp \mathcal{K}. \end{aligned}$$

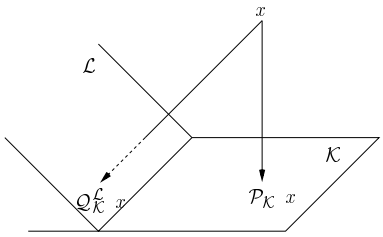


FIG. 3.1 – Projection orthogonale et oblique

Les méthodes de Krylov sont basées sur des processus de projection orthogonale ou oblique sur des sous-espaces dits de Krylov. Une méthode générale de projection pour résoudre le système linéaire $n \times n$

$$Mx = b$$

est une méthode qui donne une approximation x_m de la solution provenant d'un sous-espace affine $x_0 + \mathcal{K}_m$ de dimension m avec une contrainte de Petrov-Galerkin

$$b - Mx_m \perp \mathcal{L}_m$$

où $\mathcal{L}_m$ est un (autre) sous-espace de dimension m. Ici, x_0 est la donnée arbitraire initiale.

Une méthode de Krylov est une méthode pour laquelle $\mathcal{K}_m$ est le sous-espace de Krylov

$$\mathcal{K}_m(M, r_0) = \text{span}\{r_0, Mr_0, \ldots, M^{m-1}r_0\}$$

où $r_0 = b - Mx_0$.

Il est clair que les approximations obtenues par les méthodes de Krylov sont de la forme

$$M^{-1}b \approx x_m = x_0 + q_{m-1}(M)r_0$$

où q_{m-1} est un polynôme de degré $m-1$.

La façon de choisir les contraintes pour la construction des approximations polynomiales, c'est-à-dire le choix de $\mathcal{L}_m$, a un effet important sur la technique itérative. On présente dans la suite deux choix pour $\mathcal{L}_m$ amenant à des techniques bien connues.

3.3.1 Méthode *GMRES*

On définit ainsi la classe de méthode type variation minimum du résidu en considérant la contrainte $\mathcal{L}_m = M\mathcal{K}_m$. La méthode du *RESidu Minimum Généralisé* (*GMRES* [40]) est une méthode de projection de ce type.

On utilise tout d'abord la méthode d'Arnoldi pour créer une base orthonormée de $\mathcal{K}_m$. On note V_m la matrice $n \times m$ constituée des vecteurs colonnes $v_1, \ldots, v_m$ de la base. On sait alors qu'il existe H_m une matrice d'Hessenberg de dimension $(m+1) \times m$ telle que

$$MV_m = V_{m+1}H_m.$$

On pose $v_1 = r_0/\beta$. Ainsi, tout vecteur x dans $x_0 + \mathcal{K}_m$ peut s'écrire

$$x = x_0 + V_m y$$

où y est un m-vecteur. On définit

$$J(y) = ||b - Mx||_2 = ||b - M(x_0 + V_m y)||_2$$

Or, on a

$$\begin{aligned} b - Mx &= b - M(x_0 + V_m y) \\ &= r_0 - MV_m y \\ &= \beta v_1 - V_{m+1}H_m y \\ &= V_{m+1}(\beta e_1 - H_m y). \end{aligned}$$

Comme les vecteurs colonnes de V_{m+1} sont orthonormés, alors

$$J(y) \equiv ||b - M(x_0 + V_m y)||_2 = ||\beta e_1 - H_m y||_2. \tag{3.8}$$

L'approximation donnée par *GMRES* est l'unique vecteur de $x_0 + \mathcal{K}_m$ qui minimise (3.3.1). On l'obtient simplement par

$$x_m = x_0 + V_m y_m$$

où $y_m = \text{argmin}_y ||\beta e_1 - H_m y||_2$.

Algorithme 1 Algorithme *GMRES*

Entrée: $x^{(0)}$, ϵ
Sortie: $x = M^{-1}b$
  $r = b - Mx^{(0)}$
  $v^{(1)} = \dfrac{r}{\|r\|_2}$
  $z_1 = \|r\|_2$
  pour $i = 1, 2, \ldots$ **faire**
    $\omega = Mv^{(i)}$
    pour $k = 1, \ldots, i$ **faire**
      $h_{k,i} = \left(\omega, v^{(i)}\right)$
      $\omega = \omega - h_{k,i}v^{(k)}$
    fin pour
    $h_{i+1,i} = \|\omega\|_2$, $v^{(i+1)} = \omega / h_{i+1,i}$
    pour $k = 1, \ldots, i$ **faire**
      $\begin{pmatrix} h_{k,i} \\ h_{k+1,i} \end{pmatrix} = \begin{pmatrix} \overline{c_k} & s_k \\ -s_k & c_k \end{pmatrix} \times \begin{pmatrix} h_{k,i} \\ h_{k+1,i} \end{pmatrix}$
    fin pour
    $\alpha = \sqrt{|h_{i,i}|^2 + |h_{i+1,i}|^2}$, $s_i = \dfrac{h_{i+1,i}}{\alpha}$, $c_i = \dfrac{h_{i,i}}{\alpha}$
    $z_{i+1} = -s_i z_i$, $z_i = \overline{c_i} z_i$, $h_{i,i} = \overline{c_i} h_{i,i} + s_i h_{i+1,i}$
    si $|z_{i+1}| < \epsilon$ **alors**
      $\begin{pmatrix} y_1 \\ \vdots \\ y_i \end{pmatrix} = \begin{pmatrix} h_{1,1} & \ldots & h_{1,i} \\ \vdots & \ddots & \vdots \\ 0 & \ldots & h_{i,i} \end{pmatrix}^{-1} \begin{pmatrix} z_1 \\ \vdots \\ z_i \end{pmatrix}$
      $x = x^{(0)} + \sum_{k=1}^{i} y_k v^{(k)}$
      retourner x
    fin si
  fin pour

Le minimiseur y_m est d'un très faible coût à calculer puisque c'est la solution d'un problème aux moindres carrés d'une taille $(m+1)\times m$ où m est typiquement petit. De plus, en utilisant les rotations de Givens, trouver y_m peut se ramener à résoudre un système linéaire triangulaire supérieur de dimension $m \times m$. Il est intéressant de remarquer que d'un point de vue pratique, on doit stocker toutes les directions de descente v_i calculées au cours de l'algorithme. Ceci peut s'avérer rapidement très coûteux suivant la taille des problèmes à inverser et la vitesse de convergence. C'est l'inconvénient majeure de la méthode.

3.3.2 Méthode *BiCG* et variantes

L'algorithme du *Gradient Biconjugué* (*BICG* [41]) est une procédure de projection de $\mathcal{K}_m(M, v_1)$ orthogonalement à l'espace des contraintes

$$\mathcal{L}_m = \mathcal{K}_m(M^T, w_1) = \text{span}\{w_1, M^T w_1, \ldots, (M^T)^{m-1} w_1\}$$

en prenant $v_1 = r_0/||r_0||_2$. Le vecteur w_1 est choisi arbitrairement tel que $(v_1, w_1) \neq 0$ et on le choisit souvent égal à v_1. S'il y a un système dual $A^T x^* = b^*$ à résoudre avec A^T, alors w_1 est obtenu en décalant le résidu initial $b^* - A^T x_0^*$. On utilise le procédé de biorthogonalisation de Lanczos construisant une paire de bases orthogonales V_m et W_m des espaces respectifs $\mathcal{K}_m$ et $\mathcal{L}_m$. Il existe alors une matrice tridiagonale T_m vérifiant

$$W_m^T A V_m = T_m$$

et la solution approchée est

$$x_m = x_0 + V_m y_m, \quad y_m = T_m^{-1}(\beta e_1).$$

La matrice T_m est la projection de A obtenue à partir de la projection oblique de $\mathcal{K}_m(M, v_1)$ orthogonalement à $\mathcal{K}_m(M^T, w_1)$. De manière similaire, T_m^T est la projection de A^T obtenue à partir de la projection oblique de $\mathcal{K}_m(M^T, w_1)$ orthogonalement à $\mathcal{K}_m(M, v_1)$. Les opérateurs A et A^T joue un rôle dual à cause des opérations similaires effectuées sur eux. En fait, deux systèmes linéaires sont résolus implicitement, un avec A et l'autre avec A^T. Les opérations sur A^T sont alors essentiellement du gaspillage. D'un point de vue pratique, l'algorithme de Lanczos a un avantage significatif sur l'algorithme d'Arnoldi car il requière simplement le stockage de quelques vecteurs car il n'y a pas de réorthogonalisation.

On écrit la décomposition LU de T_m comme

$$T_m = L_m U_m$$

et on définit

$$P_m = V_m U_m^{-1}.$$

De même, on définit

$$P_m^* = W_m L_m^{-1}.$$

Clairement, les vecteurs p_i^* de P_m^* et les vecteurs p_i de P_m sont A-conjugués puisque

$$(P_m^*)^T A P_m^* = L_m^{-1} W_m^T A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I. \tag{3.9}$$

La solution est alors exprimée comme

$$\begin{aligned} x_m &= x_0 + V_m T_m^{-1}(\beta e_1) \\ &= x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) \\ &= x_0 + P_m L_m^{-1}(\beta e_1). \end{aligned}$$

On dérive maintenant l'algorithme. Le vecteur x_{j+1} peut être exprimé comme

$$x_{j+1} = x_j + \alpha_j p_j.$$

De plus, les vecteurs résidus doivent satisfaire la récurrence

$$r_{j+1} = r_j - \alpha_j A p_j \quad , \quad r_{j+1}^* = r_j^* - \alpha_j^* A^T p_j^*.$$

Il est bien connu que la prochaine direction de descente p_{j+1} est une combinaison linéaire de r_{j+1} et p_j, et il suit que

$$p_{j+1} = r_{j+1} - \beta_j p_j \quad , \quad p_{j+1}^* = r_{j+1}^* - \beta_j^* p_j^*.$$

Comme les résidus sont orthogonaux et d'après (3.9), on montre que

$$\alpha_j = \alpha_j^* = \frac{(r_j, r_j^*)}{(Ap_j, p_j^*)} \quad , \quad \beta_j = \beta_j^* = \frac{(r_{j+1}, r_{j+1}^*)}{(r_j, r_j^*)}.$$

On effectue l'algorithme appelée *Conjugate Gradient Squared* (CGS [42]) suivant

Algorithme 2 Algorithme CGS

Entrée: x_0, ϵ
Sortie: $x = M^{-1}b$

$r_0 = b - Mx_0$
$r_0^* = r_0$, $p_0 = r_0$, $p_0^* = r_0^*$
pour $k = 1, 2, \ldots$ convergence, **faire**
  $\alpha_j = \dfrac{(r_j, r_j^*)}{(Ap_j, p_j^*)}$
  $x_{j+1} = x_j + \alpha_j p_j$, $r_{j+1} = r_j - \alpha_j Ap_j$
  si $||r_{j+1}|| < \epsilon$ **alors**
    retourner x
  fin si
  $r_{j+1}^* = r_j^* - \alpha_j^* A^T p_j^*$
  $\beta_j = \dfrac{(r_{j+1}, r_{j+1}^*)}{(r_j, r_j^*)}$
  $p_{j+1} = r_{j+1} - \beta_j p_j$, $p_{j+1}^* = r_{j+1}^* - \beta_j^* p_j^*$
fin pour

Une variante de l'algorithme CGS permet de se passer de la transposée et d'accélérer la convergence. L'idée principale est basée sur l'observation suivante. Dans l'algorithme BCG, le vecteur résidu r_j et la direction de descente p_j peuvent être exprimés comme

$$r_j = \phi_j(A) r_0 \quad , \quad p_j = \pi_j(A) r_0$$

où ϕ et π sont des polynômes de degré j. D'après l'algorithme, les directions r_j^* et p_j^* sont définies au travers des mêmes récurrences et donc

$$r_j^* = \phi_j(A^T) r_0^* \quad , \quad p_j^* = \pi_j(A^T) r_0^*.$$

On note aussi que le scalaire α_j dans BCG est donné par

$$\alpha_j = \frac{(\phi_j(A) r_0, \phi_j(A^T) r_0^*)}{(A\pi_j(A) r_0, \pi_j(A^T) r_0^*)} = \frac{(\phi_j^2(A) r_0, r_0^*)}{(A\pi_j^2(A) r_0, r_0^*)},$$

ce qui indique qu'il est possible d'obtenir des récurrences pour les vecteurs $\phi_j^2(A) r_0$ et $\pi_j^2(A) r_0$, alors calculer α_j et similairement β_j ne pose pas de problème. Ainsi, l'idée est de chercher une séquence d'itérés dont les normes des résidus satisfont $r_j' = \pi_j^2(A) r_0$. Dans cet algorithme, on ne fait plus de produit matrice-vecteur avec la transposée. Toutefois, comme les polynômes sont au

carré, les erreurs d'arrondis ont tendance à causer plus de dommage que dans l'algorithme BCG.

Ce constat a été la motivation de recherche d'un algorithme plus robuste sur la base de CGS. Une version stabilisée, *Biconjugate Gradient Stabilized* (*BICGStab* [43]), produit des itérés dont les vecteurs résidus sont de la forme $r'_j = \phi_j(A)\psi_j(A)r_0$ dans laquelle ψ_j est le polynôme du résidu associé à l'algorithme BCG et ϕ_j est un nouveau polynôme défini récursivement à chaque pas dont le but est de régulariser le comportement de convergence de l'algorithme original.

On effectue l'algorithme suivant

Algorithme 3 Algorithme *BICGStab*

Entrée: x_0, ϵ
Sortie: $x = M^{-1}b$
  $r_0 = b - Mx_0$
  $r_0^* = r_0$, $p_0 = r_0$
  pour $k = 1, 2, \ldots$ convergence, **faire**
    $\alpha_j = \dfrac{(r_j, r_0^*)}{(Ap_j, r_0^*)}$
    $s_j = r_j - \alpha_j Ap_j$
    $w_j = \dfrac{(As_j, s_j)}{(As_j, As_j)}$
    $x_{j+1} = x_j + \alpha_j p_j + w_j s_j$
    $r_{j+1} = s_j - w_j As_j$
    si $||r_{j+1}|| < \epsilon$ **alors**
      retourner x
    fin si
    $\beta_j = \dfrac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \dfrac{\alpha_j}{w_j}$
    $p_{j+1} = r_{j+1} - \beta_j(p_j - w_j Ap_j)$
  fin pour

3.4 Application au système linéaire matriciel

Les fluctuations $\delta N(x, y)$ ne permettent pas d'utiliser un solveur rapide. En effet, le problème n'est alors pas séparable. Dans des méthodes comme la Réduction Cyclique, c'est une propriétés cruciale. Toutefois, il est possible d'intégrer les fluctuations à la méthode itérative. Le nombre d'itérations de la méthode va alors être en rapport avec la décomposition de domaine mais aussi avec le creusement de la densité par rapport à la densité moyenne N_0.

Dans le cas général où on prend en compte δN, la méthode de décomposition de domaine s'adapte de la façon suivante. On va s'intéresser à plusieurs méthodes de résolution, plus ou moins applicables à des tailles de géométries réelles, mais traduisant un intérêt comparatif.

On veut résoudre l'équation (1.7) :

$$-\epsilon^2 \Delta\psi - (1 - N_0)\psi + i\nu\psi = \delta N\psi \quad \text{dans} \quad \Omega.$$

Une idée naturelle pour la résolution d'un grand système linéaire est la recherche d'un préconditionneur performant. Il faut bien entendu que le préconditionneur soit très rapidement inversible pour être d'un certain intérêt. Notons que dans notre cas, le problème suivant

$$-\epsilon^2 \Delta u - (1 - N_0(x))u + i\nu u = 0 \quad \text{dans} \quad \Omega \tag{3.10}$$

est un excellent préconditionneur du problème (1.7).

On appelle A la matrice sous-jacente au problème (1.7) dont l'expression matricielle peut être formalisée comme (3.4).

On note $\delta = I_* \, \delta N$ où le vecteur δN est défini sur le maillage et où I_* est une matrice diagonale valant 1 pour les inconnues du domaine intérieur non recouvert et 0 sinon.

Donner une solution à (1.7) revient à résoudre le problème matriciel suivant :

$$A_D X' + \delta X' = A_E X' + b'. \tag{3.11}$$

On peut alors résoudre le problème complet (3.11) par une méthode de Krylov :

$$(I + A_D^{-1}\delta - A_D^{-1}A_E)X' = A_D^{-1}b'.$$

A chaque itération, on doit inverser le système associé à la matrice A_D. Notons que l'inversion du bloc interne A_G de A_D peut se faire par un solveur rapide.

Remarque En considérant le même problème (3.11), on peut aussi choisir de résoudre :

$$(I + (A_D + \delta)^{-1}A_E)X' = (A_D + \delta)^{-1}b'$$

A chaque itération de la méthode de Krylov, on inverse le système associé à la matrice $A_D+\delta$. Le plus coûteux est l'inversion du bloc interne $A_G+\delta$. D'après les sections précédentes, on connaît un préconditionneur efficace de cette matrice. On peut donc par exemple appliquer un principe de méthode itérative imbriquée (ou emboîtée) en insérant une deuxième méthode de Krylov pour résoudre les systèmes linéaires du type $A_G^{-1}(A_G + \delta)y = c$ en interne de la première. Le nombre d'itérations a priori de cette méthode devrait être le nombre d'itérations de la décomposition de domaine avec une inversion directe de la matrice $A_G + \delta$ par le nombre d'itération de l'inversion de la matrice $A_G^{-1}(A_G + \delta)$.

On utilise la méthode itérative *GMRES* pour résoudre le problème préconditionné suivant :

$$\left(I - A_D^{-1}(A_E - \delta)\right) X = A_D^{-1}b. \tag{3.12}$$

L'écriture $r = A_D^{-1}\, v$ signifie que l'on a résolu

$$\left\{ \begin{array}{rcl} A_B \, r_B & = & v_B \\ A_I \, r_I & = & v_I \\ A_H \, r_H & = & v_H \end{array} \right.$$

dont l'inversion du bloc A_I est obtenue par Réduction Cyclique. Pour les blocs A_B et A_H, on peut utiliser une méthode directe type décomposition LU car ils sont de très petites tailles.

Remarque On pense que le méthode itérative va converger assez rapidement. Il convient de stocker à l'itération k, k directions de descente, ce qui semble raisonnable vis à vis de la capacité mémoire des machines actuelles.

4

Réduction Cyclique

Sommaire

A chaque itération de la méthode itérative pour la résolution de la décomposition de domaine, on doit résoudre un système linéaire sur chaque sous-domaine. Les domaines contenant les couches *PML* sont suffisamment petits pour être traités par une méthode directe. Pour le domaine central, du fait de sa très grande taille, on opte naturellement vers un solveur rapide. On présente ici le principe général d'un type de méthode, appelé Réduction Cyclique, pour la résolution de grands systèmes linéaires provenant de la discrétisation d'opérateurs différentiels particuliers. Cette classe de méthode est basée sur l'élimination successive des inconnues du système. On ne résout finalement qu'une seule équation à une inconnue du système. On redistribue ensuite cette information (en sens inverse de l'élimination) aux inconnues éliminées. Ce type de méthode est articulée autour d'une suite d'opérations algébriques très simples. Ici, on s'intéresse à la méthode stabilisée proposée par O. Buneman [25].

Il existe une multitude de variantes de la Réduction Cyclique. On peut citer par exemple la méthode très connue *FACR* [34]. Toutefois, l'utilisation de méthodes rapides type transformée de Fourier n'est pas applicable à notre problème, excluant ainsi ce type de méthode. On se restreint dans cette partie à deux grandes approches a priori adaptées ; une première dite *standard* basée sur une analyse spectrale et une seconde appelée *PSCR* [22] basée sur une technique

de solutions partielles [26]. L'idée est de comparer les méthodes et surtout de justifier le choix d'utiliser la méthode *standard* et non la méthode *PSCR* récente et très utilisée pour les problèmes d'Helmholtz.

On présentera dans un premier temps le principe général de la Réduction Cyclique. On étudiera ensuite en détails les différentes approches. En particulier, la performance et la précision de ces deux méthodes dépendront de problèmes complexes aux valeurs et vecteurs propres. Suivant les approches, un très grand nombre de produits matrice-vecteur et de résolutions de problèmes tridiagonaux est à effectuer de manière rapide. Chaque approche a ses avantages et inconvénients que l'on présentera justifiant ainsi notre choix.

Remarque Il est important de noter qu'aucune version rapide de la Réduction Cyclique ne permet de résoudre notre problème complet (1.13) avec les couches *PML*. La symétrie ou séparabilité sont des propriétés fortes nécessaires à une résolution rapide que l'utilisation de *PML* en différences finies et la variation de densité moyenne ne permettent pas de conserver.

4.1 Présentation de la Réduction Cyclique

D'un point de vue historique, la Réduction Cyclique fut initialement proposée pour la résolution de système tridiagonaux issue de la discrétisation du problème de Poisson en différences finies puis fut étendue aux cas des matrices tridiagonales par blocs. De fait, la méthode présentée ici est complètement adaptée aux problèmes discrétisés par différences finies.

Pour la présentation de la méthode, notre problème modèle sera l'équation de Helmholtz (2.3) sur le domaine Ω_g définie par la décomposition (3.2). La discrétisation par différences finies du schéma (2.1) sur un maillage uniforme avec $n_x \times n_y$ noeuds amène à considérer le système linéaire tridiagonal par bloc suivant :

$$\begin{pmatrix} A/2+\alpha & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & A/2+\alpha \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n_y-1} \\ u_{n_y} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n_y-1} \\ f_{n_y} \end{pmatrix} \tag{4.1}$$

où les blocs A et T sont définies dans la section précédente.

En notant $B = A/2 + \alpha$, on réécrit alors la matrice (4.1) sous la forme générale

$$\begin{pmatrix} B & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & B \end{pmatrix}.$$

Pour plus de simplicité, supposons que $n_y = 2^k - 1$ avec $k \in \mathbb{N}^*$. Pour notre problème, on sait que les matrices A et T commutent. De plus, la matrice A est

symétrique. Ces deux propriétés sont primordiales pour la méthode. Toutefois, Swarztrauber [35] a étendu la méthode pour les matrices issues d'équations séparables générales mais la complexité des formules rend la méthode inintéressante. Par la suite, il sera mis en évidence l'intérêt des efforts effectués pour conserver les propriétés. Dans les formules suivantes, on pose $u_0 = u_{n_y+1} = 0$.

Considérons trois lignes successives (par bloc) de (4.1). On peut donc écrire pour $i = 2, 4, ..., n_y - 1$:

$$\left\{\begin{array}{ccccccccccc} -Tu_{i-2} & + & Au_{i-1} & - & Tu_i & & & & & = & f_{i-1} \\ & - & Tu_{i-1} & + & Au_i & - & Tu_{i+1} & & & = & f_i \\ & & & - & Tu_i & + & Au_{i+1} & - & Tu_{i+2} & = & f_{i+1}. \end{array}\right. \tag{4.2}$$

On multiplie alors les première et troisième équations de (4.2) par TA^{-1} et on les additionne à l'équation du milieu. On se ramène donc à un système réduit à $2^{k-1} - 1$ blocs d'inconnues :

$$-T^2A^{-1}u_{i-2} + \left(A - 2T^2A^{-1}\right)u_i - T^2A^{-1}u_{i+2} = f_i + TA^{-1}\left(f_{i-1} + f_{i+1}\right) \tag{4.3}$$

pour $i = 2, 4, ..., n_y - 1$.

Remarque Le traitement des bords est effectué de la même manière. On multiplie simplement la ligne du bord par TB^{-1} pour l'additionner à la ligne du milieu.

Après avoir résolu le système d'équations (4.3), on obtient les blocs solutions u_i pour $i = 2, 4, ..., n_y - 1$. Les inconnues éliminées du système peuvent être retrouvées en résolvant le système bloc diagonal :

$$\left\{\begin{array}{ll} Au_j = f_j + Tu_{j-1} + Tu_{j+1}, & j = 3, ..., n_y - 2 \\ Bu_j = f_j + Tu_{j-1} + Tu_{j+1}, & j = 1, n_y. \end{array}\right. \tag{4.4}$$

L'étape d'élimination peut être répétée $k-1$ fois pour n'avoir qu'un unique bloc équation $u_{(n_y+1)/2}$. Pour définir cette procédure récursive, on note $A^{(0)} = A$, $B^{(0)} = B$, $T^{(0)} = T$ et $f^{(0)} = f$. Ainsi, après r étapes d'élimination pour $0 \leq r \leq k-1$, le système réduit a la dimension de $2^{k-r} - 1$ blocs, et s'écrit :

$$\left(\begin{array}{ccccc} B^{(r)} & -T^{(r)} & & & \\ -T^{(r)} & A^{(r)} & -T^{(r)} & & \\ & \ddots & \ddots & \ddots & \\ & & -T^{(r)} & A^{(r)} & -T^{(r)} \\ & & & -T^{(r)} & B^{(r)} \end{array}\right)\left(\begin{array}{c} u_{2^r} \\ u_{2.2^r} \\ \vdots \\ u_{(n_y-1)-2^r+1} \\ u_{n_y-2^r+1} \end{array}\right)$$

$$= \left(\begin{array}{c} f^{(r)}_{2^r} \\ f^{(r)}_{2.2^r} \\ \vdots \\ f^{(r)}_{(n_y-1)-2^r+1} \\ f^{(r)}_{n_y-2^r+1} \end{array}\right).$$

où, pour $r = 1, ..., k-2$:

$$A^{(r)} = A^{(r-1)} - 2\left(T^{(r-1)}\right)^2 \left(A^{(r-1)}\right)^{-1} \tag{4.5}$$

$$B^{(r)} = A^{(r-1)} - \left(T^{(r-1)}\right)^2 \left[\left(A^{(r-1)}\right)^{-1} + \left(B^{(r-1)}\right)^{-1}\right] \tag{4.6}$$

$$T^{(r)} = \left(T^{(r-1)}\right)^2 \left(A^{(r-1)}\right)^{-1}. \tag{4.7}$$

Finalement, le bloc final de l'étape $r = k-1$ s'écrit

$$A^{(k-1)} = A^{(k-2)} - 2\left(T^{(k-2)}\right)^2 \left(B^{(k-2)}\right)^{-1}. \tag{4.8}$$

On obtient également une formulation récursive sur le second membre et pour $r = 1, ..., k-2$ et $i = 1, ..., 2^{k-r} - 1$:

$$f_{i.2^r}^{(r)} = \begin{cases} f_{i.2^r}^{(r-1)} + & T^{(r-1)}\left(\left(B^{(r-1)}\right)^{-1} f_{i.2^r-2^{r-1}}^{(r-1)} + \left(A^{(r-1)}\right)^{-1} f_{i.2^r+2^{r-1}}^{(r-1)}\right) \\ & \text{pour} \quad i = 1, \\ f_{i.2^r}^{(r-1)} + & T^{(r-1)}\left(\left(A^{(r-1)}\right)^{-1} f_{2^{k-r}-2^{r-1}}^{(r-1)} + \left(B^{(r-1)}\right)^{-1} f_{2^{k-r}+2^{r-1}}^{(r-1)}\right) \\ & \text{pour} \quad i = 2^{k-r} - 1, \\ f_{i.2^r}^{(r-1)} + & T^{(r-1)}\left(A^{(r-1)}\right)^{-1}\left(f_{i.2^r-2^{r-1}}^{(r-1)} + f_{i.2^r+2^{r-1}}^{(r-1)}\right) \\ & \text{autrement.} \end{cases} \tag{4.9}$$

De plus, on a pour $r = k-1$:

$$f_{2^{k-1}}^{(k-1)} = f_{2^{k-1}}^{(k-1)} + T^{(k-1)}\left(B^{(k-2)}\right)^{-1}\left(f_{2^{k-1}-2^{k-2}}^{(k-1)} + f_{2^{k-1}+2^{k-2}}^{(k-1)}\right). \tag{4.10}$$

Après les $k-1$ étapes d'élimination, l'équation bloc résultante s'écrit :

$$A^{(k-1)} u_{2^{k-1}} = f_{2^{k-1}}^{(k-1)}. \tag{4.11}$$

Après résolution de (4.8), on obtient le bloc milieu $u_{2^{k-1}}$. Le reste des inconnues sont obtenues par la résolution du système bloc diagonal des équations éliminées pour $r = k-2, ..., 0$. Ce système a la dimension $2^{k-r} - 1$ blocs, et s'écrit :

$$\begin{pmatrix} B^{(r)} & & & \\ & A^{(r)} & & \\ & & \ddots & \\ & & & B^{(r)} \end{pmatrix} \begin{pmatrix} u_{2^{r+1}-2^r} \\ u_{2.2^{r+1}-2^r} \\ \vdots \\ u_{n_y-2^r+1} \end{pmatrix} = \begin{pmatrix} g_{2^{r+1}-2^r}^{(r)} \\ g_{2.2^{r+1}-2^r}^{(r)} \\ \vdots \\ g_{n_y-2^r+1}^{(r)} \end{pmatrix} \tag{4.12}$$

où

$$g_{j.2^{r+1}-2^r}^{(r)} = f_{j.2^{r+1}-2^r}^{(r)} + T^{(r)}\left(u_{(j-1).2^{r+1}} + u_{(j).2^{r+1}}\right). \tag{4.13}$$

Dans la prochaine section, nous allons étudier une approche pour construire les seconds membres récursivement et traiter les différentes étapes de réduction et redistribution.

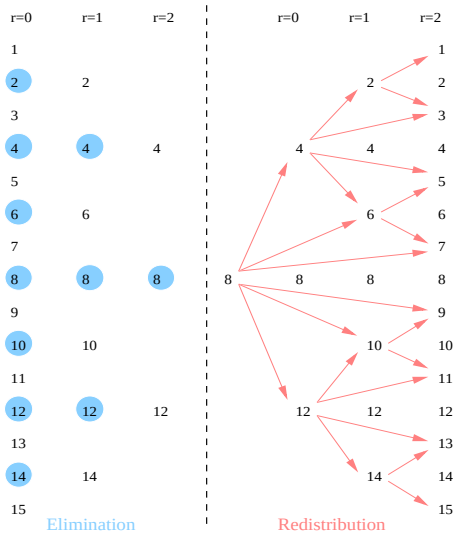


FIG. 4.1 – Etapes de la Réduction Cyclique pour 15 lignes

4.2 Approche par décomposition spectrale, la méthode *standard*

Pour la résolution des systèmes linéaires (4.10), (4.11) et (4.12), on peut se placer dans l'espace de la base des vecteurs propres de la matrice A, souvent appelé espace de Fourier. Tout d'abord, remarquons que si les matrices $A^{(r)}$ et $T^{(r)}$ commutent alors elles ont les mêmes vecteurs propres. C'est une propriété fondamentale de cette approche. On note $\Lambda^{(0)}$ et $\Gamma^{(0)}$ les matrices diagonales contenant respectivement les valeurs propres des matrices A et T. On a donc :

$$A = Q\Lambda^{(0)}Q^T, \quad T = Q\Gamma^{(0)}Q^T \tag{4.14}$$

où Q est la matrice orthogonale dont les colonnes sont les vecteurs propres des matrices A et T.

Ainsi, on obtient les formules de récurrence suivantes :

$$A^{(r)} = Q\Lambda^{(r)}Q^T, \quad T^{(r)} = Q\Gamma^{(r)}Q^T \tag{4.15}$$

où, en utilisant (4.6) et (4.7), on a :

$$\Lambda^{(r)} = \Lambda^{(r-1)} - 2\left(\Gamma^{(r-1)}\right)^2\left(\Lambda^{(r-1)}\right)^{-1} \tag{4.16}$$

$$\Gamma^{(r)} = \left(\Gamma^{(r-1)}\right)^2\left(\Lambda^{(r-1)}\right)^{-1}. \tag{4.17}$$

On voit immédiatement que pour que la matrice B ait les mêmes vecteurs propres que A et T, il est impératif que le paramètre α soit constant. C'est là encore une contrainte pour l'utilisation de cette approche. Dans ce cas,

$$B = Q\left(\frac{\Lambda^{(0)}}{2} + \alpha\right)Q^T,$$

et en posant $\Pi^{(0)} = \dfrac{\Lambda^{(0)}}{2} + \alpha$, on obtient à partir de (4.7) la formule de récurrence suivante

$$\Pi^{(r)} = \Lambda^{(r-1)} - \left(\Gamma^{(r-1)}\right)^2 \left[\left(\Lambda^{(r-1)}\right)^{-1} + \left(\Pi^{(r-1)}\right)^{-1}\right].$$

Le calcul de $x = \left(T^{(r-1)}\right)^2 \left(A^{(r-1)}\right)^{-1} y$ dans les étapes d'élimination est donné par :

$$x = Q \left(\Gamma^{(r-1)}\right)^2 \left(\Lambda^{(r-1)}\right)^{-1} Q^T y. \tag{4.18}$$

Les termes de la forme $x = \left(A^{(r)}\right)^{-1} \left(y + \left(T^{(r)}\right) z\right)$ dans les étapes de redistribution sont donnés par :

$$x = Q \left(\Lambda^{(r)}\right)^{-1} \left(Q^T y + \left(\Gamma^{(r)}\right) Q^T z\right). \tag{4.19}$$

Pour résumer, on effectue les cinq étapes suivantes :

- On calcule les composantes de f dans la base des vecteurs propres

$$\tilde{f}_i = Q^T f_i \quad \text{pour} \quad i = 1, \ldots, n_y.$$

- On effectue les étapes d'élimination sur le second membre obtenu, par exemple pour le bloc i à l'étape r

$$\tilde{f}^{(r)}_{i.2^r} = \tilde{f}^{(r-1)}_{i.2^r} + \Gamma^{(r-1)} \left(\Lambda^{(r-1)}\right)^{-1} \left(\tilde{f}^{(r-1)}_{i.2^r-2^{r-1}} + \tilde{f}^{(r-1)}_{i.2^r+2^{r-1}}\right).$$

  Les matrices $\Lambda^{(r-1)}$ et $\Gamma^{(r-1)}$ sont diagonales.

- On calcule le bloc solution réduit $\tilde{u}_{2^{k-1}}$ en résolvant le système

$$\Lambda^{(k-1)} \tilde{u}_{2^{k-1}} = \tilde{f}^{(k-1)}_{2^{k-1}}$$

  où la matrice $\Lambda^{(k-1)}$ est diagonale.

- On redistribue récursivement les blocs solutions en résolvant des systèmes du type

$$\Lambda^{(r)} \tilde{u}_{j.2^{r+1}-2^r} = \tilde{g}^{(r)}_{j.2^{r+1}-2^r}$$

  où

$$\tilde{g}^{(r)}_{j.2^{r+1}-2^r} = \tilde{f}^{(r)}_{j.2^{r+1}-2^r} + \Gamma^{(r)} \left(\tilde{u}_{(j-1).2^{r+1}} + \tilde{u}_{(j).2^{r+1}}\right).$$

- On calcule la solution u

$$u_i = Q \tilde{u}_i \quad \text{pour} \quad i = 1, \ldots, n_y.$$

Le coût global de cette approche, mis à part le calcul des vecteurs et valeurs propres de A, peut être obtenu en comptant le nombre de produit matrice-vecteur par Q et Q^T, soit $2n_y$ multiplications.

4.3 Approche par technique de solution partielle, la méthode *PSCR*

On présente maintenant une autre approche récente basée sur la technique de solution partielle [26]. L'idée est de construire les blocs de solutions des différentes étapes de redistribution en revenant à chaque fois au problème complet. On introduit dans un premier temps différentes écritures complémentaires à la Réduction Cyclique. On présente ensuite dans ce cadre la technique de solution partielle basée sur la séparabilité du problème. On montrera ensuite les avantages et limites de cette approche.

Remarque Récemment introduite par T. Rossi et J. Toivanen [22], la Réduction Cyclique avec la technique de solution partielle est appelée *PSCR*. Elle est connue pour son grand nombre d'applications, notamment pour les problèmes d'Helmholtz [24] en éléments finis et sa parallélisation [23].

4.3.1 Etapes de réduction et redistribution

On présente ici plusieurs résultats pratiques pour la construction du second membre ou la redistribution aux inconnues éliminées en vue de l'utilisation de la technique dite de solution partielle. Pour les démonstrations précises, voir [22].

Voici tout d'abord un lemme qui sera utile pour les deux prochains théorèmes.

Lemme 4.3.1 *Pour la matrice $T^{(s)}$, $s = 1, ..., k$, on a*

$$T^{(s)} = T^{(0)} \left(A^{(0)}\right)^{-1} T^{(0)} \left(A^{(1)}\right)^{-1} T^{(1)} ... \left(A^{(s-1)}\right)^{-1} T^{(s-1)}. \tag{4.20}$$

On note pour la suite

$$M(r) = \begin{pmatrix} A & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & A \end{pmatrix}$$

le bloc de A de dimension égale à $2^r - 1$. Par soucis de simplicité, on ne considère pas ici les termes de bords.

4.3.2 Construction du second membre

On s'intéresse au calcul des vecteurs $f^{(r)}$ dans (4.9). On s'appuie sur le théorème suivant :

Théorème 4.3.2 *Soit $m = 2^r - 1$. Alors,*

$$f_{i.2^r}^{(r)} = f_{i.2^r}^{(r-1)} + T\left(x_m^{(1)} + x_0^{(2)}\right) \tag{4.21}$$

où $x_m^{(1)}$ est le dernier bloc de taille n du vecteur $x^{(1)}$ et $x_0^{(2)}$ est le premier bloc de taille n du vecteur $x^{(2)}$, où les vecteurs $x^{(k)}$, $k = 1, 2$ sont donnés par les systèmes linéaires

$$M^{(r)}x^{(k)} = y^{(k)}. \tag{4.22}$$

Les vecteurs $y^{(k)}$ ont la structure par bloc suivante

$$y_j^{(k)} = \begin{cases} f^{(r)}_{i.2^{r+1}+(-1)^k 2^r}, & j = (m+1)/2 \\ 0, & autrement. \end{cases} \tag{4.23}$$

Preuve Il est évident que résoudre $M^{(r)}x = y$, avec y nul sauf pour le bloc (m+1)/2, équivaut à résoudre $A^{(r-1)}x_{(m+1)/2} = y_{(m+1)/2}$. On a donc

$$x_{(m+1)/2} = \left(A^{(r-1)}\right)^{-1} y_{(m+1)/2}. \tag{4.24}$$

En utilisant la formule de redistribution (4.12), pour chaque étape $j = r-2, ..., 0$ et en s'intéressant au premier bloc, on a

$$\begin{array}{rcl} Tx_1 & = & T^{(0)}x_1 \\ & = & T^{(0)}\left(A^{(0)}\right)^{-1} T^{(0)}\left(A^{(1)}\right)^{-1} T^{(1)}... \\ & & ...\left(A^{(r-2)}\right)^{-1} T^{(r-2)}x_{(m+1)/2}. \end{array} \tag{4.25}$$

En utilisant le lemme 4.3.1, on a

$$Tx_1 = T^{(r-1)}x_{(m+1)/2} = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} y_{(m+1)/2}. \tag{4.26}$$

Ainsi, dans un premier temps, on résout le problème associé $M^{(r)}x^{(1)} = y^{(1)}$. D'après le théorème 4.3.2, on obtient que

$$Tx_m^{(1)} = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} f^{(r)}_{i.2^{r+1}-2^r}$$

où $x_m^{(1)}$ est le dernier bloc du vecteur $x^{(1)}$.
De plus, pour le problème avec le second membre $y^{(2)}$, on a

$$Tx_1^{(2)} = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} f^{(r)}_{i.2^{r+1}+2^r}.$$

où $x_1^{(2)}$ est le premier bloc du vecteur $x^{(2)}$.
On peut alors facilement calculer le second membre $f^{(r)}$ donné par (4.9). A chaque étape de réduction, on a besoin de résoudre $2^{k-r} - 1$ systèmes linéaires avec la matrice $M^{(r)}$.

4.3.3 Redistribution

On s'intéresse maintenant à la redistribution aux inconnues éliminées. On utilise le théorème suivant :

Théorème 4.3.3 *Soit $m = 2^r - 1$. Alors, $u_{i.2^{r+1}-2^r} = x_{(m+1)/2}$, où le bloc milieu $x_{(m+1)/2}$ est de taille n_x du vecteur solution x du système $M^{(r)}x = y$. Le vecteur y est défini comme*

$$y_j = \begin{cases} Tu_{(i-1).2^{r+1}}, & j = 1, \\ f^{(r)}_{i.2^{r+1}-2^r}, & j = (m+1)/2, \\ Tu_{i.2^{r+1}}, & j = m, \\ 0, & autrement, \end{cases} \tag{4.27}$$

sauf pour $r = 0$ et $r = k-1$. Dans le cas $r = k-1$, on a $y_1 = y_2 = 0$. Dans le cas $r = 0$, l'unique bloc y_1 est donné par $y_1 = f^{(0)}_{2.(i-1)} + T(u_{2.(i-1)} + u_{2.i})$.

Preuve La linéarité du système nous permet d'écrire

$$x_{(m+1)/2} = x^{(1)}_{(m+1)/2} + x^{(2)}_{(m+1)/2} + x^{(3)}_{(m+1)/2},$$

où $x^{(k)}_{(m+1)/2}$ est le bloc milieu de la solution $x^{(k)}$ du système $M^{(r+1)}x^{(k)} = y^{(k)}$. Les vecteurs $y^{(k)}$ sont définis comme :

$$y^{(1)}_j = \begin{cases} f^{(r)}_{i.2^{r+1}-2^r}, & j = (m+1)/2 \\ 0, & autrement, \end{cases}$$

$$y^{(2)}_j = \begin{cases} Tu_{(i-1).2^{r+1}}, & j = 1 \\ 0, & autrement, \end{cases}$$

$$y^{(3)}_j = \begin{cases} Tu_{i.2^{r+1}}, & j = m \\ 0, & autrement. \end{cases}$$

Dans le même esprit que pour le théorème précèdent, on montre que

$$x^{(2)}_{(m+1)/2} = \left(A^{(r)}\right)^{-1} T^{(r)} u_{(i-1).2^{r+1}},$$

et que

$$x^{(3)}_{(m+1)/2} = \left(A^{(r)}\right)^{-1} T^{(r)} u_{i.2^{r+1}}.$$

Ainsi, chaque redistribution $0 \leq r \leq k-1$ nécessite la résolution de $2^{k-r} - 1$ problèmes avec la matrice $M^{(r+1)}$.

4.3.4 Séparabilité du problème

On définit tout d'abord le produit tensoriel de matrices $\otimes$ permettant l'écriture en variables séparées : pour $A \in \mathbb{C}^{k\times k}$ et $B \in \mathbb{C}^{s\times s}$,

$$A \otimes B = \{A_{ij}B\}_{i,j=1}^{k} \in \mathbb{C}^{ks\times ks}.$$

On s'intéresse ici en détail à l'écriture de la matrice (4.1) issue d'une discrétisation par différences finies sous la forme

$$D_y \otimes A_x + A_y \otimes D_x.$$

D'après le principe de séparation, les matrices A_x et A_y de taille respectives $n_x \times n_x$ et $n_y \times n_y$ représentent la discrétisation du problème (4.1) dans les directions respectives x et y. Les matrices D_x et D_y traduisent l'impact des conditions aux bords.

Remarque On peut également écrire sous forme le problème discrétisé par la méthode des éléments finis. C'est d'ailleurs un avantage considérable de cette approche.

On considère le cas général défini par le problème (2.3) associé aux conditions de Robin. Avec une discrétisation de la dérivée normale à l'ordre deux dans les conditions d'interfaces, on a la matrice suivante

$$\begin{pmatrix} A/2+\alpha & -T & & & \\ -T & A & -T & & \\ & \ddots & \ddots & \ddots & \\ & & -T & A & -T \\ & & & -T & A/2+\alpha \end{pmatrix}$$

où α est constant. On rappelle également que

$$A = -\epsilon^2(M_\Delta + M_{CL}) - M_{1-N}$$

où en notant $h = \frac{\delta x}{\delta y}$

$$M_\Delta = \frac{1}{\delta x^2}\begin{bmatrix} -1-2h^2 & 1 & & & & \\ 1 & -2(1+h^2) & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2(1+h^2) & 1 & \\ & & & 1 & -1-2h^2 \end{bmatrix},$$

$$M_{1-N} = \begin{bmatrix} 1-N_1 & & & \\ & 1-N_2 & & \\ & & \ddots & \\ & & & 1-N_n \end{bmatrix}$$

et

$$M_{CL} = -\frac{\mathrm{i}}{\epsilon\delta x}\left(\sqrt{1-N_1}\delta_1 + \sqrt{1-N_n}\delta_n\right)$$

où on a noté δ_i la matrice de taille $n \times n$ dont le $i$ème coefficient de la diagonale vaut un et zéro partout ailleurs. On a également

$$T = -\frac{\epsilon^2}{\delta y^2} I.$$

On peut alors écrire la décomposition en notant

$$A_x = -\epsilon^2(\frac{1}{\delta x^2}M + M_{CL}) - M_{1-N}$$

et

$$A_y = -\frac{\epsilon^2}{\Delta y^2} M + \alpha(\delta_1 + \delta_n)$$

où

$$M = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}.$$

On pose également $D_y = I - \frac{1}{2}(\delta_1 + \delta_n)$ et $D_x = I$. On a ainsi

$$D_y \otimes A_x = \begin{bmatrix} A_x/2 & & & & \\ & A_x & & & \\ & & \ddots & & \\ & & & A_x & \\ & & & & A_x/2 \end{bmatrix}$$

et

$$A_y \otimes D_x = \frac{1}{\delta y^2} \begin{bmatrix} (1+\delta y^2\alpha)D_x & -D_x & & & \\ -D_x & 2D_x & -D_x & & \\ & \ddots & \ddots & \ddots & \\ & & -D_x & 2I_x & -D_x \\ & & & -D_x & (1+\delta y^2\alpha)D_x \end{bmatrix}.$$

Par cette décomposition, on retrouve bien la matrice (4.1).

Remarque La chose importante à noter pour la suite est que la matrice D_y n'est pas la matrice identité.

4.3.5 Technique de solution Partielle

On présente ici la technique de solution partielle pour résoudre les systèmes linéaires $M_{(r)}$ et $M_{(r+1)}$. Cette méthode particulière nécessite que l'opérateur soit à variables séparables. On vient de voir que le problème (4.1) se formalise comme

$$(D_y \otimes A_x + A_y \otimes D_x)\, u = f.$$

Dans le cadre d'une discrétisation par différences finies, les matrices A_x et A_y de taille respectives $n_x \times n_x$ et $n_y \times n_y$ sont tridiagonales et issues de la discrétisation de l'opérateur dans les directions respectives x et y. De plus, les matrices D_x et D_y sont diagonales et varient en fonction des conditions aux bords. Un avantage direct de cette approche est le fait de pouvoir également utiliser une discrétisation type éléments finis.

Il suit que la matrice $M^{(r)}$ est séparable et peut s'écrire :

$$M^{(r)} = J_{2^r-1} \otimes A_x + K_{2^r-1} \otimes D_y \tag{4.28}$$

où K_{2^r-1} (resp. I_{2^r-1}) est une sous-matrice de A_y (resp. D_y) de taille 2^r-1 blocs. Soit le problème généralisé aux valeurs propres suivant :

$$K_{2^r-1}w_i = \lambda_i J_{2^r-1}w_i, \quad i = 1, ..., 2^r-1.$$

On note par la suite la base des vecteurs propres

$$W = [w_1, w_2, ..., w_{2^r-1}] = W^T \in \mathbb{C}^{(2^r-1)\times(2^r-1)}.$$

Dans l'étape de réduction, on a besoin de calculer

$$M^{(r)}x = y \tag{4.29}$$

où y a seulement un bloc non nul, celui du milieu $y_{2^{r-1}}$. On ne veut que les blocs x_1 et x_{2^r-1} de la solution x. En multipliant (4.29) par la gauche par la matrice $W^T \otimes I_{n_x}$, en utilisant (4.28) et en notant $\widetilde{x} = (W^T \otimes I_{n_x})x$, on obtient le système diagonal par bloc :

$$\begin{pmatrix} A_x + \lambda_1 I_{n_x} & & & \\ & A_x + \lambda_2 I_{n_x} & & \\ & & \ddots & \\ & & & A_x + \lambda_{2^r-1} I_{n_x} \end{pmatrix} \begin{pmatrix} \widetilde{x}_1 \\ \widetilde{x}_2 \\ \vdots \\ \widetilde{x}_{2^r-1} \end{pmatrix} = \begin{pmatrix} (w_1)_{2^{r-1}} y_{2^{r-1}} \\ (w_2)_{2^{r-1}} y_{2^{r-1}} \\ \vdots \\ (w_{2^r-1})_{2^{r-1}} y_{2^{r-1}} \end{pmatrix} \tag{4.30}$$

où I_{n_x} est la matrice identité de taille $n_x \times n_x$.

Finalement, on pose $x = (W \otimes I_{n_x})\widetilde{x}$ et, après avoir résolu le système (4.30), les blocs souhaités sont obtenus par :

$$x_1 = \sum_{j=1}^{2^r-1} (w_j)_1 \widetilde{x}_j \quad \text{et} \quad x_{2^r-1} = \sum_{j=1}^{2^r-1} (w_j)_{2^r-1} \widetilde{x}_j. \tag{4.31}$$

C'est cette variante de la méthode classique de séparation de variable que l'on appelle la technique de solution partielle.

Le problème en $M^{(r+1)}$ de redistribution se résout de manière identique.

4.4 Choix de la méthode

On a présenté deux manières différentes de résoudre le problème (4.1) par Réduction Cyclique ; la méthode par décomposition spectrale dite *standard* et la méthode par solution partielle notée *PSCR*.

La méthode *PSCR* possède un avantage certain comparé à la méthode *standard* ; elle n'est pas restreinte à des maillages cartésiens type différences finies. On peut l'utiliser pour des méthodes type éléments finis. Cette particularité permet donc une application plus générale de la méthode. Rappelons toutefois que la méthode de Réduction Cyclique est à l'origine conçue pour des problèmes issues de discrétisation type différences finies. En fait, la résolution par technique

des solutions partielles consiste à considérer les sous-problèmes issues de (4.1) sans explicitement effectuer les récurrences. On utilise plutôt le caractère séparable de l'opérateur pour calculer les blocs solutions. Dans cette approche, il est nécessaire de résoudre une série de problèmes aux valeurs propres généralisés. La résolution de ce type de problèmes s'avère très compliquée, surtout dans l'espace complexe et aux dimensions de maillage envisagés. Dans cette approche, la symétrie de la matrice 1D A_x est primordiale.

L'approche *standard* nécessite quant à elle la symétrie de la matrice A formant le bloc diagonale. Dans notre cas, on utilise le fait que non seulement la matrice globale est séparable mais aussi que la structure est simple. En effet, les blocs extra-diagonaux sont tous identiques égaux à T et les matrices A et T commutent. Les conditions limites d'ordre élevé ne sont alors pas adaptées. En contre partie, on ne doit résoudre qu'un unique problème aux valeurs propres pour lequel une méthode ultra-rapide et performante peut être utilisée. On la présente d'ailleurs dans le chapitre suivant. Toutefois, dans notre cadre d'étude et pour des raisons de conception, l'utilisation des différences finies permet de pallier ces contraintes.

L'avantage indéniable de la méthode *standard* sur la méthode *PSCR* est sa simplicité algorithmique et de mise en oeuvre. En effet, la méthode s'écrit simplement et le noeud de la méthode est une séquence de produit matrice-vecteur par une matrice dense. La méthode *PSCR* quant à elle nécessite également la résolution d'une série de systèmes tridiagonaux.

Pour ces raisons, on choisit la méthode *standard* qui est plus adaptée à la discrétisation par différences finies. L'utilisation de la méthode *PSCR* s'envisagera plutôt pour des problèmes discrétisés en éléments finis.

Remarque Dans l'article [24], les auteurs font la résolution d'un problème d'Helmholtz avec des couches *PML* et sans décomposition de domaine par la méthode *PSCR*. Il est intéressant de comprendre les différences avec notre cas. Tout d'abord, le problème que considèrent les auteurs est un problème d'Helmholtz à coefficient constant. Ensuite, la technique des éléments finis est utilisée. Dès lors, il est possible d'avoir un problème variationnel symétrique et séparable. Dans notre cas, en différences finies, la symétrie du problème avec couches *PML* n'est tout simplement pas possible. Dans un cadre éléments finis, la variation de la densité moyenne empêche de garantir à la fois la séparabilité et la symétrie.

4.5 Performance de la méthode

La performance de la Réduction Cyclique avec décomposition spectrale est directement liée aux produits faisant intervenir la matrice Q (et sa transposée). D'un point de vue historique, cette méthode fut éprouvée sur des problèmes elliptiques simples. Dans un tel cadre, on dispose du spectre (Λ, Q) et on effectue les produits par transformée de Fourier rapide (FFT, [32]). Dans notre cas, c'est plus complexe et la FFT ne s'applique pas. Dès lors, la séquence de produit matrice-vecteur doit être effectuée de manière adaptée et performante d'un point de vue informatique.

On rappelle que l'on a n_y vecteurs de taille n_x à multiplier avec la matrice dense composée des vecteurs propres Q puis sa transposée Q^T. Pour tout ceci, on doit donc effectuer exactement $2n_y n_x^2$ multiplications et $2n_y n_x(n_x-1)$ additions. Précisons que l'on parle évidement d'opérations sur des nombres complexes et que les multiplications de flottants sont beaucoup coûteuses que les additions. Il existe des algorithmes récursifs permettant de diminuer fortement ce nombre de multiplications mais augmentant beaucoup le nombre d'additions [33]. Toutefois, la structure des algorithmes ne permet pas l'exploitation du cache des processeurs.

On peut naïvement effectuer la séquence de produit matrice-vecteur en utilisant une bibliothèque scientifique ou en codant soi-même. Toutefois, grâce à des techniques dites de *cache blocking* des données [28], effectuer cette séquence en un seul produit matrice-matrice permet d'utiliser au mieux les capacités des machines informatiques actuelles. On effectue un produit d'une matrice $n_x \times n_x$ par une matrice $n_x \times n_y$. Il est possible (mais non aisé) de faire sa propre routine produit matrice-matrice performante mais le développement nécessite une forte connaissance de la machine et dépend de cette dernière. Il est alors préférable d'utiliser des bibliothèques qui s'adaptent automatiquement à la machine, par exemple les bibliothèques *LAPACK* [27] *ATLAS* [30] ou *PHIPAC* [29] ou des routines particulières faites à la main comme par Goto [31].

5

Solveur spectral

Sommaire

La Réduction Cyclique est le solveur direct utilisé lors de la résolution du système linéaire préconditionné (3.12). On a vu précédemment que cette méthode est basée sur une séquence d'opérations algébriques sur les valeurs et vecteurs propres du bloc diagonal de la matrice de discrétisation. La précision de la Réduction Cyclique est ainsi directement liée à la précision de la méthode de recherche du spectre. C'est donc une étape cruciale de la résolution. Dans ce chapitre, on présente un solveur aux valeurs et vecteurs propres ultra-rapide et précis adapté aux structures tridiagonales. Ce solveur est basé sur les travaux récents de B. Parlett [14] et I. Dhillon [15]. Initialement prévue pour les cas réels, on s'est ici intéressé à l'implantation dans le cas complexe. On comparera en particulier ce solveur à la méthode classique QR.

On considère une matrice A tridiagonale symétrique (non hermitienne dans le cas complexe) de dimension n. On souhaite trouver toutes les valeurs propres $\lambda_i \in \mathbb{C}$ et les vecteurs propres associés v_i qui vérifient

$$Av_i = \lambda_i v_i. \tag{5.1}$$

Autrement dit, on cherche les matrices Λ et Q vérifiant

$$A = Q\Lambda Q^{-1}.$$

On sait que dans le cas réel, si la matrice A est symétrique alors la matrice Q formée par la base de ses vecteurs propres est orthogonale ($Q^{-1} = Q^T$). Dans le cas complexe, si la matrice est hermitienne alors la matrice Q est unitaire ($Q^{-1} = Q^*$). On s'intéresse aux propriétés de notre matrice symétrique mais non hermitienne à laquelle on associe le produit scalaire réel

$$(u, v)_{\mathbb{R}} = \sum u_i v_i.$$

On suppose que toutes les valeurs propres de A sont distinctes. Soient (u,v) deux vecteurs propres associées aux valeurs propres respectives (λ,μ) :

$$\left\{ \begin{array}{rcl} Au & = & \lambda u, \\ Av & = & \mu v. \end{array} \right.$$

On a donc

$$\lambda(u, v)_{\mathbb{R}} = (\lambda u, v)_{\mathbb{R}} = (Au, v)_{\mathbb{R}} = (u, A^T v)_{\mathbb{R}} = (u, Av)_{\mathbb{R}} = (u, \mu v)_{\mathbb{R}} = \mu(u, v)_{\mathbb{R}}$$

et comme $\lambda \neq \mu$, on déduit que $(u, v)_{\mathbb{R}} = 0$. Les vecteurs propres sont donc orthogonaux au sens du produit scalaire réel. On peut alors écrire

$$A = Q \Lambda Q^T.$$

On voit donc qu'une contrainte importante est d'obtenir l'orthogonalité.

Il existe de nombreuses méthodes permettant d'obtenir les valeurs propres. Par exemple, on peut utiliser l'algorithme classique QR. Par contre, aucune de ces méthodes ne tient compte du caractère tridiagonal de la matrice. On s'intéresse à une méthode initialement proposée par H. Rutishauser [17], l'algorithme LR (ou QD dans sa version tridiagonale initiale). D'un point de vue historique, cette méthode précède l'algorithme QR et est basée sur le même procédé itératif. De plus, les mêmes techniques d'accélération de la méthode par décalage sont utilisables. Bien qu'applicable en théorie dans le cas de matrice générale, cette méthode est sans doute la plus performante pour le cas des matrices tridiagonales. En effet, elle repose sur des décompositions LU, très bien adaptées pour ce type de matrice. B. Parlett en fait une étude très claire [14].

Classiquement, on obtient le vecteur propre v_i relié à la valeur propre λ_i par la résolution de l'équation (5.1). On doit donc trouver le noyau de l'application $(A - \lambda_i I)$. On va utiliser une méthode adaptée à notre cas proposée par I. Dhillon en collaboration avec B. Parlett. Le but de cette méthode est de garantir l'orthogonalité des vecteurs sans passer par des procédés d'orthogonalisation type Gram-Schmit. Cette méthode est encore basée sur des décompositions LU.

Rappelons enfin que notre motivation est de ne pas se restreindre au cas réel mais de traiter le cas complexe. En effet, si de nombreuses bibliothèques permettent la résolution de ce problème dans le cas réel, aucune ne traite le cas complexe non hermitien efficacement. C'est même en utilisant des routines pour les cas de matrices complexes générales que les calculs sont généralement faits. Dès lors, l'implémentation d'une méthode adaptée et performante par rapport à ce qui est disponible devient une réelle motivation pour notre travail.

5.1 Calcul des valeurs propres

La méthode LR est particulièrement bien adaptée aux matrices de type tridiagonal. On ne tire par contre aucun avantage de la structure symétrique. On choisit alors un formalisme pour matrice générale. On écrit notre matrice

$$A = \begin{bmatrix} a_1 & c_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}.$$

5.1.1 Ecriture de l'algorithme

On présente la version proposée par R. Rutishauser [17] :

$A_1 = A$,
pour $k = 1, 2, \ldots$:
$A_k = LU$
$A_{k+1} = UL$
fin

Cet algorithme est d'écriture très proche de l'algorithme QR et repose sur le même principe de transformation : Une première réaction en voyant cette procédure est de se demander ce qu'il advient des valeurs propres et ce qu'on accomplit. Comme

$$A_{k+1} = UL = UA_kU^{-1} = L^{-1}A_kL$$

U étant une bijection, A_{k+1} peut être vu comme un changement de variable et donc il y a équivalence de spectre.

Cette algorithme converge vers une matrice bidiagonale. En effet, pendant les itérations, on vérifie que les coefficients de la diagonale inférieure de A_k convergent un à un vers zéro. Le premier à diminuer est le coefficient $(A_k)_{n,n-1}$, puis $(A_k)_{n-1,n-2}$, etc. De plus, lorsque un coefficient s'annule, on assure la convergence de la valeur propre sur la diagonale. Remarquons que cet algorithme n'ordonne pas les valeurs propres.

5.1.2 Forme tridiagonale conservée

L'intérêt de cette écriture est garder une forme tridiagonale à toutes les étapes s'il n'y a pas besoin de pivotage numérique. En effet, la décomposition LU garde le profil de la matrice initiale et si on note L la matrice inférieure

$$L = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & 1 & \\ & & l_{n-1} & 1 \end{bmatrix},$$

U la matrice supérieure

$$U = \begin{bmatrix} u_1 & c_1 & & \\ & u_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & & u_n \end{bmatrix}$$

alors le produit UL est une matrice tridiagonale et s'écrit

$$UL = \begin{bmatrix} u_1 + c_1 l_1 & c_1 & & \\ u_2 l_1 & u_2 + c_2 l_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & u_n l_{n-1} & u_n \end{bmatrix}. \tag{5.2}$$

Dès lors, on utilise efficacement la structure de la matrice, garantissant ainsi de bonnes performances par rapport aux autres algorithmes généraux.

5.1.3 Accéleration

On présente ici deux manières différentes d'accélérer l'algorithme ; l'utilisation classique de shift et une réduction de la taille du problème au fur et à mesure de la convergence.

Par l'utilisation de shift

De manière similaire à l'algorithme QR, on peut accélérer la méthode en utilisant des shifts aux différentes étapes. On réécrit l'algorithme :

$A_1 = A$,
pour $k = 1, 2, \ldots$:
$A_k - c_k I = LU$
$A_{k+1} = UL + c_k I$
fin

où $c_k \in \mathbb{C}$ est notre shift à chaque étape.

Le choix du shift s'avère primordial pour l'accélération de la méthode. Un choix classique pour c_k est

$$c_k = (A_k)_{n,n}.$$

Comme le seul terme non nul de la dernière ligne de A_k est $(A_k)_{n,n-1}$. Clairement, si $(A_k)_{n,n-1} \to 0$ alors $(A_k)_{n,n} \to \lambda_n$. On peut montrer que $(A_k)_{n,n-1}$ tend vers zéro avec le facteur de convergence

$$\left| \frac{\lambda_n - c_k}{\lambda_{n-1} - c_k} \right|.$$

Le meilleur choix possible serait directement la valeur propre λ_n. Un autre choix astucieux, notamment en complexe, est de prendre une valeur propre du bloc 2×2 extrait de A_k de coefficients diagonaux d'indices n et $n-1$

$$\begin{bmatrix} a_{n-1} & c_{n-1} \\ b_{n-1} & a_n \end{bmatrix}.$$

On choisit la valeur propre la plus proche de $(A_k)_{n,n}$. Cette technique a été introduite par J. H. Wilkinson [16].

Par la structure

La matrice A_k converge vers une matrice bidiagonale supérieure. Ceci implique qu'au fur et à mesure des itérations, le dernier coefficient de la diagonale inférieure de la décomposition LU de A_k converge vers zéro. Ainsi, le dernier coefficient de la diagonale inférieure de la matrice $A_{k+1} = UL$ est également nul. On peut alors ne considérer que la matrice réduite de la ligne et la colonne associées au coefficient aux itérations suivantes. On réduit ainsi la taille du problème au fur et à mesure que l'on converge vers les valeurs propres.

5.1.4 Remarques sur l'implantation de la méthode *LR*

L'idée de l'implémentation est de toujours se ramener à une écriture type LU des matrices A_k sans jamais les reconstruire explicitement. Ainsi, à l'itération k, on cherche directement les matrices L et U de la décomposition LU de la matrice A_{k+1} à partir de la formule (5.2). D'un point de vue informatique, on peut éviter certaines erreurs d'arrondi en limitant les additions, génératrice d'erreur, et introduire des multiplications.

On a plusieurs choix possibles pour le critère d'arrêt :

- le terme $A_{k-1,k} \to 0$. Ce test assure que la convergence a lieu vers une valeur propre sans quantifier la précision.
- la différence entre $A_{k,k}$ et D_k (la diagonale à l'itération précédente). Ce test permet de voir si la méthode stagne.
- On peut également combiner les deux tests précédents.

Si on vérifie le test, on a donc une convergence vers une valeur propre. Cette convergence locale permet de se focaliser sur la recherche de valeurs propres dans la matrice réduite.

5.2 Calcul des vecteurs propres

On se propose d'étudier le calcul des vecteurs propres $v_i \neq 0$ de la matrice symétrique (non hermitienne) A en utilisant les valeurs propres obtenues à l'aide de l'algorithme LR présenté précédemment.

On cherche donc à déterminer le noyau des applications $A - \lambda_i I$ où $\lambda_i \in \mathbb{C}$. Cela exige donc de résoudre les systèmes suivants :

$$\begin{bmatrix} a_{1,1}-\lambda_i & a_{1,2} & & \\ a_{1,2} & a_{2,2}-\lambda_i & \ddots & \\ & \ddots & \ddots & a_{n-1,n} \\ & & a_{n-1,n} & a_{n,n}-\lambda_i \end{bmatrix} \begin{pmatrix} v_i^1 \\ v_i^2 \\ \vdots \\ v_i^n \end{pmatrix} = 0. \tag{5.3}$$

Rappelons que le noyau de cette application est un espace unidimensionnel s'il n'existe pas de valeur propre double. Une manière naturelle pour trouver v_i

est de choisir un indice pivot k et de poser $v_i^k = 1$. On déduit ensuite le reste des composantes du vecteur par descente et montée du système. Par contre, cette méthode ne donne pas toujours des vecteurs propres orthogonaux au sens du produit scalaire réel. Il faut donc utiliser un algorithme d'orthogonalisation comme l'algorithme de Gram-Schmidt. De plus, dans certains cas, les vecteurs obtenus pointent dans de mauvaises directions. La raison est qu'il est rare qu'une valeur propre soit représentable en précision finie.

Pourtant, S.I. Dhillon a développé une méthode très performante donnant des vecteurs propres directement orthogonaux. Elle est basée sur un choix astucieux du pivot pour chaque vecteur propre dépendant de la factorisation particulière de la matrice du noyau. Une technique de raffinage des valeurs propres est aussi employée. Une conséquence directe est la rapidité de l'algorithme ($\mathrm{O}(n^2)$) car on évite l'orthogonalisation de Gram-Schmidt.

5.2.1 Principe de la méthode

Le principe de base est de décomposer la matrice $A-\lambda_i I$ par les factorisations de Cholesky supérieure et inférieure :

$$A - \lambda_i I = L_+ D_+ L_+^T = U_- D_- U_-^T.$$

Lors de ces factorisations, on définit plusieurs grandeurs, notamment des coefficients γ_k traduisant la qualité des vecteurs propres obtenus pour lequel $v_i^k = 1$. S. I. Dhillon [15] propose de choisir le pivot k comme l'indice pour lequel $|\gamma_k|$ est minimum. On obtient alors une bonne approximation du vecteur propre v_i. On définit ainsi une factorisation particulière dite twisted au pivot k couplant les deux précédentes

$$A - \lambda_i I = N_k D_k N_k^T.$$

On utilise la twisted factorisation pour calculer le vecteur propre. En respectant la littérature, on appelle cette procédure *GetVec*. On définit également l'Algorithme X en obtenant tous les vecteurs propres avec *GetVec*.

On définit la fonction suivante :

$$\mathrm{reldist}(\lambda_i, \lambda_j) = \left| \frac{\lambda_i - \lambda_j}{\lambda_j} \right|$$

traduisant la distance relative entre 2 valeurs propres. On considérera que 2 valeurs propres sont proches si leur distance est inférieure à 10^{-3}.

Dans le cas de valeurs propres isolées, les vecteurs propres par l'Algorithme X sont obtenus précisément. Par contre, dans le cas de valeurs propres proches, l'orthogonalité n'est pas assurée suffisamment précisément. Dhillon a alors introduit la notion d'éloignement relatif et d'affinage des valeurs propres.

On appelle cluster la liste des valeurs propres proches. On cherche à éloigner relativement ces valeurs propres. On décale alors la matrice par un scalaire σ choisi dans le cluster.

On doit ensuite affiner les valeurs propres qui sont clustérisées et maintenant décalées. Pour cela, on utilise un dérivé de la méthode inverse, le Rayleigh Quotient Iteration (RQI). Cet algorithme est utilisé pour raffiner une petite

valeur propre λ d'une matrice A. Il repose sur la méthode de la puissance inverse. On calcule alors

$$v_{k+1} = (A - \lambda I)^{-1} v_k.$$

On raffine donc toutes les valeurs propres du cluster et on vérifie qu'il n'y ait pas d'autres clusters dans le cluster. Si une valeur n'appartient pas à un cluster, alors on applique alors la procédure $GetVec$. On appelle cette procédure l'Algorithme Y. On propose plutôt une variante qui consiste à raffiner toutes les valeurs propres par la méthode RQI et de calculer ensuite tous les vecteurs propres avec $GetVec$. On l'appelle l'Algorithme X'.

5.2.2 La twisted factorisation

Soit J une matrice tridiagonale complexe telle que l'on ait la double factorisation

$$J = L_+ D_+ L_+^T = U_- D_- U_-^T. \tag{5.4}$$

On définit la twisted factorisation au pivot k comme

$$J = N_k D_k N_k^T \tag{5.5}$$

où les matrices N_k et D_k ont les formes suivantes

$$N_k = \begin{bmatrix} x & & & & & \\ x & x & & & & \\ & x & x & x & & \\ & & & x & x & \\ & & & & x & \end{bmatrix} \quad \text{et} \quad D_k = \begin{bmatrix} x & & & & & \\ & x & & & & \\ & & x & & & \\ & & & x & & \\ & & & & x & \end{bmatrix}.$$

On vérifie que

$$N_k = \begin{bmatrix} 1 & & & & & & & & & & \\ l^+(1) & 1 & & & & & & & & & \\ & \ddots & \ddots & & & & & & & & \\ & & l^+(k-2) & 1 & & & & & & & \\ & & & l^+(k-1) & 1 & u^-(k) & & & & & \\ & & & & & 1 & u^-(k+1) & & & & \\ & & & & & & \ddots & \ddots & & & \\ & & & & & & & \ddots & u^-(n-1) & \\ & & & & & & & & 1 & \end{bmatrix}$$

et

$$D_k = \mathrm{diag}(d^+(1), \ldots, d^+(k-1), \gamma_k, d^-(k+1), \ldots, d^-(n)).$$

5.2.3 Le coefficient γ_k

On rappelle le théorème suivant

Théorème 5.2.1 *Soit J une matrice tridiagonale complexe $n \times n$ admettant la double factorisation (5.4). Considérons la twisted factorisation au pivot k donnée par (5.5). Alors, pour $1 \leq k \leq n$,*

$$\gamma_k = d^+(k) + d^-(k) - J_{kk}$$

et si J est inversible

$$\frac{1}{\gamma_k} = e_k^* J^{-1} e_k.$$

Ce théorème découle de l'écriture de la twisted factorisation.

5.2.4 Connexion avec les vecteurs propres

On établit maintenant la connexion des parties précédentes avec le problème des vecteurs propres.

Soit λ une valeur propre calculée de la matrice J. On montre comment la double factorisation peut être utilisée pour trouver la "bonne" equation à omettre pour la résolution de $(J - \lambda I)x = 0$. Si on écrit $J = V \Lambda V^*$, on trouve

$$\begin{aligned} \frac{1}{\gamma_k} &= e_k^* V^* (\Lambda - \lambda I)^{-1} V e_k \\ \Rightarrow \quad \frac{1}{\gamma_k} &= \frac{|v_j(k)|^2}{\lambda_j - \lambda} + \sum_{i \neq j} \frac{|v_i(k)|^2}{\lambda_i - \lambda}. \end{aligned}$$

Donc, si λ est proche d'une valeur propre isolée λ_j, γ_k reflète la valeur de $v_j(k)$. On admet dès lors que choisir γ_k minimum revient à effectuer un bon choix d'indice pivot k. Pour la démonstration exacte, voir [15].

5.2.5 Calcul des vecteurs propres et orthogonalité

On s'intéresse maintenant au calcul du vecteur propre. On effectue pour ce faire une descente et une remontée de la matrice $A - \lambda I$. Le théorème suivant montre comment calculer le vecteur v qui satisfait toutes les équations de $(A - \lambda I)v = 0$ sauf la $k-$ème

Théorème 5.2.2 *Soit $J = A - \lambda I$ une matrice tridiagonale irréductible qui permet la twisted factorisation (5.5) pour un indice fixé k. Alors, le système*

$$(A - \lambda I)v^{(k)} = \gamma_k e_k$$

a une unique solution donnée par

$$\begin{array}{rcll} v^{(k)}(k) & = & 1, & \\ v^{(k)}(j) & = & -u^+_{j,j+1} v^{(k)}(j+1), & j = k-1, \ldots, 1, \\ v^{(k)}(i) & = & -l^-_{i,i-1} v^{(k)}(i-1), & i = k+1, \ldots, n. \end{array}$$

Preuve Comme $N_k e_k = e_k$ et $D_k e_k = \gamma_k$, on a

$$(A - \lambda I)v^{(k)} = \gamma_k e_k \Rightarrow N_k^T v^{(k)} = e_k$$

Le résultat est immédiat.

Un point fort de la méthode est d'obtenir des vecteurs propres directement orthogonaux. Dans les démonstrations, très techniques, on regarde entre autres les angles formés par les vecteurs. On a le théorème suivant.

Théorème 5.2.3 *Les vecteurs propres $(v_k)_{k=1,\ldots,n}$ obtenus par l'algorithme X associés aux valeurs propres (λ_k) vérifient*

$$|v_i^T v_j| \leq \frac{O(n\epsilon)}{reldist(\lambda_i, \lambda_j)} \text{ pour } i < j$$

où ϵ est la précision machine.

Pour plus de détails, voir [15].

5.2.6 Résultats numériques

On présente ici les résultats numériques de la méthode *LR* couplée à la méthode de calcul des vecteurs propres proposée par I. Dhillon. On effectuera une comparaison avec l'unique routine disponible (de la bibliothèque *LAPACK*) pour le calcul des matrices complexes non symétriques. Précisons que cette dernière est basée sur la méthode *QR* et la méthode de la puissance itérée et adaptée pour les matrices denses complexes quelconques. Les temps de calcul sont alors trop long pour être acceptable. Notre motivation première fut donc d'obtenir le spectre dans un temps raisonnable. Toutefois, la précision est aussi primordiale pour la Réduction Cyclique. On va montrer la performance impressionnante en termes de rapidité et de performance de la méthode *LR* au travers d'un cas de validation et d'un cas réaliste.

Remarque Les méthodes présentées jusqu'ici sont implémentées dans la bibliothèque *LAPACK* pour les matrices tridiagonales symétriques uniquement réelles. Après étude, l'implémentation dans le cas complexe a été alors effectuée sans de modification majeure.

Cas de validation

Notre cas test de validation sera la matrice complexe de taille $n \times n$

$$A = \begin{bmatrix} 1+\mathrm{i} & -1 & & \\ -1 & 2+\mathrm{i} & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 1+\mathrm{i} \end{bmatrix}.$$

Il est bien connu (voir annexes) que les valeurs propres sont

$$\lambda_i = 2 + \mathrm{i} - 2\cos\left(\frac{j\pi}{n+2}\right), \quad j = 1, \ldots, n.$$

On compare la méthode de *LAPACK* à la méthode *LR* avec le calcul des vecteurs propres adapté. Les résultats obtenus sont alors résumés dans le tableau suivant

	$\|\|\Lambda - \Lambda_e\|\|_\infty$	$\|\|Av_i - \lambda_i v_i\|\|_\infty$	$\|\|Q^TQ - I\|\|_\infty$	$\|\|A - Q^T\Lambda Q\|\|_\infty$	temps
$n = 1000$					
$LAPACK$	1.3×10^{-13}	1.4×10^{-14}	$5. \times 10^{-12}$	5.8×10^{-12}	$128s$
$LR + Dhillon$	$2. \times 10^{-14}$	8.3×10^{-17}	7.2×10^{-13}	9.2×10^{-14}	$2s$
$n = 5000$					
$LR + Dhillon$	5.3×10^{-13}	2.1×10^{-15}	3.2×10^{-11}	5.8×10^{-13}	$55s$

TAB. 5.1 – Comparaison des méthodes QR et LR

D'un point de vue qualitatif, les deux méthodes donnent des résultats très satisfaisants. On sait que la comparaison en temps ne serait pas appropriée mais il est impressionnant de constater la rapidité et la qualité de la méthode LR pour ces tailles de matrice. En particulier, la base des vecteurs propres est orthogonale à une précision remarquable pour $n = 5000$.

Cas réaliste

On présente maintenant des résultats pour notre matrice tridiagonale complexe symétrique non-hermitienne issue de la discrétisation d'une ligne de maillage du problème (2.3).

	$\|\|Av_i - \lambda_i v_i\|\|_\infty$	$\|\|Q^TQ - I\|\|_\infty$	$\|\|A - Q^T\Lambda Q\|\|_\infty$	temps
$n = 1000$				
$LAPACK$	1.4×10^{-13}	1.5×10^{-11}	1.8×10^{-11}	$159s$
$LR + Dhillon$	2.7×10^{-15}	1.8×10^{-14}	4.8×10^{-14}	$3s$
$n = 5000$				
$LR + Dhillon$	9.9×10^{-16}	9.5×10^{-14}	5.7×10^{-14}	$73s$

TAB. 5.2 – Comparaison des méthodes QR et LR dans un cas réaliste

Les résultats sont encore satisfaisants pour les deux méthodes bien que la méthode LR soit bien plus précise. Il est particulièrement intéressant de noter la précision de l'orthogonalité pour ce cas pratique réaliste.

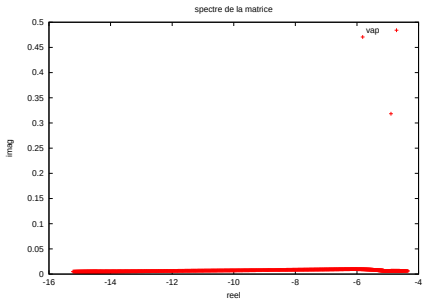


FIG. 5.1 – Spectre de la matrice A

Troisième partie

Parallélisme

6

Parallélisation de la méthode

Sommaire

On intègre un module dans le code parallèle *HERA* calculant l'hydrodynamique et le paraxial. Nos efforts seront centrés sur les méthodes de résolution du modèle Helmholtz, il s'avère que ce modèle est beaucoup plus contraignant que

les deux autres. Notre but est d'utiliser et d'adapter les techniques de pointe de parallélisation et d'optimisation à nos méthodes afin d'effectuer un calcul massivement parallèle sur un domaine réaliste avec couplage entre les modèles paraxial, Helmholtz et hydrodynamique. Dans cette partie, on présente la stratégie de parallélisme utilisée pour les méthodes présentées jusqu'ici, de la décomposition de domaine à la Réduction Cyclique pour la résolution de l'équation (1.7) sur un domaine de simulation de taille réaliste. On montrera en particulier que pour obtenir de bonnes performances, l'aspect informatique scientifique a une place cruciale. En effet, on cherche à exploiter au mieux l'architecture des machines disponibles. La décomposition de domaine, la répartition des charges ainsi que les protocoles de communication sont les points fondamentaux de notre étude. En particulier, on présentera une stratégie de communication dite *hybride* particulièrement adaptée à la structure du calculateur. On présentera également les grands axes d'optimisation de code.

6.1 Le supercalculateur *Tera-1*

Les supercalculateurs peuvent être rangés en trois catégories suivantes :

- Les calculateurs à mémoire partagée (shared memory, fig.6.1) sont composés de processeurs ayant un accès égal à la mémoire centrale. Ainsi, les processeurs peuvent dialoguer de manière simple entre eux en partageant les données et en utilisant les adresses connues.
- Les calculateurs à mémoire distribuée (distributed memory / message passing systems, fig.6.2) sont constitués de processeurs ayant une mémoire interne et communiquant par échanges de messages. Ces machines sont réparties en deux classes ; celles dont le contrôle est synchrone (SIMD, Single Instruction Multiple Data) et celles dont le contrôle est asynchrone (MIMD, Multiple Instruction Multiple Data).
- Les « clusters de SMP » (Symetric MultiProcessors = multiprocesseurs à mémoire distribuée, fig.6.3) sont des machines distribuées en *noeuds* ayant leur mémoire interne. Chaque noeud est composé de processeurs qui se partagent la mémoire interne. Ceux sont donc des machines mixtes par rapport aux deux types précédents.

L'architecture retenue pour les supercalculateurs du projet *Tera* est du type SMP. Bénéficiant d'un fort soutien de l'industrie informatique, sa pérennité devrait permettre de conserver la même architecture pendant toute la durée du projet.

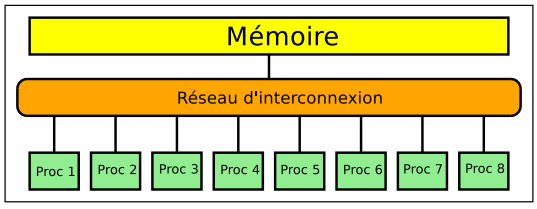


FIG. 6.1 – Machine à mémoire partagée

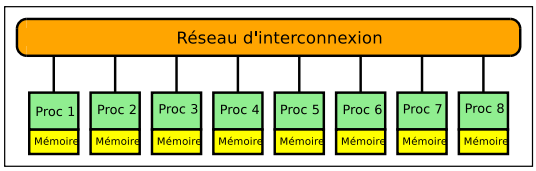


FIG. 6.2 – Machine à mémoire distribuée

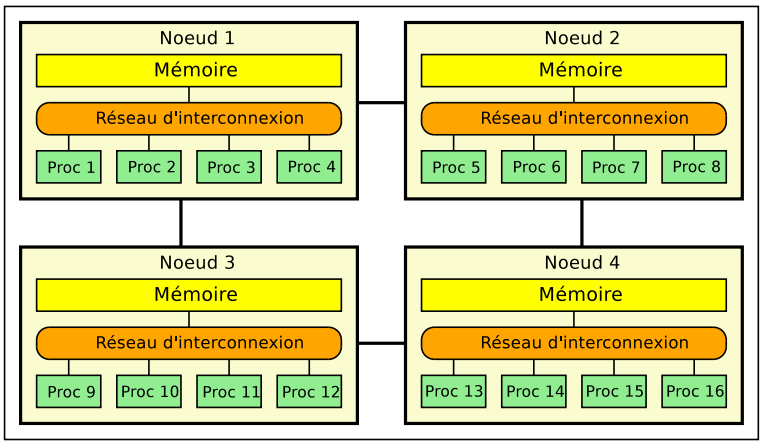


FIG. 6.3 – Machine type SMP

Installée en décembre 2001 au CEA-DAM-Ile-de-France, et fournie par la société *Compaq*, la machine *Tera-1* a une puissance crête totale de 5 *téraflops* (mille milliards d'opérations en virgule flottante). Sa puissance soutenue est de 1 téraflops. Le *noeud*, brique de base de *Tera-1*, est une mini-ordinateur à quatre processeurs *Alpha EV68* à 1 *gigahertz*. Ils partagent une mémoire de 4 gigaoctets et fournissent une puissance totale de 8 gigaflops. Par sa bande passante mémoire importante, ce mini-ordinateur a de bonnes performances pour le calcul scientifique. Les 640 noeuds de base (2560 processeurs au total) sont interconnectés par un réseau rapide conçu par la société *Quadrics* et intégré par Compaq. L'architecture du réseau d'interconnexion comporte deux étages. Pour de raisons de performance et de tolérance aux pannes, il y a deux réseaux indépendants identiques dans la machine, chaque noeud étant connecté aux deux réseaux. Pour les *entrées-sorties*, le système de fichiers global offre un espace de stockage de 50 téraoctets, avec une bande passante totale de 7,5 gigaoctets par seconde.

6.2 Le code *HERA*

Développé au sein du CEA/SNEC, le code *HERA* [19] est une plateforme de calcul élaborée permettant la simulation parallèle de problèmes physiques variés tels que la détonique, l'hydrodynamique compressible, les instabilités multifluides ou encore la propagation des ondes en régime paraxial.

Le code *HERA* est basée sur la méthode *AMR* (Adaptive Mesh Refinement). Le coût et la précision d'une simulation est contrôlée par le raffinement dynamique du maillage au cours des itérations temporelles à partir d'un maillage initial cartésien. Une telle approche en parallèle occasionne des problèmes délicats pour l'équilibrage de charges sur chaque processeur.

Une approche *SPMD* (Single Program Multiple Data) a été retenue. Plusieurs tâches exécutent le même programme sur des processeurs différents et des données différentes. Pour le bon déroulement et la cohérence des résultats, les différentes tâches mises en contribution en parallèle communiquent par le réseau d'interconnexion *inter-noeud* ou, le cas échéant, par la mémoire locale à chaque noeud.

Enfin, le code *HERA* est programmé en langage *C++* [44]. Très populaire dans l'industrie informatique, le *C++* s'avère aussi un outil puissant pour le calcul scientifique. Ce langage multiparadigme issu du *C* permet la programmation procédurale, la programmation orientée objet et la programmation générique. La manipulation d'objets et concepts alliés à des fonctionnalités d'encapsulation, héritage, polymorphisme et généricité permet d'avoir une structure de code claire, maintenable et souple. En contre partie, l'effort de programmation pour l'efficacité s'en trouve accrue.

Remarque Dans notre cas, la méthode *AMR* s'applique au maillage *grossier* fluide pour l'hydrodynamique et le paraxial. Le modèle Helmholtz est lui résolu sur le maillage *fin* qui en général est 10 fois plus fin que le grossier. La stratégie de solveur rapide ne convient pas avec les techniques *AMR*. On a en effet besoin d'une régularité sur le maillage. Le coût de la méthode étant principalement dû au modèle Helmholtz et par soucis de simplicité, on choisit de ne pas utiliser l'*AMR* dans notre implémentation.

6.3 Stratégie de parallélisation

On distingue deux modèles de programmation des machines SMP :

- Le parallélisme de données consiste à effectuer des actions successives sur des données parallèles. On s'intéresse au cas où le découpage des séquences est effectué au niveau du code. Il s'agit de programmes de type SPMD. Ce type de parallélisme est guidé par la distribution des données.
- Le parallélisme de tâche consiste à effectuer différentes tâches par plusieurs *processus* de manière asynchrone. Plusieurs processus peuvent être exécutés sur un même processeur. Dans ce cas, le parallélisme est alors guidé par l'allocation des tâches aux processeurs.

Le premier modèle constitue une parallélisation naturelle. On répartit les données du problème de manière équilibrée entre les processeurs. On précise que sans stratégie *AMR*, les calculs restent équilibrés au cours du temps. Le second modèle est directement lié à une approche dite par *multithreading*. On répartit alors les tâches sur les processeurs. On peut aussi coupler les deux modèles.

Tous ces modèles reposent sur des moyens de communication entre les différentes ressources disponibles. On s'intéresse à trois approches possibles pour les

échanges de messages entre les différentes tâches fortement liées aux modèles de parallélisation.

6.3.1 Première approche, *MPI*

L'approche classique en calcul scientifique repose sur l'utilisation de la bibliothèque *MPI* (*Message Passing Interface*). C'est une bibliothèque de procédures de communication et de synchronisation qui à l'heure actuelle est la référence en la matière et l'unique outil standard portable. C'est un mode de communication à deux interfaces utilisant le réseau, c'est-à-dire que deux processeurs doivent être actifs pour communiquer. Un processeur envoie un message via le réseau et un autre le reçoit. *MPI* n'assure pas de contrôle de flux, c'est à l'utilisateur de s'assurer que le processus est prêt à recevoir. L'avantage de *MPI* est qu'il n'y a pas de stockage intermédiaire d'où des communications plus efficaces. Pondérons ceci par le fait que ce n'est pas garanti par tous les systèmes.

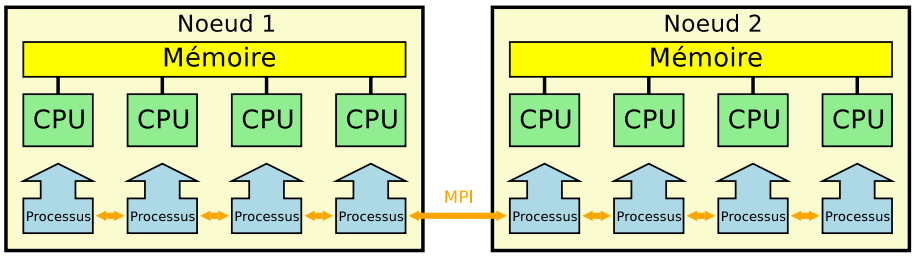


Fig. 6.4 – Approche *MPI*

En général, les communications par le réseau consomment peu de ressource de type processeur mais relativement beaucoup de temps en raison des protocoles, latences et débits du réseau. Sur la machine *Tera*, ceci est géré de manière ultra-performante par le réseau *Quadrics* dont le *MPI* propriétaire Compaq forme une interface. Toutefois, cela peut être très pénalisant et nécessite une attention particulière. Il existe deux types de communication

- *bloquante* : les requêtes de communication se terminent lorsque la communication est effectivement accomplie ;
- *non bloquante* : les requêtes se terminent immédiatement et les communications s'accomplissent en tâche de fond, nécessitant alors des tests de complétude.

Ainsi, nous utiliserons quand cela est possible des communications *non bloquantes*, c'est-à-dire que nous recouvrerons les communications par des calculs. Pour les programmes type SPMD, cela se réduit à un flot d'instruction calcul entrecoupé de requêtes non bloquantes et de point de contrôle validant la fin de ces dernières.

6.3.2 Deuxième approche, Multithreading

Le *multithreading* autorise la programmation événementielle multi-tâche au sein d'un noeud.

Un *thread* (ou processus léger) est un flux d'exécution indépendant au sein d'un processus. Tous les threads issus d'un processus *parent* partagent le même contexte d'exécution (en particulier l'espace mémoire). C'est une propriété fondamentale très utile en calcul scientifique. L'*ordonnancement* des threads, qui est la stratégie de répartition sur les ressources type processeurs, est du ressort du système. Cette politique dite de *scheduling* est alors déterminante pour les applications utilisant le *multithreading*.

Le calcul est mené sur un nœud multiprocesseur et n'utilise qu'un seul processus. Dans le cadre d'une décomposition de domaine, chaque sous-domaine est calculé simultanément par un thread. Le système d'exploitation distribue les threads sur les différents processeurs du noeud à partir du processus parent et les communications se font en mémoire partagée, tout en restant à la charge de l'application. La bibliothèque *POSIX pthread* permet ce type de programmation sur toute machine. Les algorithmes d'ordonnancement et de migration des threads semblent parfois inadaptés et peuvent entraîner une mauvaise utilisation des processeurs. Par exemple, créer quatre threads sur un noeud ne garantie pas d'utiliser les quatre processeurs. La pratique montre qu'il peut falloir utiliser un certain nombre de threads pour obtenir une efficacité optimale. Ceci est principalement dû au fait que les *schedulers* (ordonnanceurs) sont adaptés à des besoins type système d'exploitation ou serveur mais pas spécifiquement au calcul scientifique ; la réactivité est privilégiée à la performance. Notons certains travaux d'exploration dans ces domaines [45], [46].

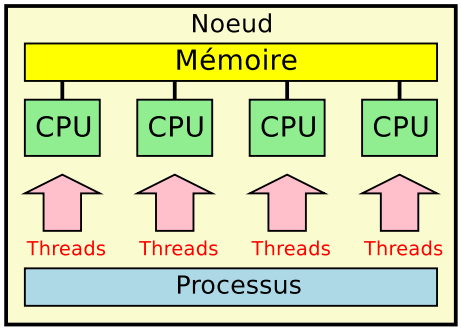


FIG. 6.5 – Approche *multithreading*

Le premier intérêt de cette approche est d'obtenir des communications efficaces entre processeurs d'un même noeud en utilisant le fait que les communications par le réseau sont beaucoup plus lentes qu'un accès à la mémoire locale de type RAM. Dans notre cas, la grande performance du réseau *Quadrics* limite grandement cet intérêt. Toutefois, l'intérêt majeur pour nous est que la mémoire d'un processus *parent* est accessible par les *threads* qu'il engendre permettant un minimum de redondance des données au sein d'un même noeud. C'est le principe de la parallélisation de tâches mais aussi cette économie de mémoire qui nous intéressent.

Remarque Par la suite, nous ferons la distinction entre le *processeur*, unité physique de calcul, et le *processus*, programme en exécution allouant des res-

sources de type *processeur*. Un *processeur* peut gérer de manière concurrentes plusieurs *processus* par une méthode de partage du temps. Un *processus* ne peut être partagé par un autre *processus* à moins qu'il soit *multithreadé*.

Remarque Il existe un outil haut niveau de parallélisation basé sur le *multithreading* très utilisé en calcul scientifique appelé *OpenMP*. Des directives compilateurs *#pragma* sont placées dans le code pour définir des zones parallèles. Bien qu'adapté au parallélisme de données, on préfère à *OpenMP* la bibliothèque *pthread* adaptée au parallélisme de tâche malgré un code engendré plus hermétique et plus portable sur monoprocesseur.

6.3.3 Troisième approche, Méthode mixte hybride

Le *multithreading* permet une forme de communication et parallélisation intranoeud efficace. Un manière de généraliser le *multithreading* consiste à considérer une troisième approche dite *hybride*, combinaison des deux premières. Chaque processus *MPI* est associé à un unique processeur par noeud et gère plusieurs threads, donc plusieurs sous-domaines ou tâches à effectuer en parallèle. S'il y a besoin de communication au sein d'un même processus, l'échange de messages se fait par simple copie mémoire, sinon l'échange se fait par requête *MPI*.

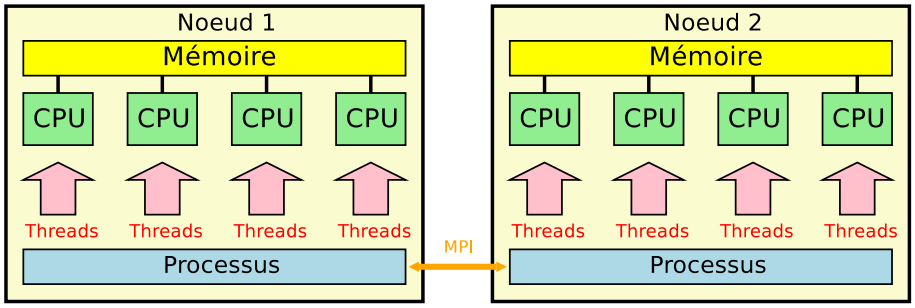


FIG. 6.6 – Approche hybride *MPI/multithreading*

6.4 Répartition des données

La distribution des données est dictée par celle effectuée dans le code *HERA*. Le maillage *grossier* fluide est ainsi distribué en bandes horizontales de manière équilibrée entre chaque processeur. Une telle décomposition en bandes permet une parallélisation efficace du schéma numérique utilisé pour le modèle paraxial qui est une marche en espace. Chaque sous-domaine est constitué d'un maillage avec un recouvrement d'une maille entre sous-domaines que l'on appelle *maille fantôme*. Ceci permet en particulier d'appliquer les conditions aux limites du domaine du calcul et de recevoir les informations provenant des sous-domaines voisins.

On répartit globalement le maillage *fin* Helmholtz par bandes horizontales pour coïncider avec le maillage *grossier* fluide. On rappelle que le domaine $\Omega \subset \mathcal{D}$ est le domaine dans lequel l'approximation paraxiale (1.18) n'est plus valide. On résout le problème (2.3) en effectuant la décomposition de domaine (3.2). Le domaine Ω est considéré rectangulaire. On décompose alors Ω en trois sous-domaines se recouvrant, Ω_b et Ω_h contenant les couches *PML* et Ω_g. Pour le domaine Ω_g, on note n_x et n_y les nombres de points de discrétisation respectifs d'une ligne et d'une colonne. Le recouvrement n_r est de l'ordre de quelques lignes de maillage. Les domaines Ω_b et Ω_h contiennent typiquement quelques dizaines de lignes de maillage. On note leur épaisseur n_y^p.

On suppose disposer de n_p processus *MPI*. Le domaine Ω est subdivisé en n_p sous-domaines contenant $(n_x \times (n_y + 2n_y^p))/n_p$ points du maillage. On équilibre ainsi les charges pour chaque processus (voir fig.6.7). Notons que le premier et le dernier processeur contiennent à la fois les couches *PML* et quelques lignes de la Réduction Cyclique. Le vecteur second membre est naturellement découpé en n_p morceaux respectivement liés aux lignes du maillage.

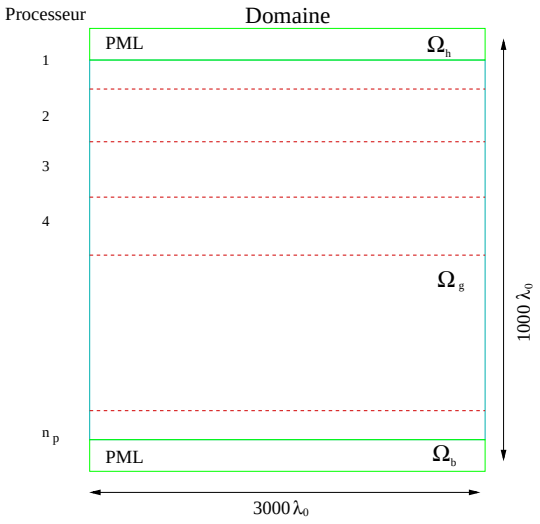


FIG. 6.7 – Répartition des données

6.5 Analyse préliminaire du coût de stockage

Le domaine Ω réaliste est une boîte d'une dimension de $1000\lambda_0$ par $3000\lambda_0$. Avec le critère de 10 points par longueur d'onde, on est amené à considérer un maillage de 300 millions de points. On a $n_x = 10000$ et $n_y = 30000$.

On présente le coût de stockage de la méthode de décomposition de domaine associée à une méthode de Krylov type *GMRES*. Pour la résolution des systèmes linéaires, un solveur direct LU est utilisé pour les domaines Ω_b et Ω_h contenant les couches *PML* et le solveur rapide Réduction Cyclique pour le domaine Ω_g.

6.5.1 Vecteurs

La solution de notre problème est l'enveloppe complexe du laser. D'un point de vue informatique, on suppose qu'un nombre complexe est codé sur 16 octets. Ainsi, on constate que pour un vecteur défini sur tout le domaine, on a besoin d'environ $4,8$ gigaoctets (Go) de mémoire.

Dans la méthode *GMRES*, on stocke les directions de descente à chaque itération. On a besoin du vecteur résidu r_0 et de la base $(v_i)_{1\leq i\leq m}$, où m est le nombre d'itérations de la méthode. Il est maintenant évident qu'il est nécessaire de converger rapidement pour ne pas sortir de l'espace mémoire disponible. Par exemple, converger en 10 itérations oblige à stocker près de 50 Go d'information sur tous les processeurs.

6.5.2 Matrices

Pour la méthode de Réduction Cyclique appliquée sur le domaine Ω_g, on stocke les matrices diagonales de récurrence $\Lambda^{(r)}$, $\Gamma^{(r)}$ et $\Pi^{(r)}$ ainsi que la matrice normale Q composée des vecteurs propres. Pour un nombre de ligne n_y de 30000, la méthode effectue 15 étapes de réduction. On doit alors stocker $2,4$ Mo pour les matrices de récurrence. La base des vecteurs propres Q est une matrice dense de taille $n_x \times n_x$, soit $1,6$ Go de mémoire. On doit effectuer le produit matrice-matrice de Q (et Q^T) par la matrice $n_x \times n_y$ issue du second membre.

Remarque On comprend bien qu'on ne peut recopier cette matrice sur chaque processeur pour des raisons de limite d'espace mémoire. Une approche naturelle est de la distribuer. L'intérêt du *multithreading* est d'avoir la matrice sur la mémoire du noeud et de faire les produits matrice-vecteur par bloc répartis par thread.

Le stockage de la décomposition LU des matrices des couches *PML* peut être minimiser en ordonnant la matrice suivant la plus petite largeur de bande, c'est-à-dire quelques dizaines de lignes de maillage. Utilisant la bibliothèque native *LAPACK* pour la décomposition LU, le stockage est de type *band storage*. Cela nécessite alors environ 7 Mo de mémoire.

6.5.3 Bilan de la gestion mémoire

Voici le bilan mémoire des principales variables du problème Helmholtz que l'on peut dresser

Type	Coût
Second membre	$4,8$Go
Base$Q$	$1,6$Go
Matrices$\Lambda^{(r)}, \Gamma^{(r)}, \Pi^{(r)}$	$2,4$Mo
Résidur_0	$4,8$Go
Base de Krylov	$m \times 4,8$Go
DécompositionLU	7Mo
Bilan pour$m = 10$	$59,2$Go

A ce bilan s'ajoute également les données nécessaires à l'hydrodynamique et au paraxial. Définies sur le maillage *grossier* fluide, le stockage de ces données est toutefois bien moindre. Il faut également ajouter le mémoire consommée par le système.

Remarque L'opération coûteuse de la résolution est les m produits matrice-matrice de Q et Q^T par la matrice issue du second membre dans le solveur rapide. Le second membre est distribué naturellement suivant le maillage. D'un point de vue stratégique, on va montrer qu'il est primordial que chaque processus dispose de la matrice Q. Les séquences de produit matrice-matrice n'auront alors pas besoin de synchronisation. C'est dans ce contexte que la stratégie hybride *MPI-multithreading* prendra toute son importance.

D'après tout ceci, résoudre ce problème réaliste nécessite environ 200 processeurs.

6.6 Du bon usage du multithreading

On a déjà vu les multiples intérêts de la programmation par *multithreading*. Notre but est maintenant de quantifier réellement ceux-ci et de mettre en évidence les difficultés d'utilisation sur des architectures *alpha EV68* .

Mis à part la décomposition de domaine, notre motivation d'utilisation du *multithreading* fut en premier lieu le partage possible des données afin d'effectuer localement des opérations algébriques simples. En particulier, on s'intéresse à la combinaison linéaire de vecteurs et au produit matrice-matrice. On souhaite regarder de près les performances de la parallélisation de ces opérations sur un noeud par *multithreading*.

On définit alors l'efficacité E d'une parallélisation sur un noeud de quatre processeurs par

$$E = \frac{t_s}{4t_p}$$

où t_s est le temps séquentiel et t_p le temps parallèle de calcul. En pratique, on souhaite obtenir E proche de un.

Remarque D'un point de vue informatique, l'utilisation du C++ permet une encapsulation de la bibliothèque *POSIX pthread* rendant la programmation *multithreading* simple et accessible (voir annexes). En particulier, le passage d'argument et de fonction aux threads en devient naturelle.

6.6.1 Combinaison linéaire de vecteurs

Regardons l'opération représentative $y = \alpha x + y$ ou x et y sont deux vecteurs de taille n. Outre le produit matrice-vecteur, ce type d'opération est le coeur des différentes séquences des méthodes de Krylov. D'un point de vue pratique, la parallélisation de cette opération (appelée généralement *axpy*) ne nécessite aucune communication entre les processus (et threads). Le principe général est de subdiviser les différents vecteurs en N_b bloc afin de créer une pile de tâches à

transmettre à N_p threads. Ainsi, chaque thread effectue une partie de l'opération *axpy* puis, lorsqu'il termine, va rechercher une nouvelle tâche dans la pile jusqu'à ce qu'elle soit vide. Ce principe de pile pouvant compter un nombre de tâche supérieur au nombre de threads permet de découpler les effets bloc optimal de thread optimal. Cela nécessite toutefois une programmation attentive quant à la gestion de la pile et l'attribution des tâches aux threads. La recherche de ces paramètres optimaux est une étape intéressante de l'étude. Remarquons d'abord que l'opération *axpy* est très simple et bénéficie efficacement des optimisations des compilateurs tels que par exemple le déroulement de boucle. Ainsi, la taille des vecteurs devient un paramètre crucial de la parallélisation. En effet, sur des vecteurs de « petites » tailles, de l'ordre de 10^5 composantes (stockage inférieur à 1 Mo en double précision), la parallélisation est désastreuse, on atteint au mieux une efficacité de 20% pour une opération de 5 ms.

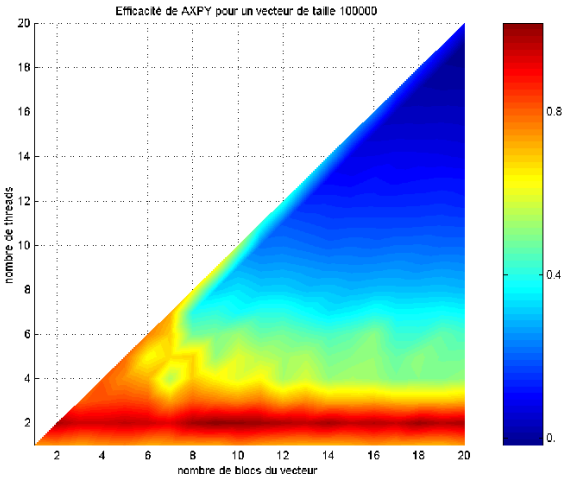


FIG. 6.8 – Efficacité pour $n = 10^5$

On constate même que pour des mauvais choix de paramètre, le temps parallèle peut être le double du temps séquentiel. Pour expliquer ceci, il est bon se rappeler que la taille du cache des processeurs *alpha* est de 8 Mo. Les vecteurs de cette taille tiennent alors complètement en mémoire cache et les opérations sont alors effectuées très rapidement sur un unique processeur. Ensuite, les temps de créations et de coordination des threads ne sont pas négligeables comparés aux temps de calcul. Pondérons tout ceci par l'intérêt inexistant de paralléliser une opération aussi courte. On montre ainsi que le *multithreading* n'est pas un procédé de parallélisation aussi générique que *MPI*.

Pour des vecteurs de taille de l'ordre de plusieurs millions de composantes (stockage sur plusieurs dizaines de Mo en double précision), les performances deviennent meilleures bien que non réellement intéressantes. Ceci s'explique par le fait que les temps de gestion système des threads ne sont pas encore négligeable. Pour $n = 10^6$, l'efficacité optimale est de 30% pour une durée calcul de 85 ms.

Avec des tailles de vecteur de l'ordre de $n = 10^7$ (80 Mo en double précision),

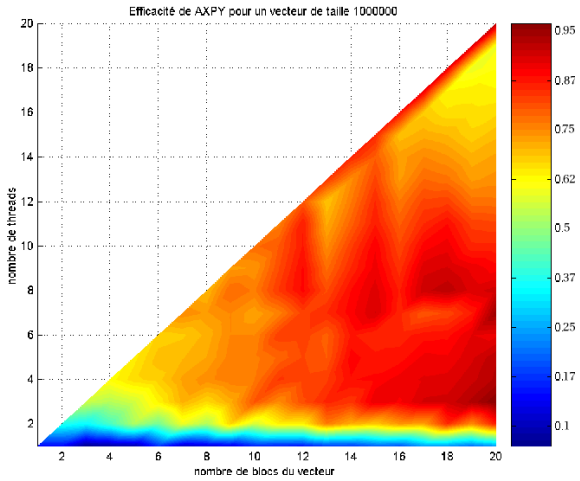


FIG. 6.9 – Efficacité pour $n = 10^6$

l'opération s'effectue en 0,22 s et la gestion des threads devient transparente. Les tâches étant de taille beaucoup plus importante, on obtient des résultats

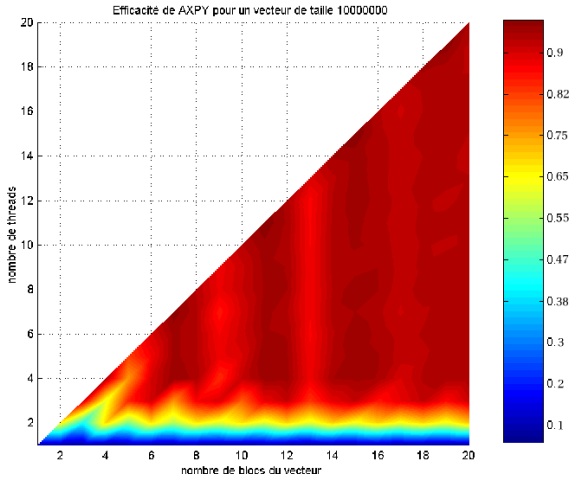


FIG. 6.10 – Efficacité pour $n = 10^7$

d'efficacité de 98% ce qui est très bon. De plus, mis à part quand on utilise trop peu de threads, le nombre de blocs ou threads n'importe plus vraiment pour avoir des performances supérieures à 90%.

Une dernière chose importante est qu'il est possible d'observer des effets *cache*, c'est-à-dire qu'on obtient une efficacité supérieure à un avec ici 104%. En fait, il existe des tailles de blocs pour lesquelles la gestion du cache en chargement et utilisation devient optimale et les calculs en deviennent plus rapides. Il n'y a malheureusement pas de critère à cela.

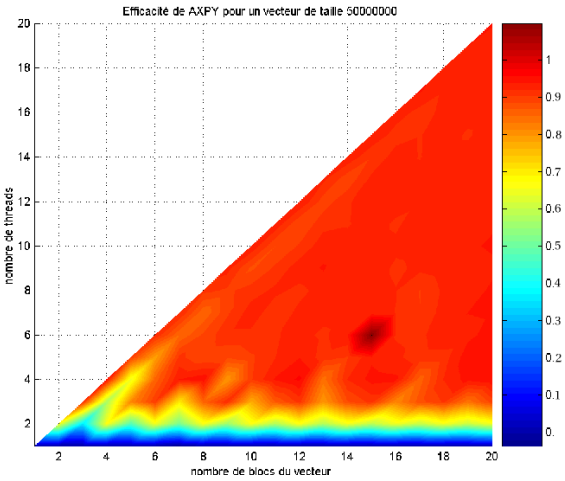


FIG. 6.11 – Efficacité pour $n = 5 \times 10^7$

6.6.2 Produit matrice-matrice

D'après ce qui précède, il faut des tâches nécessitant un minimum de travail pour avoir une parallélisation efficace. L'opération du produit matrice-matrice est suffisamment coûteuse pour que la programmation par *multithreading* donne de bons résultats.

Certaines précautions sont à prendre pour le découpage des matrices. L'idée est d'éviter que plusieurs threads aient à écrire aux mêmes endroits afin de ne pas générer d'attente lors des recopies des sous-matrices. Par exemple, pour le produit $Q \times B$, le découpage en $N_q \times N_b$ blocs est le suivant

$$\begin{bmatrix} Q_1 \\ \vdots \\ Q_{N_q} \end{bmatrix} \times \begin{bmatrix} B_1 & \dots & B_{N_b} \end{bmatrix} = \begin{bmatrix} Q_1B_1 & \dots & Q_1B_{N_b} \\ \vdots & \ddots & \vdots \\ Q_{N_q}B_1 & \dots & Q_{N_q}B_{N_b} \end{bmatrix}.$$

Dans ce cas, une tâche est un produit $Q_i \times B_i$ qu'on effectue par un appel à une routine de la bibliothèque *BLAS*. On ne considère dans la suite que l'exemple où les matrices Q et B sont de taille 3000×3000. Naturellement, on n'utilisera pas plus de threads que les $N_q \times N_b$ tâches à effectuer. Sur nos graphiques, en abscisse, on représente le nombre de threads et en ordonnée l'efficacité du produit. Les différentes courbes indiquent le nombre de blocs pour la matrice B. On présente les résultats obtenus pour un découpage de la matrice Q en 5 blocs.

Les résultats obtenus sont très satisfaisants. Comme pour le cas précédent, la taille des matrices est encore cruciale. Pour des grandes tailles de matrice, on est au dessus de 95% d'efficacité optimale. En particulier, nos résultats montrent que quelque soit les découpages, on peut toujours atteindre l'efficacité optimale avec 4 threads.

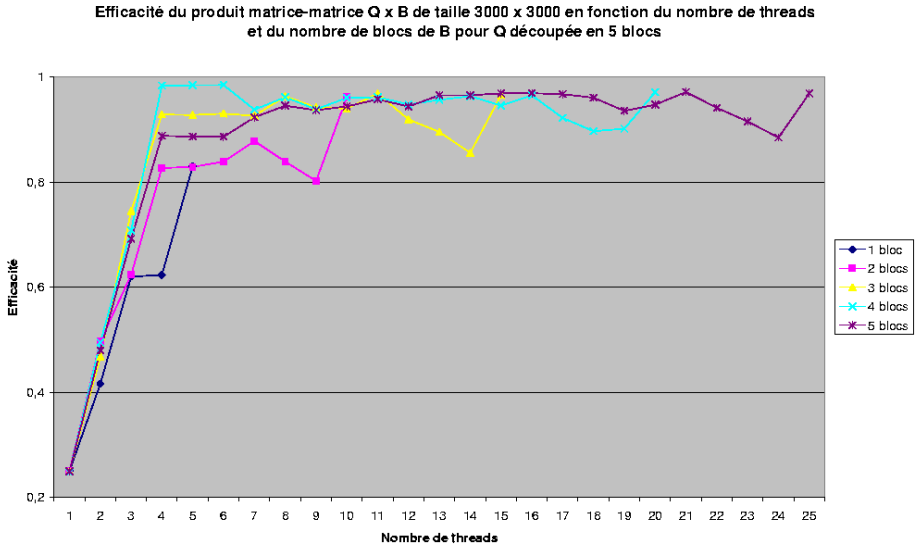


FIG. 6.12 – Efficacité suivant le découpage de la matrice Q

6.6.3 Conclusion

En raison du réseau haute performance *Quadrics*, utiliser le *multithreading* pour des gains sur les communications intranoeud est difficile. Dans la mesure où l'on souhaite plutôt bénéficier de la localité des données, on se doit d'avoir une grande efficacité dans la parallélisation. Une bonne gestion du cache devient alors primordiale mais difficilement controlable. En particulier, le caractère très mobile des threads sur les processeurs du noeud invalide souvent les données du cache. Pour pallier ces effets et ceux des temps de gestion ds threads par le système, il faut avoir des tâches particulièrment conséquentes à paralléliser. Dès lors, le *multithreading* ne s'appliquera pas dans toutes les opérations par la suite, notamment celles de petites tailles.

6.7 Parallélisation de la méthode *GMRES*

On présente maintenant notre stratégie de parallélisation de la méthode *GMRES* (voir page 51). On suppose exécuter un processus *MPI* par noeud et utiliser les techniques de *multithreading* afin d'utiliser toutes les ressources du noeud.

Les vecteurs b, $x^{(0)}$, x, r et $v^{(i)}$ sont distribués sur tous les processus. Les données $h_{i,j}$, z_i, c_i, s_i sont stockées de manière redondante sur tous les processeurs.

L'algorithme *GMRES* est composé d'une suite d'opérations algébriques que l'on parallélise par la méthode *hybride*. Par exemple, les opérations du type *axpy* sont effectuées localement par *multithreading*. Par contre les opérations type produit scalaire nécessite en plus une communication entre tous les processus *MPI*.

A chaque itération de la méthode *GMRES*, on a besoin d'effectuer un produit matrice-vecteur

$$y = \left(I - A_D^{-1}\left(A_E - \delta_n\right)\right) x$$

où on rappelle que

$$A_D = \begin{pmatrix} A_H & 0 & 0 \\ 0 & A_I & 0 \\ 0 & 0 & A_B \end{pmatrix} \qquad \text{et} \qquad A_E = \begin{pmatrix} 0 & C_1 & 0 \\ C_2 & 0 & C_3 \\ 0 & C_4 & 0 \end{pmatrix}.$$

A chaque itération, on a donc besoin d'inverser les matrices A_B, A_H et A_I associées aux domaines respectifs Ω_h, Ω_b et Ω_g. Pour le domaine central A_I, on utilise la Réduction Cyclique dont on présentera la parallélisation. La majeure partie des ressources sera d'ailleurs utilisée pour le traitement de ce bloc central. Autrement, on utilise une décomposition LU des matrices A_B et A_H à partir de bibliothèques scientifiques connues. A chaque itération de la méthode de Krylov, on effectue alors une simple résolution de deux systèmes triangulaires simultanément à la résolution du bloc central par la Réduction Cyclique. Notons que le temps de la décomposition LU est largement inférieur au temps de traitement de A_I.

On rappelle que les premier et dernier processeurs contiennent à la fois les lignes de maillage des couches *PML* et des lignes à traiter par Réduction Cyclique. Pour la résolution parallèle du système linéaire $A_D\tilde{y} = \tilde{x}$, on répartit alors la résolution des systèmes $A_I\tilde{y}_I = \tilde{x}_I$ et $A_B\tilde{y}_B = \tilde{x}_B$ par *multithreading*.

Algorithme 4 Produit Matrice-vecteur parallèle

Entrée: x, y, A_D, A_E, δ_n
Sortie: $y = \left(I - A_D^{-1}\left(A_E - \delta_n\right)\right) x$

```
x̃ = (A_E − δ_n) x
Création des threads
  si on est sur le premier processeur alors
    thread
      ỹ_B = A_B^{-1} x̃_B {LU}
    fin thread
  fin si
  si on est sur le dernier processeur alors
    thread
      ỹ_H = A_H^{-1} x̃_H {LU}
    fin thread
  fin si
fin Création des threads { Et lancement parallèle }
ỹ_I = A_I^{-1} x̃_I { Réduction Cyclique parallèle }
Attente des threads
y = x − ỹ
retourner y
```

6.7.1 Algorithme *GMRES*

Dans l'algorithme (6.7.1) présentant les différentes étapes de *GMRES*, on a indicé par

- ⋆ les opérations complètement locales ;
- ⋆ les opérations algébriques locales (type *axpy*) sur des données distribuées ;
- ⋆ les opérations algébriques (type produit scalaire) sur les données distribuées nécessitant une communication ;
- ⋆ les produits matrice-vecteur.

Algorithme 5 Algorithme *GMRES* parallèle

Entrée: $x^{(0)}$, ϵ, b, A_D, A_E, δ_n
Sortie: $x = \left(I - A_D^{-1}\left(A_E - \delta_n\right)\right) b$

⋆ $\tilde{b} = A_D^{-1} b$, $\tilde{r} = \left(I - A_D^{-1}\left(A_E - \delta_n\right)\right) x^{(0)}$
⋆ $r = \tilde{b} - \tilde{r}$
⋆ $z_1 = \|r\|_2$
⋆ $v^{(1)} = \dfrac{r}{z_1}$
pour $i = 1, 2, \ldots$ **faire**
  ⋆ $\omega = \left(I - A_D^{-1}\left(A_E - \delta_n\right)\right) v^{(i)}$
  pour $k = 1, \ldots, i$ **faire**
    ⋆ $h_{k,i} = \left(\omega, v^{(i)}\right)$
    ⋆ $\omega = \omega - h_{k,i} v^{(k)}$
  fin pour
  ⋆ $h_{i+1,i} = \|\omega\|_2$
  ⋆ $v^{(i+1)} = \omega / h_{i+1,i}$
  pour $k = 1, \ldots, i$ **faire**
    ⋆ $t = h_{k,i}$, $h_{k,i} = \overline{c_k} t + s_k h_{k+1,i}$, $h_{k+1,i} = -s_k t + c_k h_{k+1,i}$
  fin pour
  ⋆ $\alpha = \sqrt{|h_{i,i}|^2 + |h_{i+1,i}|^2}$, $s_i = \dfrac{h_{i+1,i}}{\alpha}$, $c_i = \dfrac{h_{i,i}}{\alpha}$
  ⋆ $z_{i+1} = -s_i z_i$, $z_i = \overline{c_i} z_i$, $h_{i,i} = \overline{c_i} h_{i,i} + s_i h_{i+1,i}$, $h_{i+1,i} = 0$
  si $|z_{i+1}| < \epsilon$ **alors**
    pour $k = i-1, i-2, \ldots, 1$ **faire**
      pour $j = k+1, \ldots, i$ **faire**
        ⋆ $z_k = z_k + h_{k,j} z_j$
      fin pour
      ⋆ $z_k = z_k / h_{k,k}$
    fin pour
    ⋆ $x = x^{(0)}$
    pour $k = 1, \ldots, i$ **faire**
      ⋆ $x = x + y_k v^{(k)}$
    fin pour
    retourner x
  fin si
fin pour

6.8 Parallélisation de la Réduction Cyclique

Au cours de la réduction, les processus éliminent au fur et à mesure les inconnues qu'ils contiennent jusqu'à ce qu'il ne reste plus qu'un seul processus contenant le problème réduit (4.11). On voit ici le paradoxe de la parallélisation de méthodes de réduction. Dès lors, on joue sur deux tableaux en parallélisant à la fois les étapes de la méthode de Réduction Cyclique et les opérations effectuées durant ces étapes. La très grande taille du domaine associée au fait d'utiliser une méthode itérative où l'on doit stocker les directions de descente sont des contraintes très fortes à gérer simultanément.

On présente dans cette section la parallélisation de la méthode de Réduction Cyclique. On va notamment voir que les performances reposent principalement sur la parallélisation de l'opération la plus coûteuse, le produit matrice-matrice. A la parallélisation de données, le *multithreading* propose une alternative efficace à la réduction des communications et à la gestion mémoire.

6.8.1 Calcul du spectre

La première étape de la méthode est le calcul des valeurs et vecteurs propres de la matrice A issue de (4.1). C'est une étape de préparation de la Réduction Cyclique coûteuse.

La méthode de calcul des valeurs propres basée sur des décompositions LU successives est intrinsèquement séquentielle (voir page 73). Tous les processus calculent les valeurs propres Λ_i de manière redondante. Par contre, le calcul d'un vecteur propre Q_i associé à la valeur propre Λ_i peut être fait de façon indépendante avec l'Algorithme X' (voir page 76). On peut alors répartir les tâches que sont le calcul d'un vecteur propre à plusieurs threads. On gagne au moins un facteur deux en parallélisant par *multithreading*.

6.8.2 Récurrence sur le spectre

La deuxième partie de la préparation de la Réduction Cyclique est les récurrences sur le spectre (voir page 61). Notons que cette séquence est très peu coûteuse notamment parce que les opérations sont de type *axpy* sur des vecteurs de taille n_x environ égale à 10000. On a vu précédemment que le *multithreading* pour chaque opération n'est pas du tout adapté. D'autres manières de paralléliser pourraient être envisagées mais les gains seraient tellement minimes que cela n'a pas été implémenté.

6.8.3 Liste des tâches de la Réduction

On introduit ici une notion de liste nécessaire pour le traitement du second membre et de la reconstruction. Pour fixer les idées, considérons à un cas très

simple de douze lignes réparties sur quatre processus :

	r	0	1	2	3
Proc 1	1	•			
	2	•	•		
	3	•			
Proc 2	4	•	•	•	
	5	•			
	6	•	•		
Proc 3	7	•			
	8	•	•	•	•
	9	•			
Proc 4	10	•	•		
	11	•			
	12	•	•	•	

A l'étape $r = 1$, on fait une combinaison linéaires des lignes trois à trois et on élimine une ligne sur deux. On continue jusqu'à l'étape $r = 3$ où il ne reste plus que la ligne 8.

On définit alors une liste pour chaque ligne indiquant pour l'étape r si la ligne est éliminée, à éliminer ou reste à l'étape suivante. L'idée d'utiliser cette liste de tâche est de contenir toutes les informations de la Réduction Cyclique, évitant ainsi de convertir les indices globaux des lignes en indices locaux et inversement. Il suffit pour chaque processus de parcourir cette liste (ou la portion de liste le concernant) pour connaître ses opérations et les données dont il a besoin pour l'étape de réduction.

En notant pour 0 une ligne éliminée, 1 pour une ligne à éliminer et 2 pour une ligne conservée à l'étape suivante, on peut écrire le cas simple proposé ainsi :

	r	0	1	2	3
Proc 1	1	1	0	0	0
	2	2	1	0	0
	3	1	0	0	0
Proc 2	4	2	2	1	0
	5	1	0	0	0
	6	2	1	0	0
Proc 3	7	1	0	0	0
	8	2	2	2	1
	9	1	0	0	0
Proc 4	10	2	1	0	0
	11	1	0	0	0
	12	2	2	1	0

Dès lors, il suffit de parcourir la liste définie ainsi pour connaître les opérations à effectuer pour le second membre (ou la redistribution). Lorsqu'on rencontre un 2, on cherche ses voisins 1. On connaît immédiatement les indices locaux et processeurs des blocs à utiliser. Cette liste est évidemment distribuée. Par la suite, on note l_n^r la liste des tâches du processus n à l'étape r.

6.8.4 Calcul dans la base des vecteurs propres

Durant la méthode, on calcule les composantes de f dans la base des vecteurs propres (voir page 62)

$$\tilde{f}_i = Q^T f_i \quad \text{pour} \quad i = 1, \ldots, n_y,$$

puis après élimination et redistribution, on obtient la solution u à partir de $\tilde{u}$

$$u_i = Q\tilde{u}_i \quad \text{pour} \quad i = 1, \ldots, n_y.$$

Ceci amène à considérer deux produits de la matrice Q de taille $n_x \times n_x$ par la matrice notée B composée des blocs du second membre et de la solution dans la base des vecteurs propres, de tailles $n_x \times n_y$.

Rappelons que dans un cas réaliste, la taille de la matrice Q dépasse la mémoire théorique de chaque processeur. On a besoin du produit et de sa transposée ; aucune optimisation de mémoire contiguë n'est donc possible pour les deux produits.

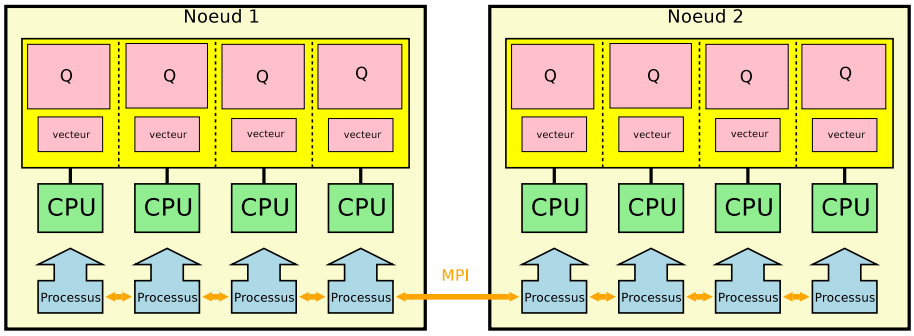


FIG. 6.13 – Programmation avec *MPI*

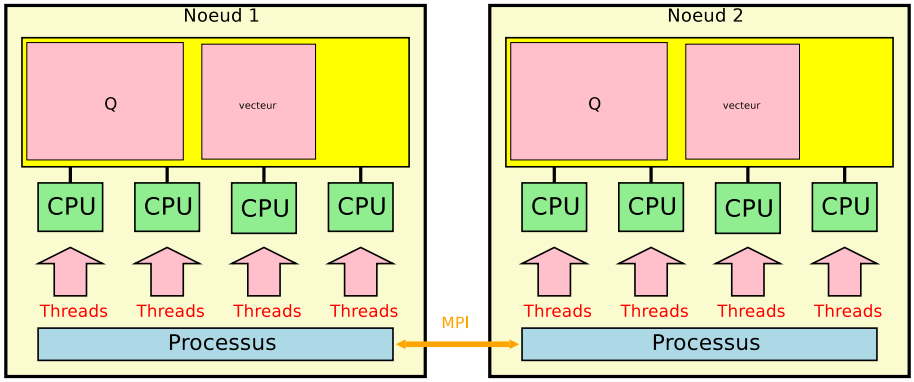


FIG. 6.14 – Programmation hybride avec *MPI- multithreading*

Dans une optique simple *MPI*, une idée naturelle consiste à répartir la matrice sur chaque processeur du noeud. Les blocs de vecteurs à multiplier ne sont

pas distribués ; il faut les envoyer sur chaque processeur. Il faut ensuite redistribuer les blocs solutions des produits à chaque processeur. Dans cette approche classique, les communications entre processeurs effectuées par *MPI* sont très nombreuses et pénalisantes pour les performances.

En utilisant le *multithreading* sur un processus par noeud, on bénéficie d'un double avantage sur ce type de problème ; premièrement, les communications au sein du noeud sont simplement remplacées par des accès à la mémoire locale et deuxièmement, la matrice tient globalement en mémoire. On peut effectuer les produits matrice-matrice par blocs en répartissant les tâches par threads.

6.8.5 Récurrence sur le second membre

On s'intéresse ensuite à la construction du second membre aux différentes étapes de la réduction (voir page 60).

Le second membre est réparti par blocs de lignes successives sur chaque processus associé à chacun des sous-domaines. Rappelons qu'à chaque étape $r \leq k-1$ de la réduction et pour les lignes i concernées, on veut calculer le vecteur

$$\tilde{f}_{i.2^r}^{(r)} = \tilde{f}_{i.2^r}^{(r-1)} + \Gamma^{(r-1)} \left(\Lambda^{(r-1)}\right)^{-1} \left(\tilde{f}_{i.2^r-2^{r-1}}^{(r-1)} + \tilde{f}_{i.2^r+2^{r-1}}^{(r-1)}\right).$$

Aux premières étapes de réduction, chaque processeur a plusieurs vecteurs à traiter. Il n'y a pas de processeur inactif. Par contre, à partir du rang r_0 tel que $2^{r_0} > \dfrac{n_y}{n_p}$, certains processeurs deviennent inactifs.

Algorithme 6 Récurrence sur le second membre sur le processeur m

Entrée: $\tilde{f}$, Λ, Γ, k
Sortie: Récurrence sur $\tilde{f}$

```
pour r = 1, ..., k faire
  c = 0
  pour j = 1, ... n_y / n_p faire
    si l_m^r(j) = 2 alors
      c = c + 1
      v_c^m = f̃_{c+}^{(r)} + f̃_{c-}^{(r)}
    fin si
    pour p = 1, ..., c faire
      si (p = 1 et m = 1) ou (p = c et m = n_p) alors
        f̃_c^{(r)} = Γ^{(r-1)} (Π^{(r-1)})^{-1} v_c^m
      sinon
        f̃_c^{(r)} = Γ^{(r-1)} (Λ^{(r-1)})^{-1} v_c^m
      fin si
    fin pour
  fin pour
fin pour
retourner f̃
```

Précisons qu'on stocke $\tilde{f}^{(r)}$ dans le vecteur $\tilde{f}$.

On utilise le principe de liste définie ci-dessus. On note $\tilde{f}_{c+}^{(r)}$ (resp. $\tilde{f}_{c-}^{(r)}$) le bloc vecteur second membre le plus proche par dessus (resp. dessous) du c-ième bloc local $\tilde{f}_c^{(r)}$ que l'on garde dans la liste. Cette étape très courte ne justifie pas une parallélisation par *multithreading*.

6.8.6 Résolution du problème réduit et redistribution

A ce stade, on possède les valeurs propres, vecteurs propres et second membre à toutes les étapes de la réduction. On entre alors dans une phase de résolution et redistribution des équations éliminées (voir page 60). On résout d'abord le problème (4.11) associé à :

$$A^{(k-1)} u_{2^{k-1}} = f_{2^{k-1}}^{(k-1)}.$$

Autrement dit, on calcule

$$\tilde{u}_{2^{k-1}} = \left(\Lambda^{(k-1)}\right)^{-1} \tilde{f}_{2^{k-1}}^{(k-1)}.$$

Enfin, on s'intéresse à la redistribution des équations éliminées (voir page 60). On souhaite donc résoudre le système (4.12) associé de dimension $2^{k-r}-1$ blocs s'écrivant :

$$\begin{pmatrix} B^{(r)} & & & \\ & A^{(r)} & & \\ & & \ddots & \\ & & & B^{(r)} \end{pmatrix} \begin{pmatrix} u_{2^{r+1}-2^r} \\ u_{2.2^{r+1}-2^r} \\ \vdots \\ u_{n-2^r+1} \end{pmatrix} = \begin{pmatrix} g_{2^{r+1}-2^r}^{(r)} \\ g_{2.2^{r+1}-2^r}^{(r)} \\ \vdots \\ g_{n-2^r+1}^{(r)} \end{pmatrix}$$

où

$$g_{j.2^{r+1}-2^r}^{(r)} = f_{j.2^{r+1}-2^r}^{(r)} + T^{(r)} \left(u_{(j-1).2^{r+1}} + u_{(j).2^{r+1}}\right).$$

On calcule donc pour $i = 1, \ldots, 2^{k-r}-1$

$$\tilde{u}_{i.2^{r+1}-2^r} = \left(\Lambda^{(r)}\right)^{-1} \left(\tilde{f}_{i.2^{r+1}-2^r}^{(r)} + \left(\Gamma^{(r)}\right) \left(\tilde{u}_{(i-1).2^{r+1}} + \tilde{u}_{i.2^{r+1}}\right)\right).$$

Comme pour le second membre $\tilde{f}^{(r)}$, on stocke $\tilde{u}$ dans le vecteur $\tilde{f}$. Dans le même esprit, on applique une procédure similaire de calcul en utilisant le principe de liste. Cette étape ne justifie également pas une parallélisation par *multithreading*.

Algorithme 7 Problème réduit et redistribution sur le processeur m

Entrée: $\tilde{u}$, Λ, Γ
Sortie: Récurrence sur $\tilde{u}$

```
pour r = 1, ..., k − 1 faire
  c = 0
  pour j = 1, ..., n_y / n_p faire
    si l_m^r(j) = 1 alors
      c = c + 1
      v_c^m = ũ_{c−} + ũ_{c+}
    fin si
  fin pour
  pour p = 1, ..., c faire
    si (p = 1 et m = 1) ou (p = c et m = n_p) alors
      g_c = (Π^(r))^{-1} f̃_c^r
      s_c = Γ^(r) (Π^(r))^{-1} v_c^m
    sinon
      g_c = (Λ^(r))^{-1} f̃_c^r
      s_c = Γ^(r) (Λ^(r))^{-1} v_c^m
    fin si
    ũ_c = s_c + g_c
  fin pour
fin pour
retourner ũ
```

6.9 Performances

Dans cette section, on présente les performances obtenues par la parallélisation des méthodes en utilisant les techniques présentées jusqu'ici dans le code *HERA*. On s'intéressera au modèle Helmholtz et en particulier aux temps de calcul de la méthode de Krylov et on donnera les temps d'initialisation de la méthode LR et des décompositions LU.

6.9.1 Approche *hybride*

Considérons un cas test avec $n_x = n_y = 1500$ d'environ 2 millions d'inconnues. L'idée est de comparer les performances d'une exécution parallèle classique *MPI* et une execution *multithreading* pour un pas de temps donné. On effectue la résolution sur un unique noeud de 4 processeurs où l'on fait varier le nombre de threads et de processus *MPI*. Pour information, la décomposition LU prend une seconde et la méthode LR calcule le spectre complet en 8 secondes en séquentiel à 4 secondes avec un nombre variable de threads.

La méthode itérative a convergé en 5 itérations pour le critère d'arrêt choisi. Les résultats de calcul sont résumés dans les tableaux suivants. En rouge, on souligne les résultats séquentiels et les résultats de comparaison parallèle.

Temps calcul					
Nb processus *MPI*	séquentiel	1 thread	2 threads	3 threads	4 threads
1	155s	157s	82s	57s	41s
2	78s	79s	42s	44s	44s
3	56s	55s	45s	45s	50s
4	41s	41s	47s	47s	51s
Efficacité					
Nb processus *MPI*	séquentiel	1 thread	2 threads	3 threads	4 threads
1	1	0.99	0.94	0.9	0.94
2	0.99	0.98	0.92	0.88	0.88
3	0.92	0.94	0.86	0.86	0.77
4	0.94	0.94	0.82	0.82	0.76

TAB. 6.1 – Performances *MPI* vs *multithreading* sur un nœud de 4 processeurs

Procs	16	32	64	128
Temps	492s	249s	126s	64s
Efficacité	1	0.987	0.976	0.96

TAB. 6.2 – Performances pour problème de taille fixe

On constate que la programmation *multithreading* peut donner des résultats similaires à l'utilisation de *MPI*. Par contre, le couplage des deux nécessite de ne pas utiliser trop de threads par processus *MPI*. Typiquement, on remarque une perte de performance quand il y a plus de 4 threads sur le noeud. Par exemple, avec 4 processus *MPI* utilisant 4 threads, on atteind seulement 76% d'efficacité, ce qui est loin des 94% obtenu avec un seul thread ou processus *MPI*.

Par la suite, on utilise la parallélisation *hybride* consistant à utiliser un unique processus *MPI* par noeud et la programmation *multithreading*.

6.9.2 Scalabilité, problème fixe

On présente maintenant les temps calcul d'un cas sur un maillage d'environ 40 millions d'inconnues ($n_x = 4000$, $n_y = 10000$) à un pas de temps donné. La méthode itérative a convergé en 3 itérations, la décomposition LU prend 3 secondes et la méthode LR calcule le spectre complet en 53 secondes en séquentiel et 31 secondes avec 4 threads.

Procs	1	4	16	64	256
$\#ddl \times 10^6$	0.4	1.5	6.3	25.4	101.6
Temps LR	$1s$	$3s$	$12s$	$48s$	$189s$
Temps $GMRES$	$4.75s$	$11.6s$	$24s$	$47s$	$93s$

TAB. 6.3 – Performances pour problème de taille doublé

De part la répartition des données (6.7), ce test consiste à regarder la scalabilité dans la direction y. Du fait de la répartition en bande des données, augmenter le nombre de processeurs revient à diminuer le nombre de vecteurs à multiplier par base des vecteurs propres. Il est ainsi cohérent d'avoir le temps calcul se divisant par deux si on multiplie par deux le nombre de processeurs. On constate des très bons résultats de scalabilté pour un problème fixe bien que la Réduction Cyclique soit une méthode de parallélisation particulière du fait précisément de la réduction. On peut ainsi conclure qu'augmenter le nombre de point dans la direction y augmente linéairement le temps calcul.

6.9.3 Scalabilité, problème doublé

Le dernier cas test consiste à étudier les performances dans le cas où l'on double le nombre d'inconnues dans chaque direction. La section précédente montre la très bonne scalabilité pour la direction y. Doubler le nombre d'inconnues dans la direction x a pour principal impact de doubler le rang de la base des vecteurs propres. En plus d'augmenter le temps de calcul du spectre, on augmente ainsi non linéairement le temps de multiplication de la base par le second membre. En particulier, le produit matrice-matrice requiert $2n_x^2 n_y$ opérations. Toutefois, doubler n_x n'implique pas quadripler le temps calcul pour des raisons purement informatiques (*blocking*).

On s'intéresse ici au coût initial de diagonalisation (méthode LR) et au temps moyen d'une itération de la méthode de Krylov. Le nombre d'itérations de la méthode n'est pas

On remarque que chaque fois qu'on double le nombre de point en x, on double environ le temps calcul. On pourrait améliorer la mauvaise scalabilité en x en effectuant une décomposition de domaine dans cette direction. On réduirait ainsi le temps moyen d'une itération de la méthode mais on augmenterait également le nombre d'itérations. Toutefois, bien qu'envisagée, cela ne représente pas encore une nécessité.

7

Résultats numériques et performances

Sommaire

Dans ce chapitre, on présente les résultats numériques obtenus en utilisant la méthode décrite jusqu'ici, de la décomposition de domaine à la parallélisation. On s'intéressera à la cohérence des résultats par rapport évidement à la physique mais aussi par rapport au modèle existant. On abordera aussi des aspects purement informatique de performance.

Dans un premier temps, on présentera une simulation ayant pour but la validation de la méthode et du code. On comparera nos résultats avec les résultats validés *a priori* du code *HERA* par le modèle paraxial. On simulera alors la propagation d'un faisceau dans un plasma sous-dense par les modèles paraxial et Helmholtz. Après avoir constaté la cohérence des résultats, on présentera également des résultats nouveaux de couplage des modèles Paraxial et Helmholtz avec le modèle hydrodynamique. Ces simulations sont effectuées sans absorption physique afin d'éprouver la robustesse de la méthode. Ceci engendre également un certains nombre d'itérations de la méthode de Krylov. Un cas réaliste de propagation droite avec absorption sera alors présenté.

On présentera ensuite plusieurs résultats comparatifs d'analyse de la méthode. La condition de raccord entre les modèles sera étudiée. On montrera également que le critère de discrétisation de 10 points par longueur d'onde est suffisant. Enfin, on comparera les comportements des méthodes de Krylov *GMRES*, *BICGStab* et *CGS*.

On s'intéressera ensuite à un cas de propagation d'un faisceau composé d'un unique speckle dans un plasma de densité variant jusqu'à la densité critique. L'intérêt de ce type de simulation près des caustiques est de pouvoir prendre en compte l'éclatement et l'interaction des faisceaux qui peuvent déplacer les dépôts d'énergie en comparaison des résultats obtenus par des calculs avec des méthodes type *ray-tracing*. Le maillage *fin* Helmholtz considéré sera de plusieurs dizaines de millions de points. On simulera ensuite des propagations de faisceaux plus complexes composés de plusieurs speckles. Des phénomènes d'interaction de faisceaux seront alors visibles. Ces résultats de propagation près de la caustique sont à notre connaissance nouveaux.

Enfin, un cas sur un domaine réaliste sera présenté. En particulier, une vitesse de l'hydrodynamique transverse à la propagation sera appliquée afin de mettre en évidence des déplacements de dépôts d'énergie. On considerera un maillage *fin* Helmholtz de 200 millions résolu sur 256 processeurs.

Remarque On ne peut comparer nos résultats avec ceux de la littérature. En effet, les seuls résultats disponibles sont obtenus avec des codes basés sur le modèle paraxial donc ne pouvant traiter des faisceaux courbes. Il existe toutefois des codes de résolution du modèle Helmholtz écrits par des physiciens utilisant une transformée de Fourier en y qui permettent des variations de densité jusqu'à la densité critique. Ces codes traitent généralement des faisceaux droits afin d'observer le phénomène de retrodiffusion Brillouin.

Remarque Tous les résultats seront visualisés sur le maillage *fluide* grossier. En effet, la visualisation pose problème pour les tailles de maillage *fin* Helmholtz. Dans les simulations, on calcule la norme de l'intensité laser $|\psi|^2$ et la densité N_e electronique.

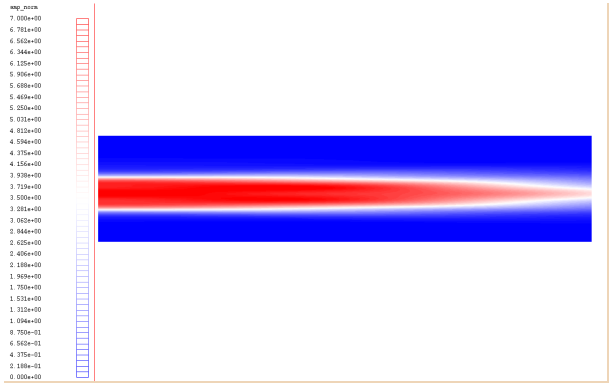
7.1 Cas de validation ; Paraxial vs Helmholtz

On considère le domaine de simulation rectangulaire $\mathcal{D} = [0, L_x] \times [0, L_y]$ où on a noté la longueur $L_x = 700\lambda_0$ et la hauteur $L_y = 150\lambda_0$ avec une longueur d'onde $\lambda_0 = 0.351\mu m$. L'onde entrante est un faisceau sans angle d'incidence composé de trois speckles de fortes intensités. Le pas de temps $\delta_t = 0.1\ ps$ restera toujours fixe le long des simulations afin ici d'atteindre un temps maximum de 10 ps. La densité initiale $N = N_e/N_c$ est égale à 25% de la densité critique. Précisons également que nous ne considérons pas dans ce cas d'absorption physique, $\nu = 0$. Ainsi, et compte tenu de la valeur de la densité initiale élevée pour le modèle paraxial, on compte rapidement observer des instabilités dans la propagation du faisceau telles que la focalisation et la filamentation de celui-ci. On résout dans un premier temps, l'équation (1.18) couplée au système hydrodynamique (1.19), (1.20) sur le maillage *grossier* fluide de pas de discrétisation $\Delta_x = \Delta_y = \lambda_0/2$, soit un maillage de 840000 mailles. En appliquant la méthode de résolution décrite dans le chapitre (2), les temps de calcul pour cette simulation sont de l'ordre du quart d'heure. On résout dans un second temps l'équation (1.6) sur le maillage *Helmholtz* fin couplée au même système hydrodynamique résolu sur la maillage *fluide* grossier. On utilise les mécanismes d'interpolation et de résolution décrits dans le chapitre (2) et la partie (II). Avec le critère de discrétisation de 10 points par longueur d'onde $\delta_x = \delta_y = \lambda_0/10$, le maillage *Helmholtz* fin compte alors environ 10,5 millions d'inconnues. Sur 64 processeurs (16 noeuds) et avec les techniques de parallélisation discutées dans le chapitre précédent, on constate que l'itération de Krylov occupe environ 15s de temps calcul. Le calcul du spectre complet prend environ 160s pour la matrice de taille 7000×7000. Le calcul global a duré environ 7 heures.

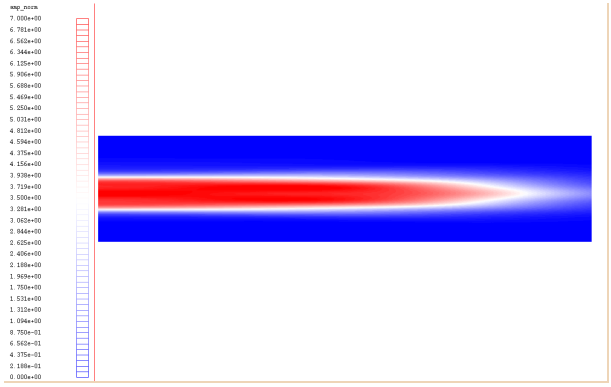
Remarque Pour des raisons de validation en cours, le *multithreading* n'est pas encore utilisé pour les opérations type *axpy* de la méthode de Krylov.

On présente les résultats à différents pas de temps et on propose également une coupe suivant la longueur du domaine sur le maillage *fluide* grossier pour comparaison.

– Temps $t = 1\ ps$

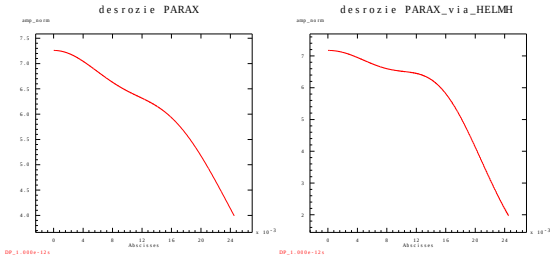


(a) Paraxial



(b) Helmholtz

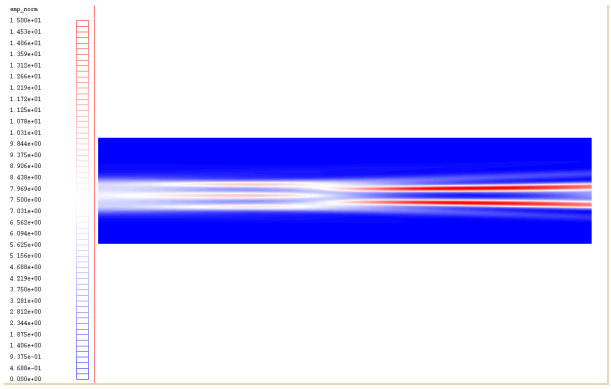
FIG. 7.1 – Intensité laser $|\psi|^2$ à $t = 1\ ps$, Parxial vs Helmholtz



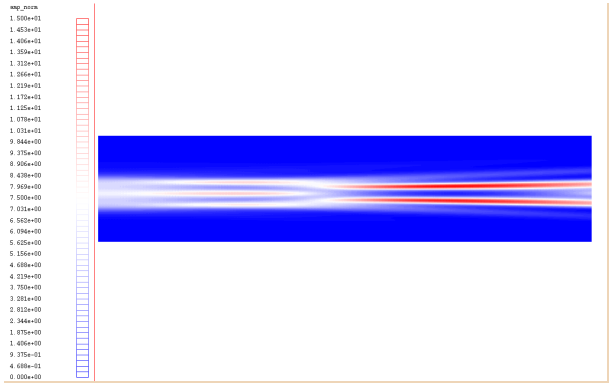
(a) Paraxial (b) Helmholtz

FIG. 7.2 – Coupe de $|\psi|^2$ à $t = 1\ ps$, Parxial vs Helmholtz

– Temps $t = 5\ ps$

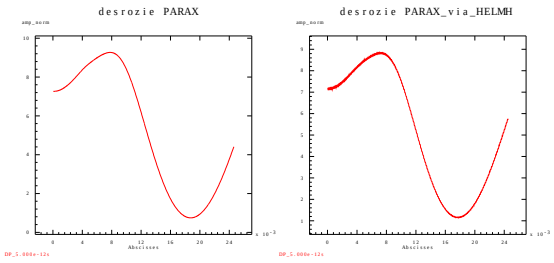


(a) Paraxial



(b) Helmholtz

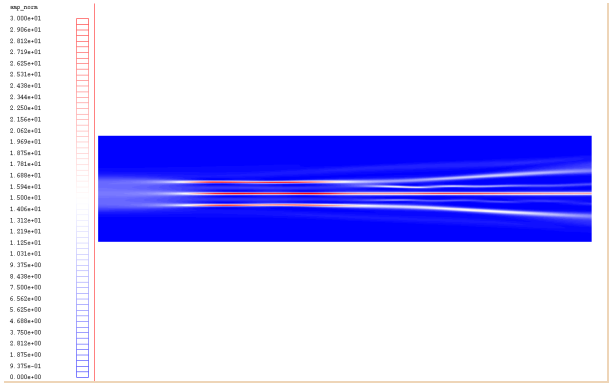
FIG. 7.3 – Intensité laser $|\psi|^2$ à $t = 5\ ps$, Parxial vs Helmholtz



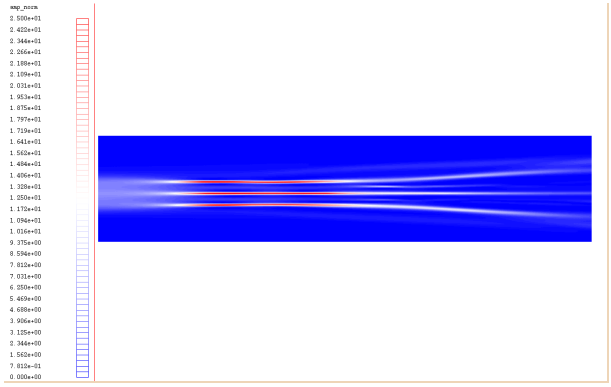
(a) Paraxial (b) Helmholtz

FIG. 7.4 – Coupe de $|\psi|^2$ à $t = 5\ ps$, Parxial vs Helmholtz

– Temps $t = 10\ ps$

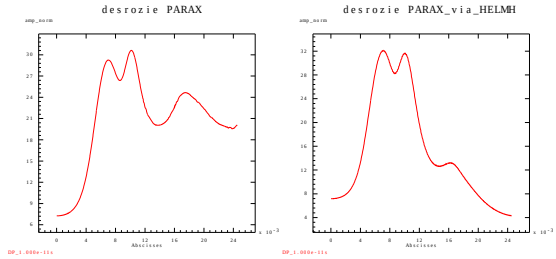


(a) Paraxial



(b) Helmholtz

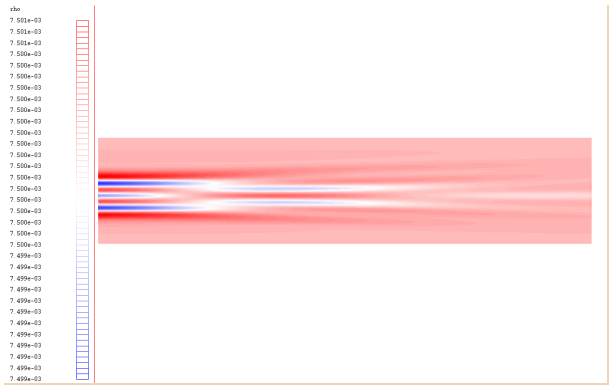
FIG. 7.5 – Intensité laser $|\psi|^2$ à $t = 10\ ps$, Parxial vs Helmholtz



(a) Paraxial (b) Helmholtz

FIG. 7.6 – Coupe de $|\psi|^2$ à $t = 10\ ps$, Parxial vs Helmholtz

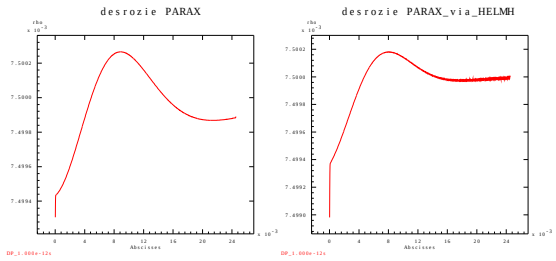
– Temps $t = 1\ ps$



(a) Paraxial

(b) Helmholtz

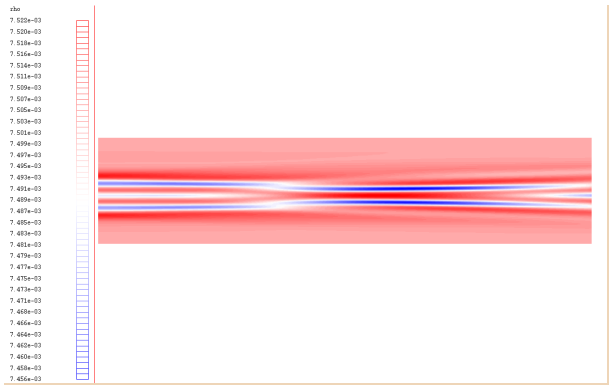
FIG. 7.7 – Densité $N_e(x, y)$ à $t = 1\ ps$, Parxial vs Helmholtz



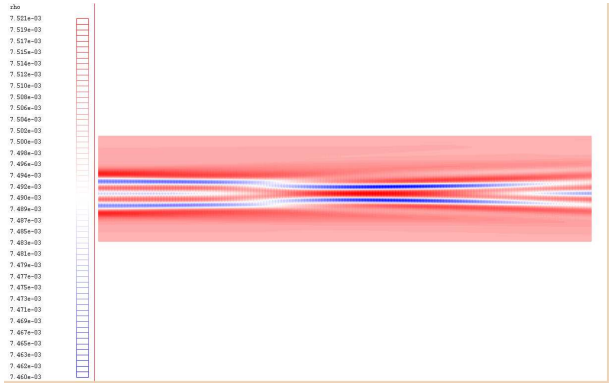
(a) Paraxial (b) Helmholtz

FIG. 7.8 – coupe de $N_e(x, y)$ à $t = 1\ ps$, Parxial vs Helmholtz

– Temps $t = 5\ ps$

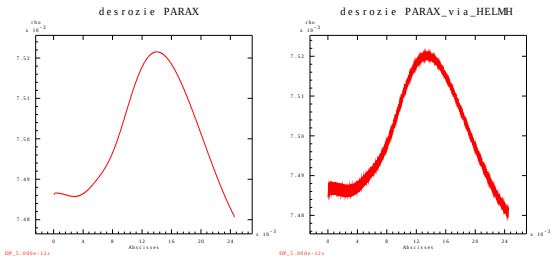


(a) Paraxial



(b) Helmholtz

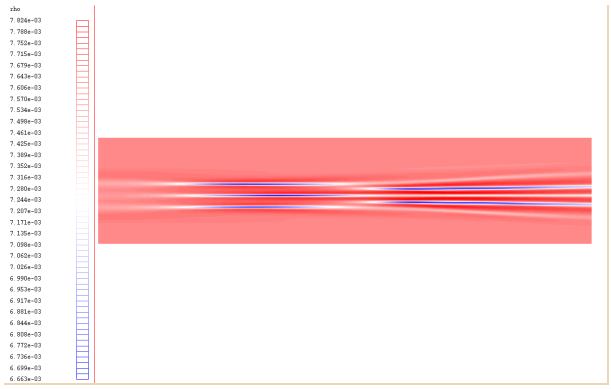
FIG. 7.9 – Densité $N_e(x, y)$ à $t = 5\ ps$, Parxial vs Helmholtz



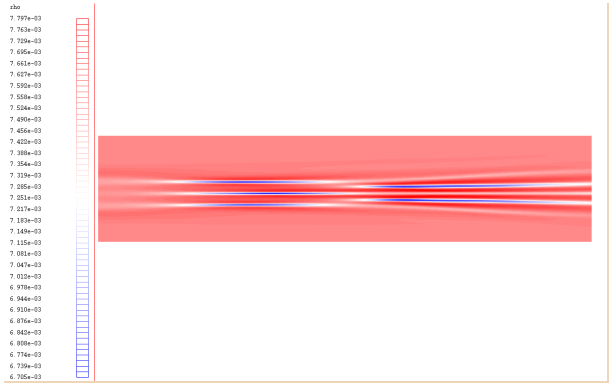
(a) Paraxial (b) Helmholtz

FIG. 7.10 – Coupe de $N_e(x, y)$ à $t = 5\ ps$, Parxial vs Helmholtz

– Temps $t = 10\ ps$

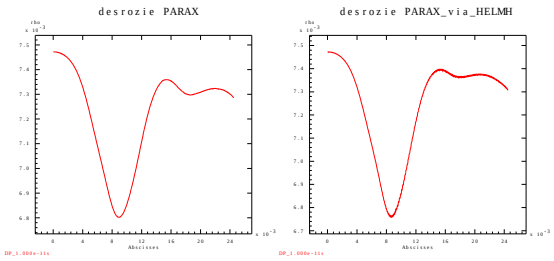


(a) Paraxial



(b) Helmholtz

FIG. 7.11 – Densité $N_e(x, y)$ à $t = 10\ ps$, Parxial vs Helmholtz



(a) Paraxial (b) Helmholtz

FIG. 7.12 – Coupe de $N_e(x, y)$ à $t = 10\ ps$, Parxial vs Helmholtz

La simulation, pour les deux modèles, montre comment le faisceau se propage dans le domaine et autofocalise en plusieurs points chauds aux temps $t = 5\ ps$ et $t = 7\ ps$. On observe ensuite la diffraction de l'onde à $t = 10\ ps$. Les résultats de la comparaison sont très satisfaisants et montrent une certaine cohérence entre les modèles. En effet, on remarque que le modèle paraxial, par nature, accentue légèrement plus la propagation de l'onde, le modèle Helmholtz diffractant plus. Cette simulation montre particulièrement la robutesse du modèle paraxial. On aurait pu être en droit de se demander la validité des résultats au vue de la diffraction de l'onde.

Il est également intéressant de remarquer les légères oscillations de l'intensité et de la densité aux premiers temps de résolution. Elles sont dues à la discrétisation de la condition limite de sortie. On les remarque surtout dans la densité car les échelles sont très fines, les fluctuations étant très faibles aux premiers pas de temps.

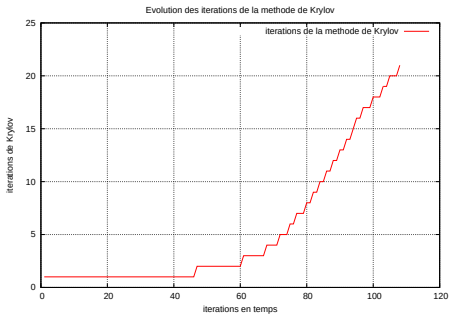


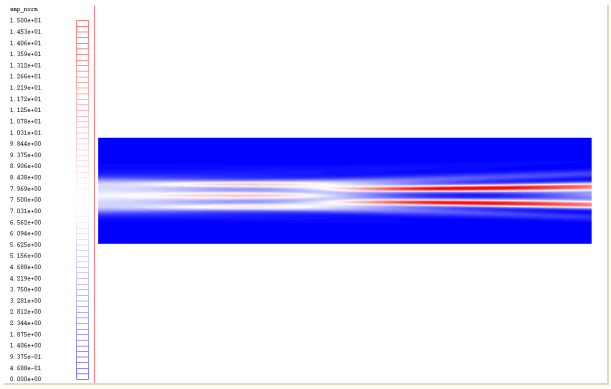
FIG. 7.13 – Evolution des itérations de Krylov en temps, Helmholtz

L'évolution au cours des itérations en temps du nombre d'itérations de la méthode de Krylov est un élément intéressant. Cette évolution dépend directement de la densité et des fluctuations de densité. Dans notre cas, la densité commence à se creuser et les flucuations s'amplifient, augmentant ainsi le nombre d'itérations de la méthode de Krylov.

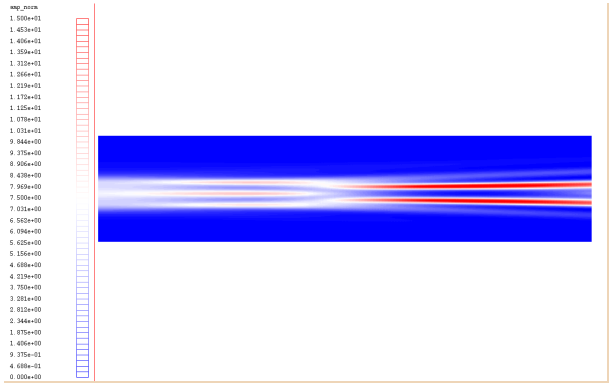
7.2 Couplage Paraxial/Helmholtz, Paraxial vs Paraxial + Helmholtz

On effectue maintenant la même simulation en couplant les modèles Paraxial et Helmholtz. Précisons que coupler deux modèles avec l'hydrodynamique est nouveau. On définit alors le domaine $\Omega = [x_f, L_x] \times [0, L_y] \subset \mathcal{D}$ dans lequel on utilise le modèle Helmholtz et où on a noté $x_f = 300\lambda_0$. Ainsi, on résout d'abord le modèle paraxial dans $\mathcal{D} \setminus \Omega$ puis le modèle Helmholtz dans Ω. On est alors amené à considérer un maillage *Helmholtz* grossier d'environ 6 millions d'inconnues. On présente les résultats d'une simultation jusqu'à 15 ps sur 64 processeurs (16 noeuds). On exécute un processus *MPI* par noeud. Le temps de calcul d'une iteration de Krylov est de 5,8s et le calcul du spectre de 54s. La simulation est faite en environ 5h de calcul.

– Temps $t = 5\ ps$

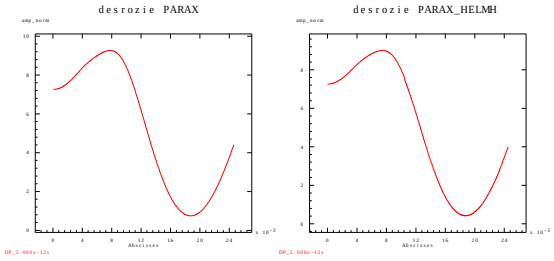


(a) Paraxial



(b) Paraxial + Helmholtz

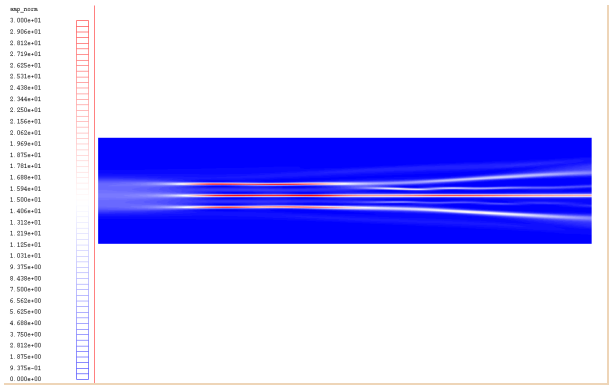
FIG. 7.14 – Intensité laser $|\psi|^2$ à $t = 5\ ps$, Parxial vs Paraxial + Helmholtz



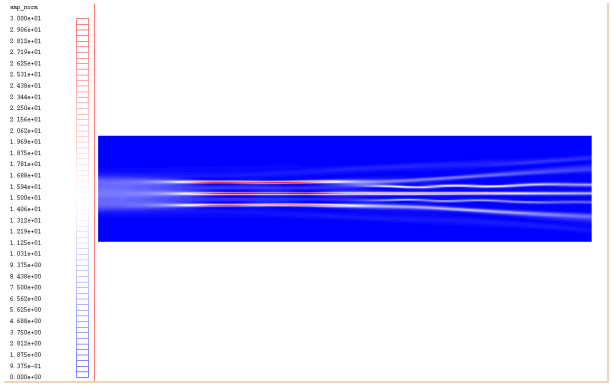
(a) Paraxial (b) Parax + Helmh

FIG. 7.15 – Coupe de $|\psi|^2$ à $t = 5\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 10\ ps$

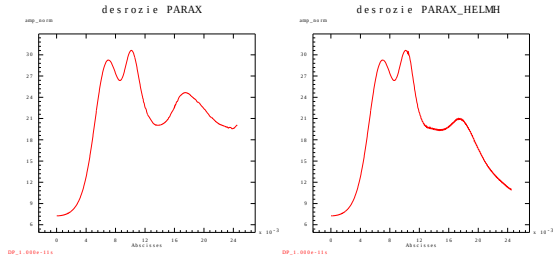


(a) Paraxial



(b) Parax + Helmh

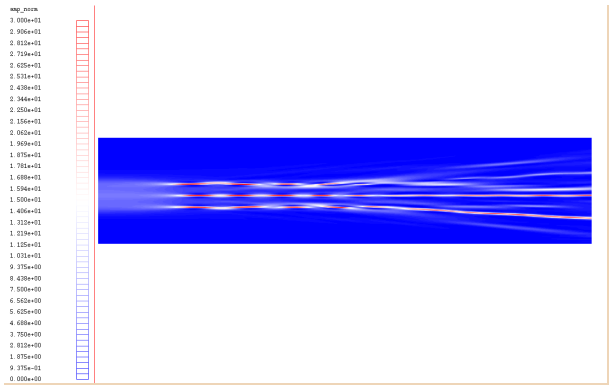
FIG. 7.16 – Intensité laser $|\psi|^2$ à $t = 10\ ps$, Parxial vs Paraxial + Helmholtz



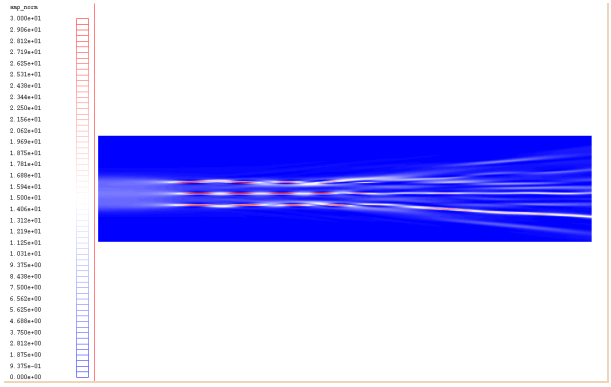
(a) Paraxial (b) Parax + Helmh

FIG. 7.17 – Coupe de $|\psi|^2$ à $t = 10\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 15\ ps$

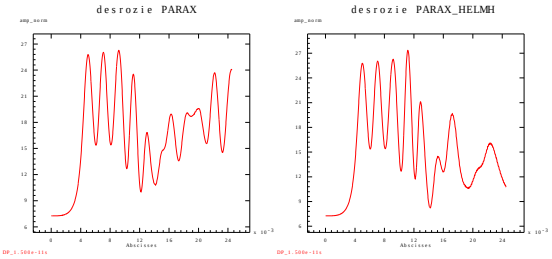


(a) Paraxial



(b) Paraxial + Helmholtz

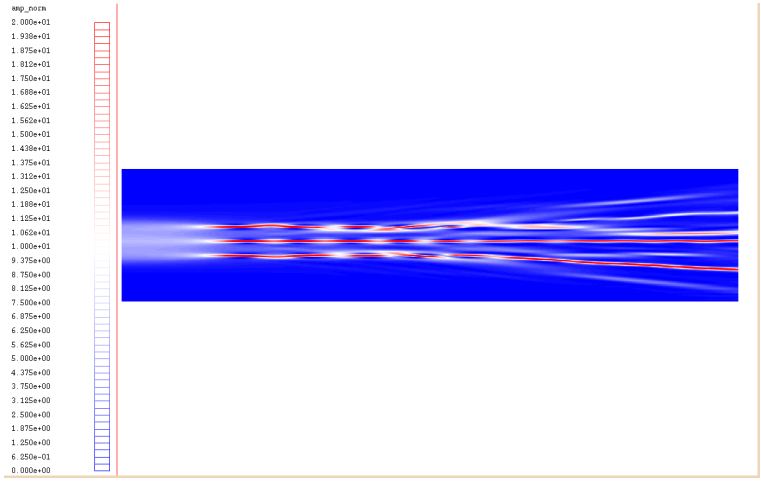
FIG. 7.18 – Intensité laser $|\psi|^2$ à $t = 15\ ps$, Parxial vs Paraxial + Helmholtz



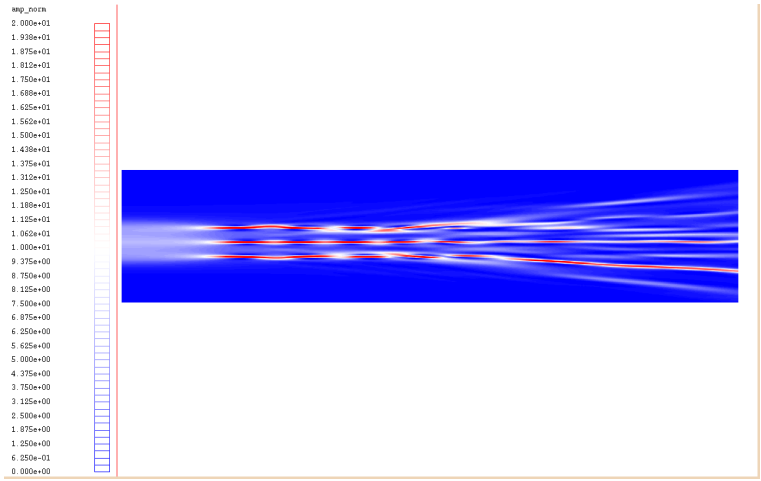
(a) Paraxial (b) Parax + Helmh

FIG. 7.19 – Coupe de $|\psi|^2$ à $t = 15\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 15\ ps$ avec une échelle de valeur différente



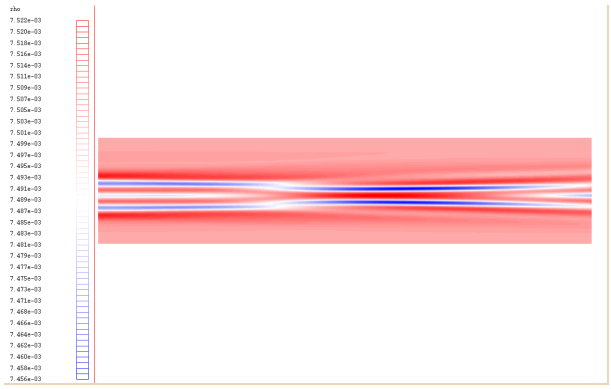
(a) Paraxial



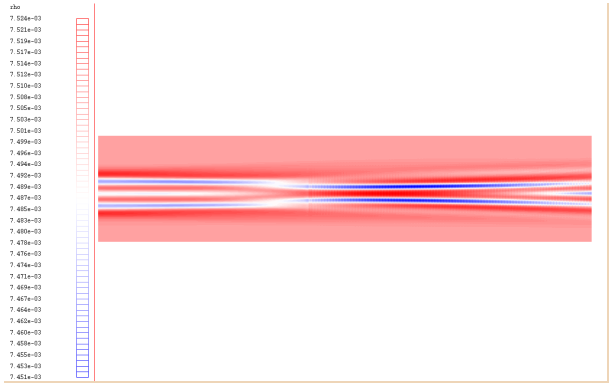
(b) Paraxial + Helmholtz

FIG. 7.20 – Intensité laser $|\psi|^2$ à $t = 15\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 5\ ps$

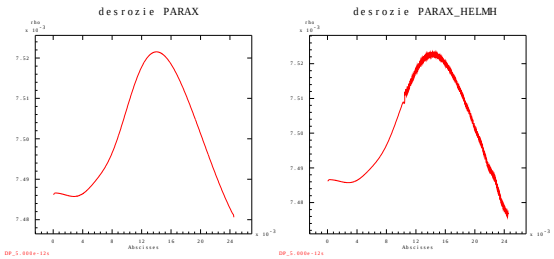


(a) Paraxial



(b) Paraxial + Helmholtz

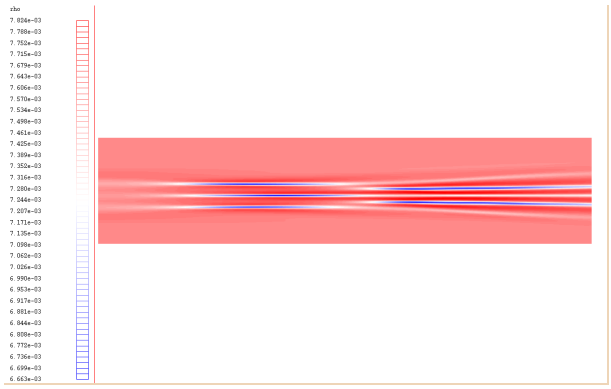
FIG. 7.21 – Densité $N_e(x, y)$ à $t = 5\ ps$, Parxial vs Paraxial + Helmholtz



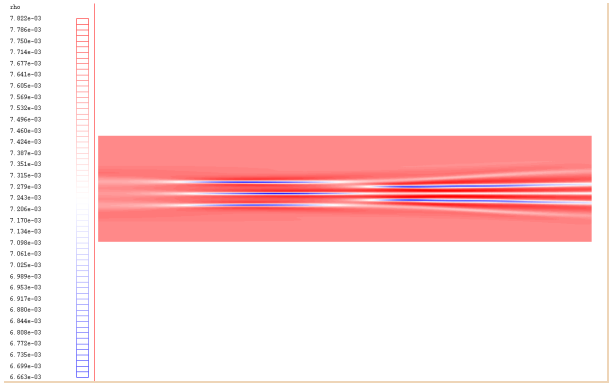
(a) Paraxial (b) Parax + Helmh

FIG. 7.22 – Coupe de $N_e(x, y)$ à $t = 5\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 10\ ps$

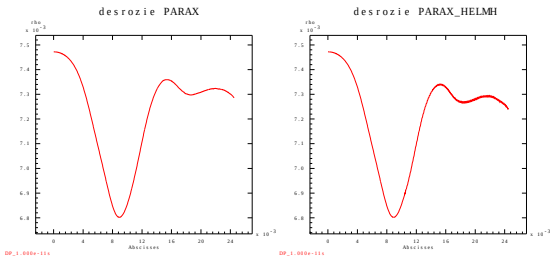


(a) Paraxial



(b) Paraxial + Helmholtz

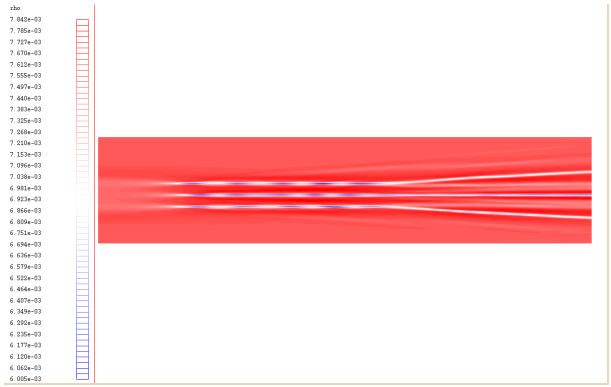
FIG. 7.23 – Densité $N_e(x, y)$ à $t = 10\ ps$, Parxial vs Paraxial + Helmholtz



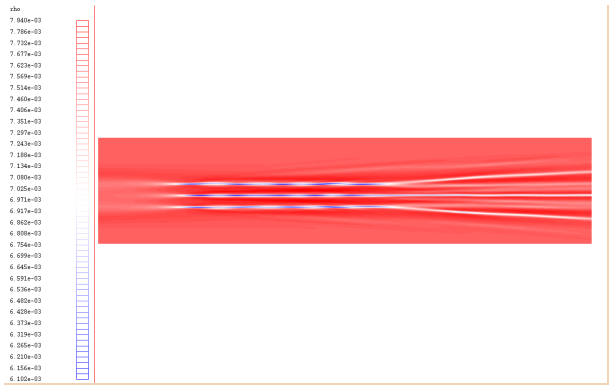
(a) Paraxial (b) Parax + Helmh

FIG. 7.24 – Coupe de $N_e(x, y)$ à $t = 10\ ps$, Parxial vs Paraxial + Helmholtz

– Temps $t = 15\ ps$

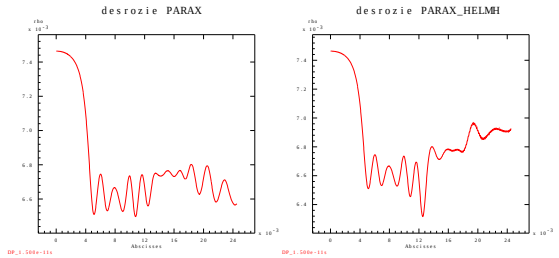


(a) Paraxial



(b) Paraxial + Helmholtz

FIG. 7.25 – Densité $N_e(x, y)$ à $t = 15\ ps$, Parxial vs Paraxial + Helmholtz



(a) Paraxial (b) Parax + Helmh

FIG. 7.26 – Coupe de $N_e(x, y)$ à $t = 15\ ps$, Parxial vs Paraxial + Helmholtz

Les résultats avec le couplage sont là aussi très satisfaisants et cohérents avec les résultats déjà obtenus. A l'interface, le couplage est invisible sauf par les petites oscillations engendrées par la condition de sortie. L'évolution du nombre d'itérations de la méthode de Krylov permet de montrer que la densité s'homogénéise vers $t = 12\ ps$ mais le creusement s'accentue encore après.

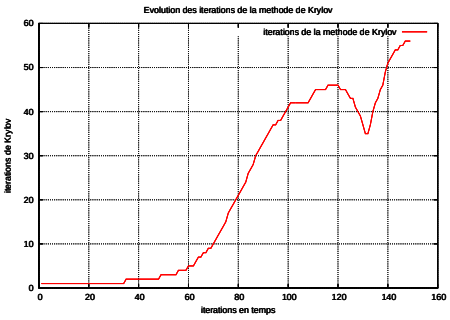


FIG. 7.27 – Evolution des itérations de Krylov en temps, Parxial + Helmholtz

7.3 Cas Helmholtz avec absorption

Dans les cas précédents, on constate que le laser creuse le plasma au cours du temps. Sans absorption, le creusement est très fort, en particulier au début de la simulation. Ensuite, l'hydrodynamique joue, le plasma se lisse et le faisceau diffracte. Ces effets expliquent le nombre d'itérations croissant de la méthode puis sa diminution. Dans un premier temps, le fort creusement local implique de fortes fluctuations de densité. Au fur et à mesure, les fluctuations s'homogénéisent vers la densité moyenne. L'absorption physique rend le problème numérique moins raide. Le nombre d'itérations de la méthode de Krylov diminue car le terme d'absorption correspond à de la coércivité. On s'intéresse donc au même cas que précédemment avec un coefficient d'absorption $\nu = \alpha N_0^2(x)$ où N_0 est la densité moyenne et $\alpha = 6 \times 10^{-3}$.

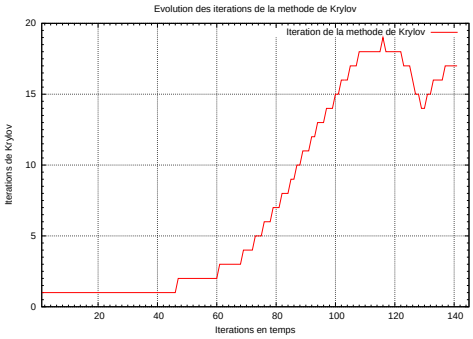
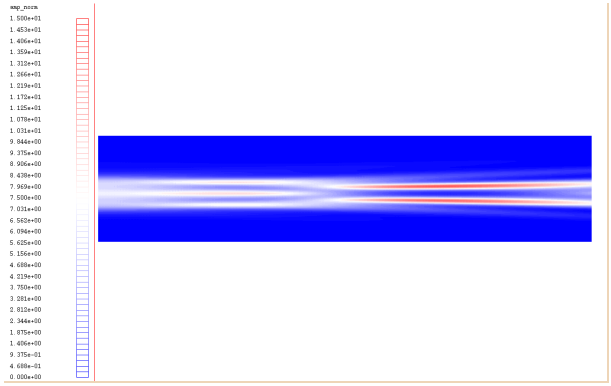
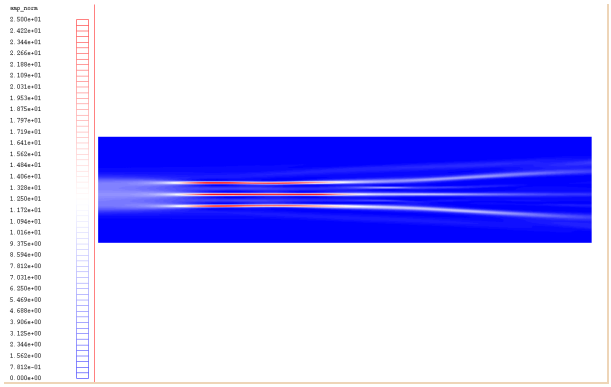


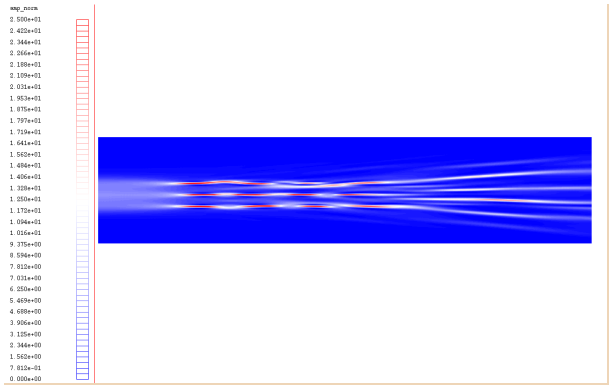
FIG. 7.28 – Evolution des itérations de Krylov en temps, Helmholtz absorption



(a) $t = 5\ ps$



(b) $t = 10\ ps$

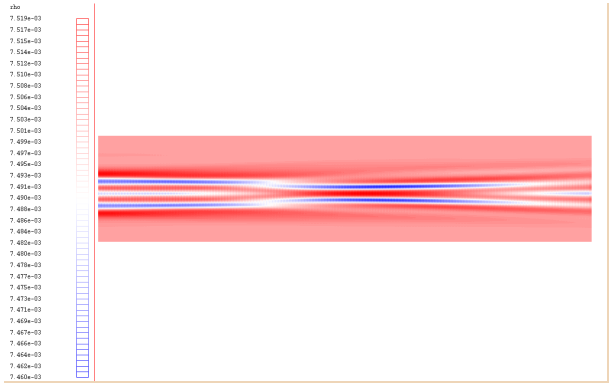


(c) $t = 14\ ps$

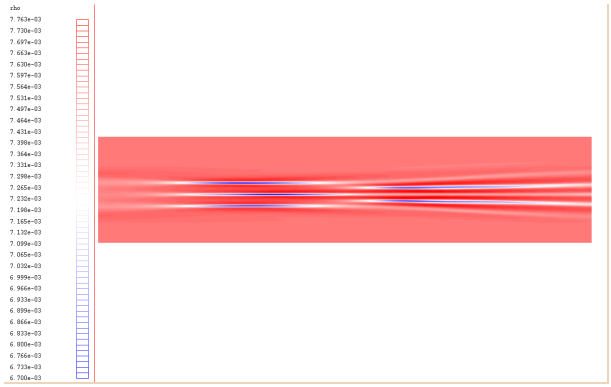
FIG. 7.29 – Cas Helmholtz avec absorption, Intensité laser $|\psi|^2$

7.4 Condition de raccord paraxial - Helmholtz

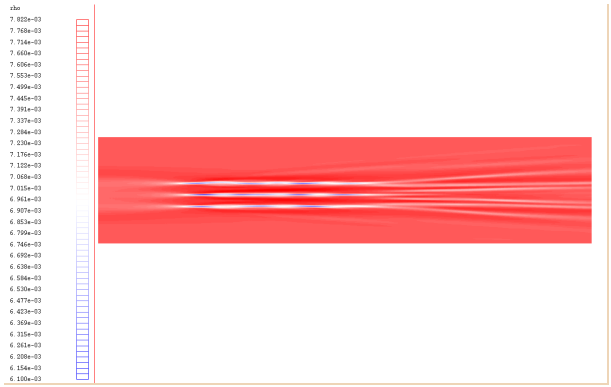
Le modèle paraxial ne permet la propagation de l'onde que dans une unique direction. En particulier, aucune onde ne peut venir de la zone Helmholtz dans la zone paraxiale. C'est pourquoi le couplage du modèle paraxial au modèle Helmholtz ne se fait que par la prise en compte de la condition (1.9) sur le bord



(a) $t = 5\ ps$



(b) $t = 10\ ps$



(c) $t = 14\ ps$

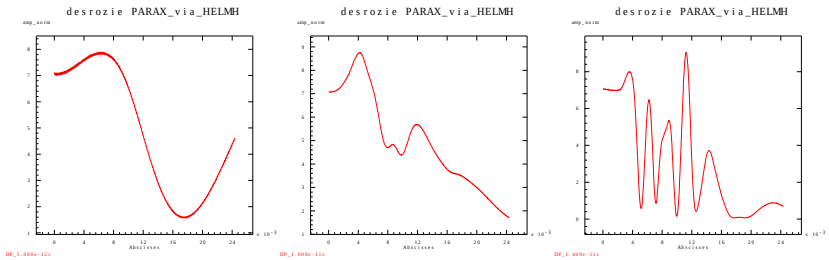
FIG. 7.30 – Cas Helmholtz avec absorption, Densité $N_e(x, y)$

entrant Γ^{in}

$$\left(n.\nabla + ik_0\sqrt{1 - N}e_b.n\right)\left(\psi - \psi^{in}\right) = 0$$

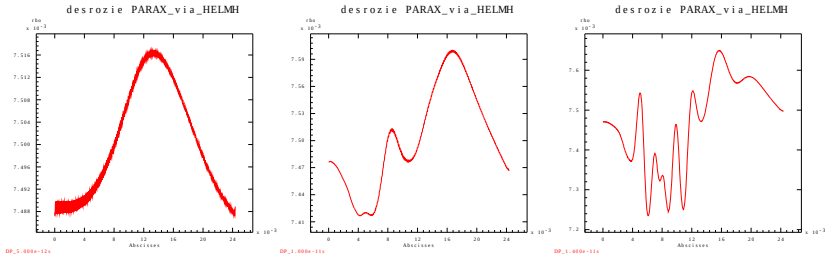
où e_b est le vecteur unitaire caractérisant la direction de propagation du faisceau laser et ψ^{in} l'onde incidente supposée de la forme $\alpha^{in}e^{ik_0\sqrt{1-N}e_b.x}$.

Dans un premier temps, l'écriture analytique de l'onde ψ^{in} fut utilisée dans



(a) $t = 5\ ps$ (b) $t = 10\ ps$ (c) $t = 14\ ps$

FIG. 7.31 – Cas Helmholtz avec absorption, coupe de $|\psi|^2$



(a) $t = 5\ ps$ (b) $t = 10\ ps$ (c) $t = 14\ ps$

FIG. 7.32 – Cas Helmholtz avec absorption, coupe de $N_e(x, y)$

le second membre de la condition limite (1.9) donnant

$$\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\psi^{in} = (1+\cos\alpha)\mathrm{i}\sqrt{1-N}\psi^{in}.$$

Cette approche naturelle ne donne toutefois pas de résultats satisfaisants. Le cas test considéré est une propagation d'un faisceau dans un plasma sous-dense avec un angle d'incidence. On observe clairement un saut à l'interface des modèles engendrant par la suite une force pondéromotrice génante.

Une seconde approche consiste à discrétiser la dérivée normale de ψ^{in} par un schéma décentré classique au bord

$$\left(\epsilon\frac{\partial}{\partial n} + i\sqrt{1-N}\right)\psi^{in} = \left[\xi|_0 + i\sqrt{1-N}\right]\psi^{in}$$

où on a noté $\xi|_0 = \dfrac{e^{\mathrm{i}\sqrt{1-N}\frac{\delta x}{\epsilon}} - 1}{\delta x}$.

On remarque alors que la prise en compte d'une discrétisation pour le second membre de la condition limite permet une certaine cohérence avec le schéma numérique utilisé pour le modèle Helmholtz. On constate que le saut à l'interface est très largement réduit.

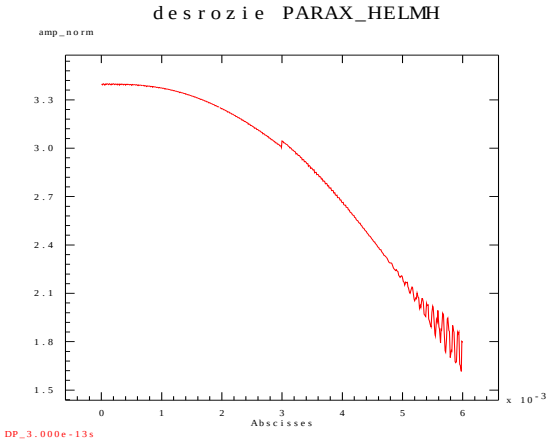


FIG. 7.33 – Raccord des modèles par second membre analytique

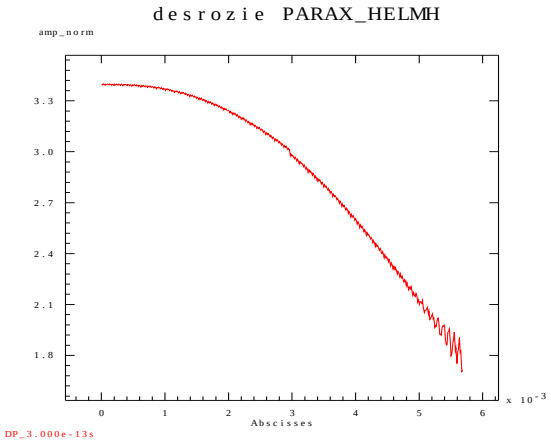


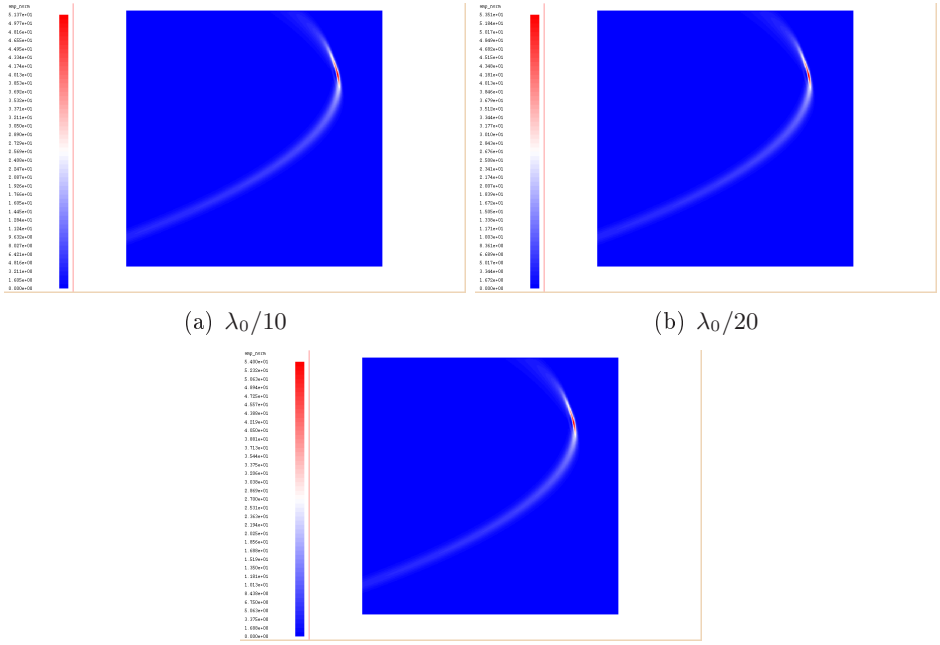
FIG. 7.34 – Raccord des modèles par second membre discrétisé

7.5 Discrétisation de la longueur d'onde

Un point crucial de la précision des solutions obtenues dans les problèmes d'ondes est le nombre de point de discrétisation par longueur d'onde. Dans notre cas, il est aussi crucial de se fixer un nombre de points acceptable. Le critère usuel est d'utiliser au moins dix points par longueur d'onde. On vérifie ici que cela est suffisamment précis dans nos cas.

On s'intéresse à un cas de propagation d'un faisceau près d'une caustique. On compare ici les solutions obtenues par 10, 20 et 30 points par longueur d'onde au temps $t = 2\ ps$.

Les résultats montrent qu'une discrétisation de dix points par longueur d'onde est suffisammet précise.



(a) $\lambda_0/10$ (b) $\lambda_0/20$

(c) $\lambda_0/30$

FIG. 7.35 – Comparaison de discrétisation de , intensité laser $|\psi|^2$, $t = 2\ ps$

7.6 Méthodes de Krylov

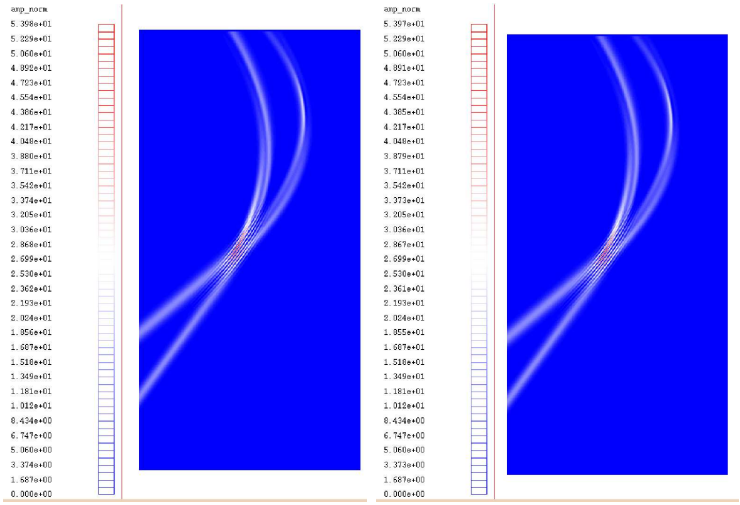
Le domaine considéré est une boîte d'une longueur de $300\lambda_0$ pour une hauteur de $600\lambda_0$. On considère alors un système de 18 millions d'inconnues résolu sur 32 processeurs. On considère un cas de croisement de deux faisceaux avec des angles d'incidences différents dans une zone proche de la caustique. L'idée est de générer rapidement un certains nombre d'iterations afin de permettre une comparaison pertinente des methodes iteratives de Krylov.

La méthode *GMRES* fait réference comme technique itérative pour la résolution des systèmes linéaires non symétriques. Toutefois, la contrainte liée à la forte consommation de mémoire peut être un problème majeure dans notre cas. L'utilisation d'autres méthodes comme *BICGStab* ou *CGS* (voir 51) peut alors s'envisager. Il est intéressant de comparer les comportements de ces différentes méthodes dans notre cas.

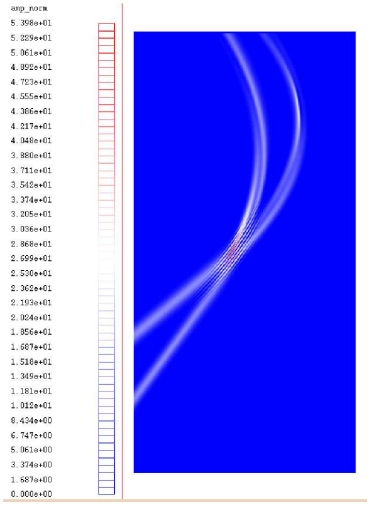
Les résultats obtenus sont similaires. On constate que les méthodes ont également un comportement proche mais que *BICGStab* et *GMRES* sont les plus rapides. En particulier, si un problème mémoire apparaît, on utiliserait alors la méthode *BICGStab*.

7.7 Cas *monospeckle*

On s'intéresse maintenant à la propagation d'un faisceau composé d'un unique *speckle* (ou point chaud) dans un plasma dont la densité croît de manière



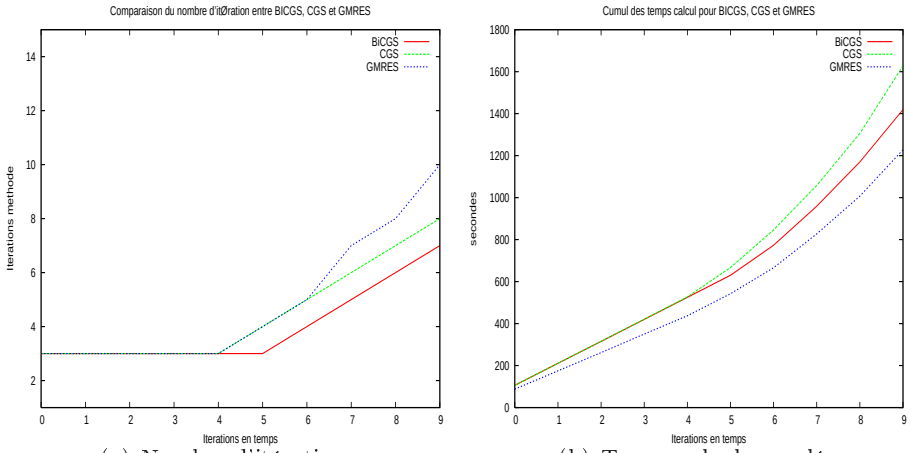
(a) *GMRES* (b) *BICGStab*



(c) *CGS*

FIG. 7.36 – Comparaison des méthodes de Krylov, intensité laser $|\psi|^2$, $t = 1\ ps$

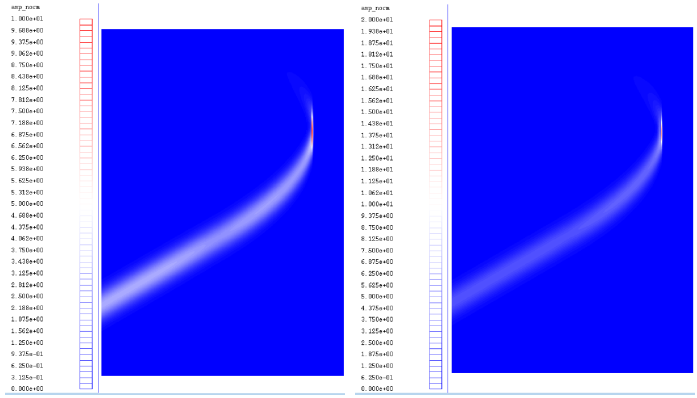
linéaire jusqu'à la densité critique. Le faisceau a un angle d'indidence de 30°. On considère le domaine de simulation $\mathcal{D}$ de longueur $L_x = 700\lambda_0$ et de hauteur $L_y = 1000\lambda_0$. On note $\Omega = [x_f, L_x] \times [0, L_y]$ où $x_f = 300\lambda_0$. Dans le domaine $\mathcal{D} \setminus \Omega$, la densité est sous-dense, constante égale à 10% de la densité critique et on y utilise le modèle paraxial. Dans le domaine Ω, la densité croît linéairement jusqu'à la densité critique et on y utilise le modèle Helmholtz avec un coefficient d'absorption $\nu = \alpha N_0^2(x)$ où $\alpha = 10^{-3}$. En posant $\Delta_x = \Delta_y = \lambda_0/2$, le maillage *fluide* grossier est alors composée de 2,8 millions de mailles. Pour le maillage *Helmholtz* fin, en posant $\delta_y = \delta_x = \lambda_0/10$, on est amené à considérer 40 millions d'inconnues. Le temps calcul d'une itération de Krylov est de 18,4s sur 128 processeurs. La simulation dure 8 heures.



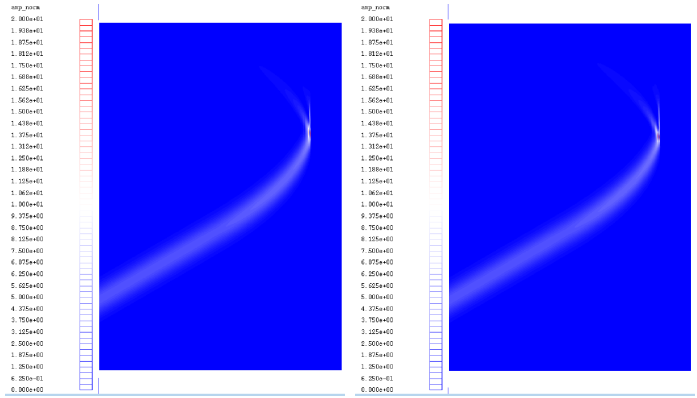
(a) Nombre d'itérations (b) Temps calcul cumulé

FIG. 7.37 – Comparaison de *BICGStab*, *CGS*, *GMRES* au cours du temps

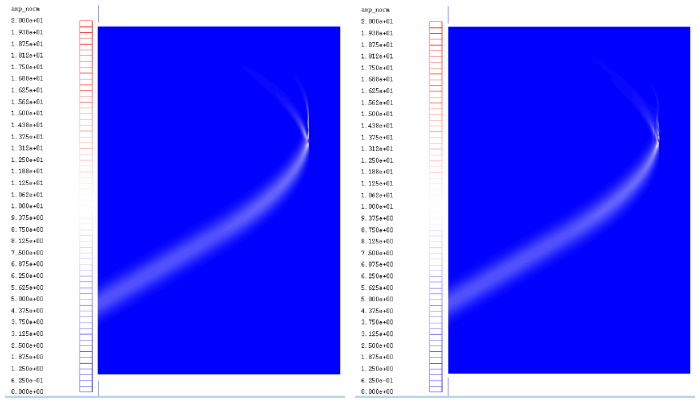
Ces résultats sont nouveaux. Le faisceau se propage tout droit dans la partie où la densité est constante pour ensuite dévier dans la zone proche de la caustique et resortir par le haut de domaine dans la couche *PML*. On observe alors un fort dépôt d'énergie sur la caustique. Tout ceci coïncide avec les résultats de l'optique géométrique. Dans les premiers temps de la simulation, on distingue clairement la fonction d'Airy dans la courbure du faisceau. Ensuite, les fluctuations de l'hydrodynamique jouent activement un rôle sur les perturbations de la propagation du faisceau ; le laser continue de se propager dans la rainure creusée sur la caustique mais il commence à autofocaliser. En fin de simulation, on observe la focalisation du faisceau avant la courbure de direction puis l'éclatement du faisceau. Ce comportement est très similaire du cas de propagation sous-critique. On observe simplement la courbure du faisceau. Le dépôt d'énergie est alors légèrement en amont de la caustique.



(a) $t = 1\ ps$ (b) $t = 2\ ps$



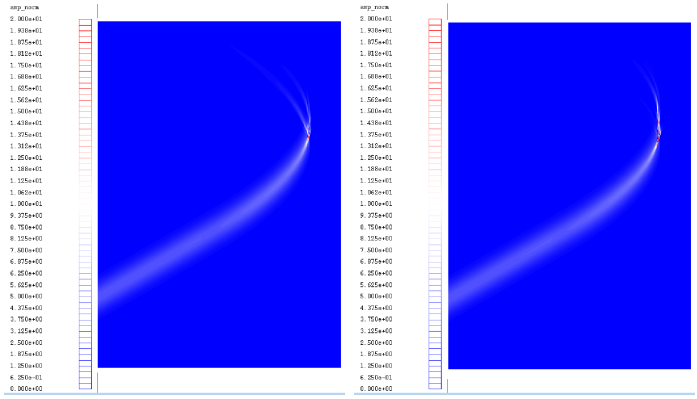
(c) $t = 3\ ps$ (d) $t = 4\ ps$



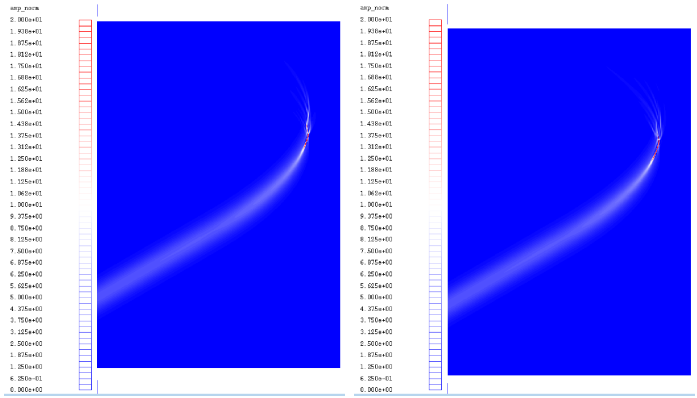
(e) $t = 5\ ps$ (f) $t = 6\ ps$

FIG. 7.38 – Cas monospeckle, intensité laser $|\psi|^2$, $t = 1\ ps$ à $t = 6\ ps$

On constate malgré l'absorption une évolution croissante du nombre d'itération de la méthode de Krylov. En effet, le creusement de la densité est fort près de la caustique.



(a) $t = 7\ ps$ (b) $t = 8\ ps$



(c) $t = 9\ ps$ (d) $t = 10\ ps$

FIG. 7.39 – Cas monospeckle, intensité laser $|\psi|^2$, $t = 7\ ps$ à $t = 10\ ps$

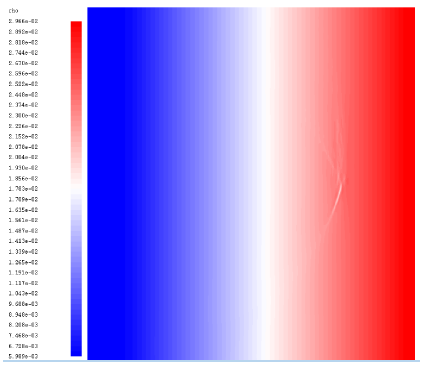
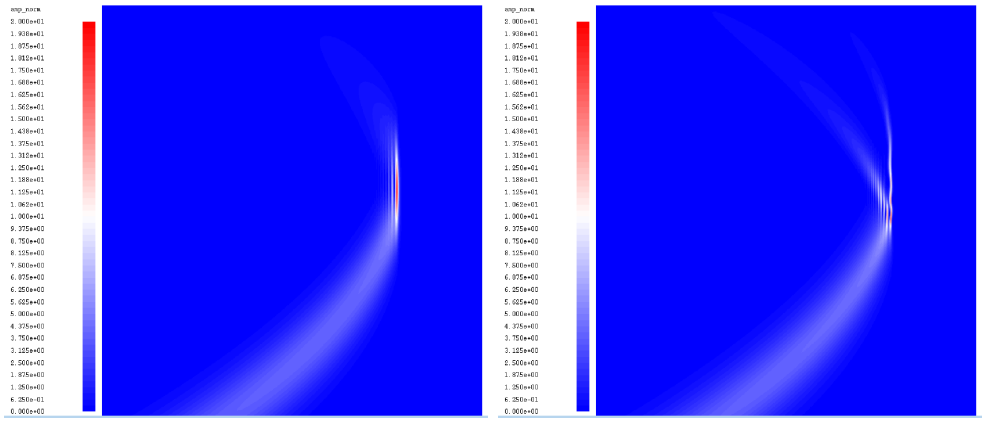


FIG. 7.40 – Cas monospeckle, Densite N_e, $t = 10\ ps$

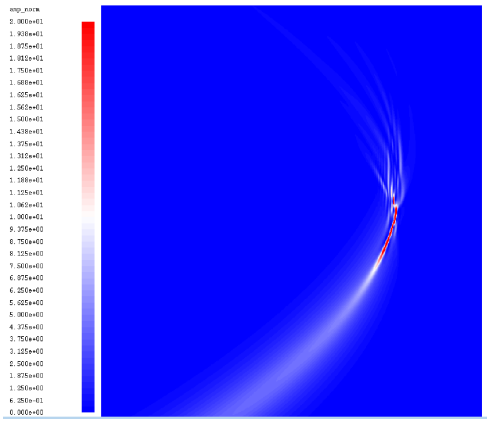
7.8 Cas *multispeckle*

7.8.1 Cas 4 speckles

On s'intéresse maintenant à la propagation d'un faisceau composé de quatre speckles dans le même cadre que précédemment. Le but est ici de voir si comme



(a) $t = 1\ ps$ (b) $t = 5\ ps$



(c) $t = 10\ ps$

FIG. 7.41 – Cas monospeckle, intensité laser $|\psi|^2$, $t = 1\ ps$, $t = 5\ ps$, $t = 10\ ps$

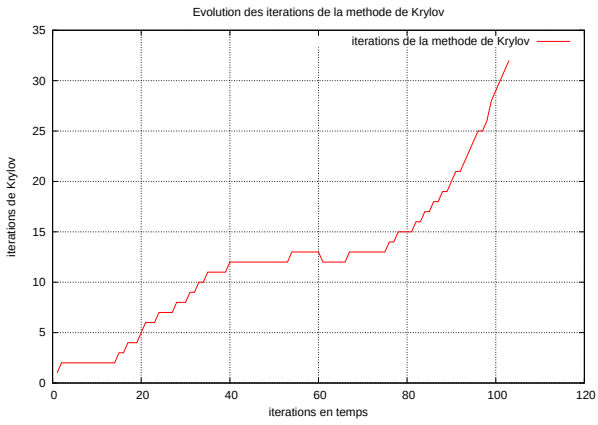
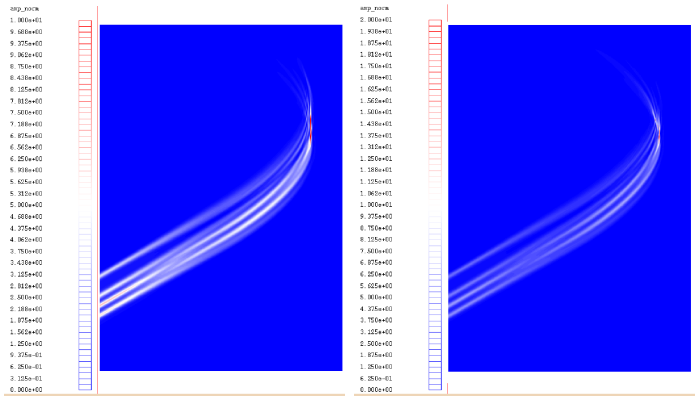


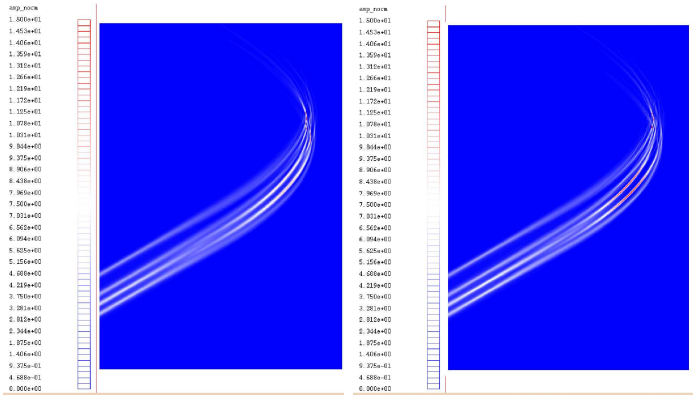
FIG. 7.42 – Evolution des itérations de Krylov en temps, cas monospeckle

dans le cas paraxial, l'interaction des speckles entre eux engendre des instabilités dans la propagation du laser.

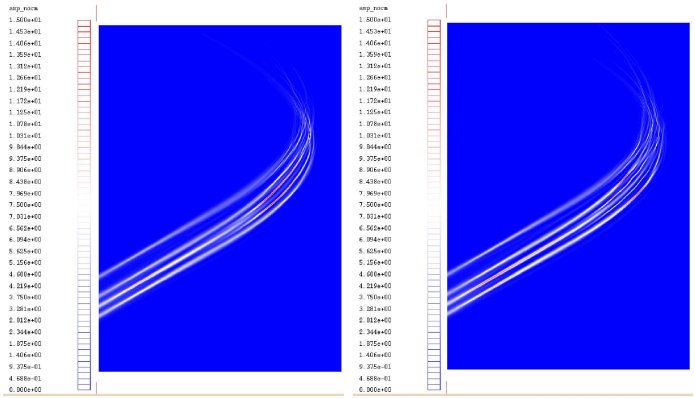


(a) $t = 1\ ps$ (b) $t = 2\ ps$

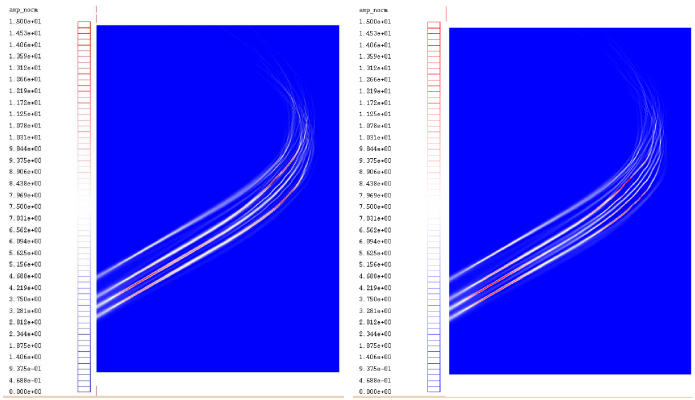
FIG. 7.43 – Cas 4 speckles, intensité laser $|\psi|^2$, $t = 1\ ps$ et $t = 2\ ps$



(a) $t = 3\ ps$ (b) $t = 4\ ps$



(c) $t = 5\ ps$ (d) $t = 6\ ps$



(e) $t = 7\ ps$ (f) $t = 8\ ps$

FIG. 7.44 – Cas 4 speckles, intensité laser $|\psi|^2$, $t = 3\ ps$ à $t = 8\ ps$

Comme pour le cas monospeckle, le faisceau se propage dans le plasma, change de direction près de la caustique et ressort par le haut du domaine. Dans les premiers temps, le faisceau se comporte comme le faisceau monospeckle. Le dépôt d'énergie est là où attendu. Au fur et à mesure du temps, des points chauds dus à l'autofocalisation se forment. Pour une absorption $\nu = 0.002$, on voit que l'interaction des sepckles les uns avec les autres crée de grandes perturbations

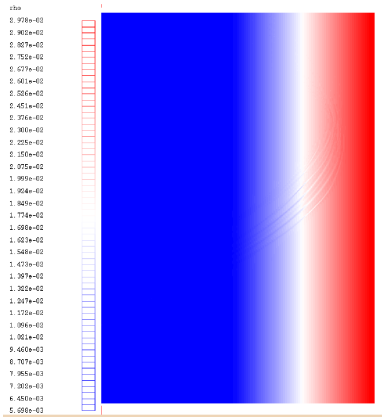


FIG. 7.45 – Cas 4 speckles, Densité N_e, $t = 8\ ps$

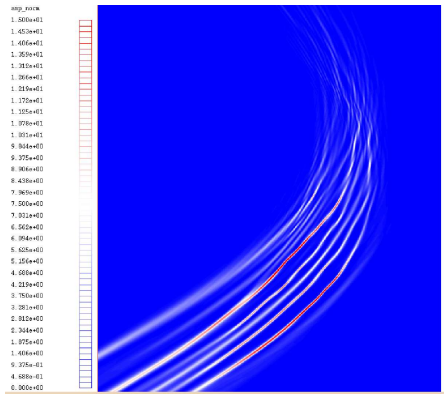


FIG. 7.46 – Cas 4 speckle, intensité laser $|\psi|^2$, $t = 8\ ps$

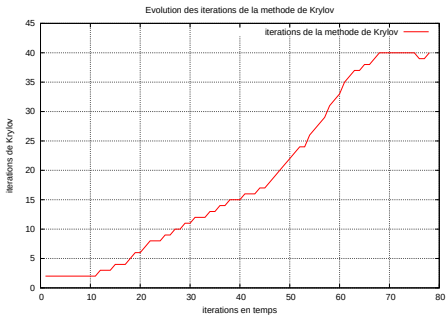
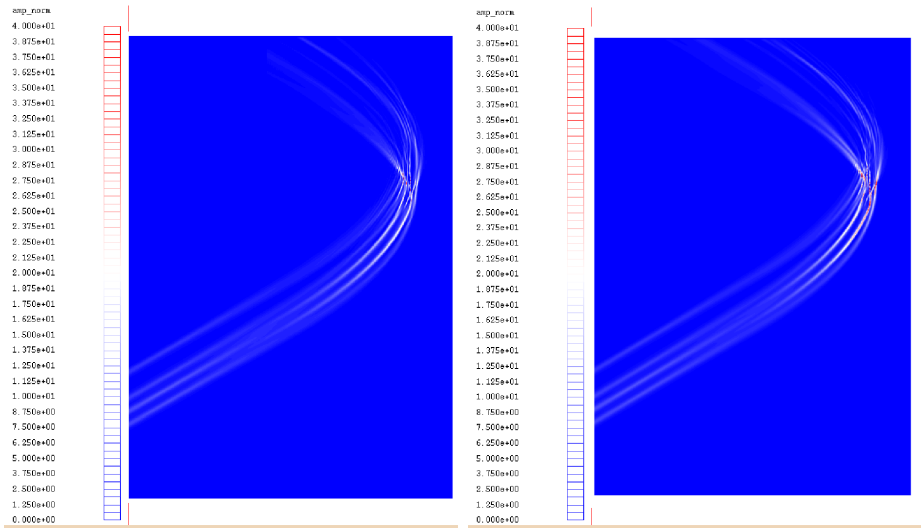


FIG. 7.47 – Evolution des itérations de Krylov en temps, Parax vs Parax + Helmh

dans le dépôt d'énergie quand l'hydrodynamique commence à jouer.

7.8.2 Validation du couplage Paraxial / Helmholtz

C'est pour ce cas qu'on a choisi de comparer la solution obtenue avec le modèle paraxial dans la zone loin de la caustique couplé au modèle Helmholtz avec la solution obtenue qu'avec le modèle Helmholtz pour $\nu = 0$. En effet, ce cas est suffisamment complexe pour qu'un problème de couplage puisse engendrer de grandes différences. Pour le modèle Helmholtz seul, le maillage *Helmholtz* fin est alors composé de 70 millions d'inconnues.



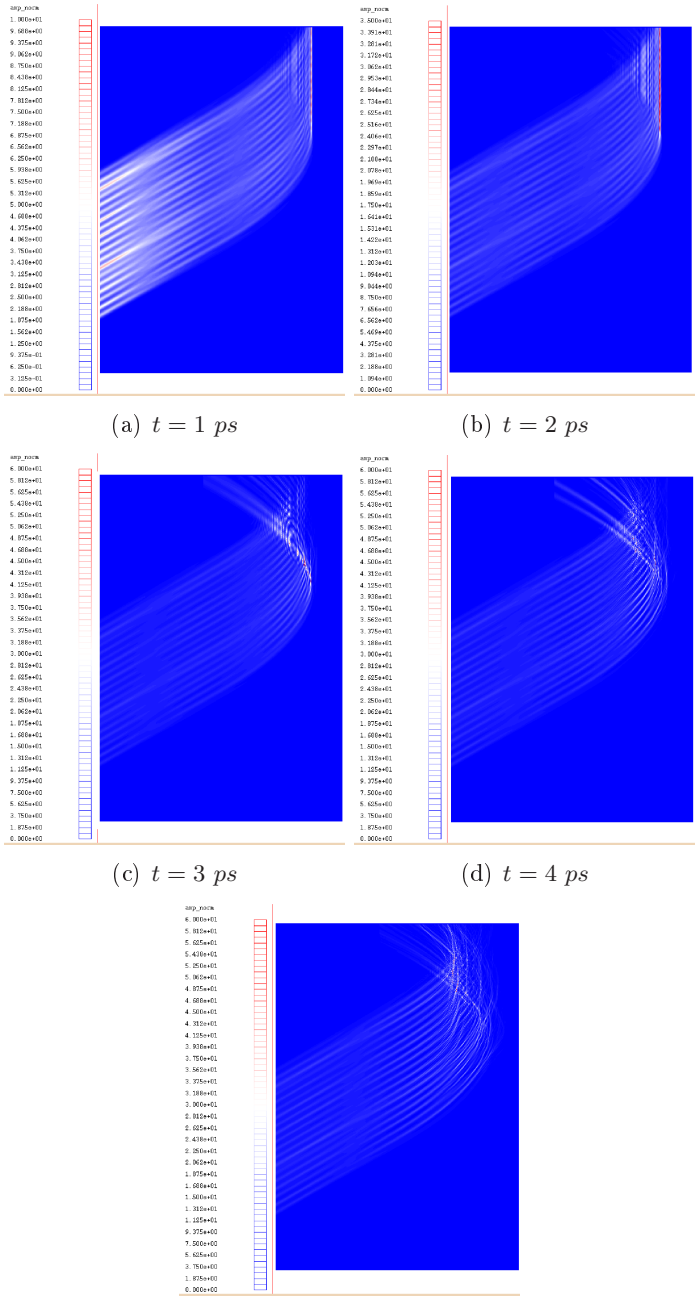
(a) Paraxial + Helmholtz (b) Helmholtz

FIG. 7.48 – Comparaison de $|\psi|^2$par Parax + Helmh / Paraxial, $t = 2\ ps$

La figure (7.48) montre l'intensité laser des deux simulations à 2 ps. Sans absorption physique dans les modèles, les solutions ont très vite creusées la densité et les faisceaux ont largement diffracté. On constate que les solutions sont globalement les mêmes. Toutefois, l'intensité laser de la solution avec seulement le modèle Helmholtz est plus forte que celle obtenue par le couplage des modèles. Cela reste très satisfaisant au vu de la difficulté du cas.

7.8.3 Cas 20 speckles

On place maintenant dans le cas d'un faisceau entrant composé de 20 speckles avec un angle d'incidence de 30°.



(a) $t = 1\ ps$ (b) $t = 2\ ps$

(c) $t = 3\ ps$ (d) $t = 4\ ps$

(e) $t = 5\ ps$

FIG. 7.49 – Cas 20 speckles, intensité laser $|\psi|^2$, $t = 1\ ps$ à $t = 5\ ps$

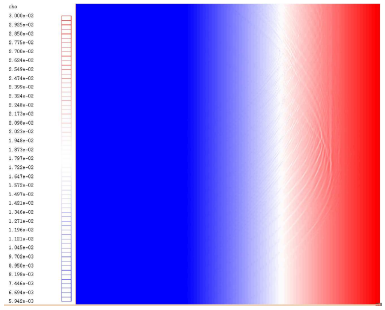
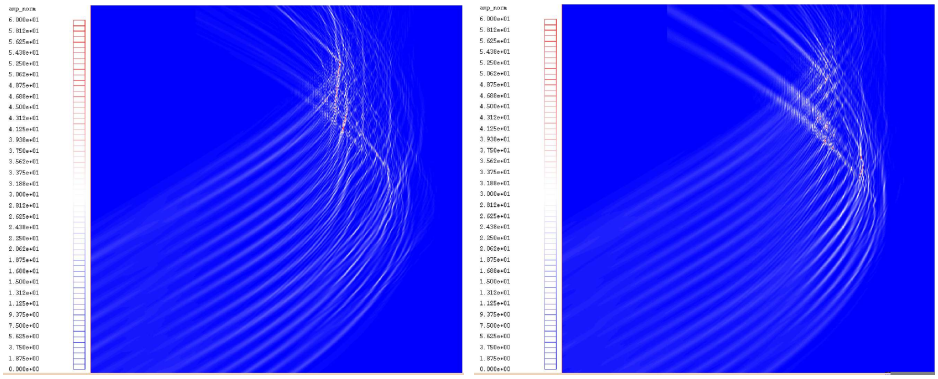


FIG. 7.50 – Cas 20 speckles, Densité N_e, $t = 5\ ps$



(a) $t = 4\ ps$ (b) $t = 5\ ps$

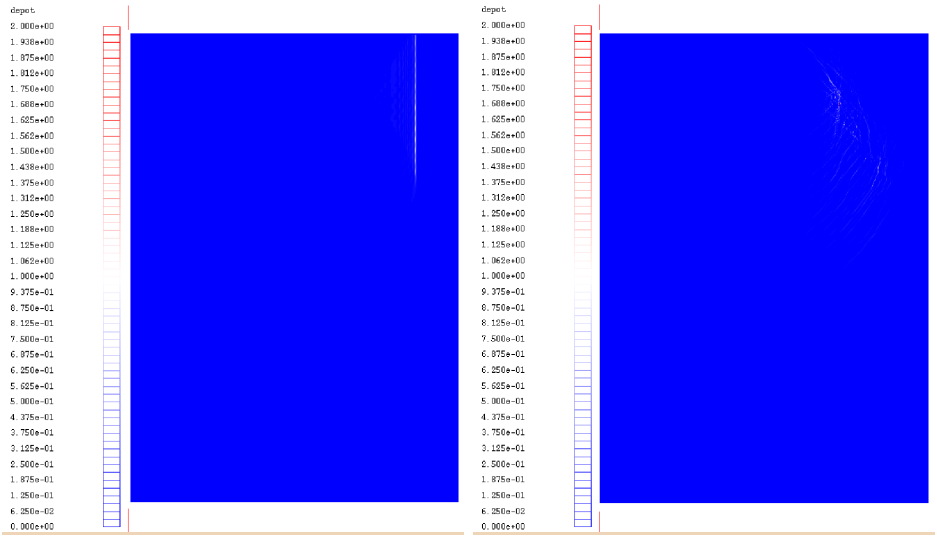
FIG. 7.51 – Cas 20 speckles, intensité laser $|\psi|^2$, $t = 4\ ps$ et $t = 5\ ps$

L'intérêt de ce type de simulation près des caustiques est de pouvoir prendre en compte l'éclatement et l'interaction des faisceaux qui peuvent déplacer les dépôts d'énergie en comparaison des résultats obtenus par des calculs avec des méthodes type *ray-tracing*.

On constate qu'en début de simulation (a), la caustique est clairement visible à l'endroit de courbure du faisceau. C'est le type de résultats obtenus par les méthodes classiques de *ray-tracing*. Il est également intéressant de remarquer (b) l'interaction des faisceaux arrivant vers la caustique avec ceux sortant du domaine. Le dépôt d'énergie est uniforme dessus (7.52). Par contre, dès que l'hydrodynamique commence à interagir (c), (d) et (e), de fortes perturbations dans la trajectoire du faisceau apparaissent. A $t = 5\ ps$, une multitude de petits points chaud parsème les zones d'interaction des faisceaux et le dépôt d'énergie est alors très différent de celui à $t = 2\ ps$.

7.9 Cas physique réaliste ; déflection du laser

On présente la propagation d'un faisceau composé d'une vingtaine de speckles dans un plasma dont la densité croît de manière linéaire jusqu'à la densité



(a) $t = 4\ ps$ (b) $t = 5\ ps$

FIG. 7.52 – Cas 20 speckles, Dépôt d'énergie $|\psi|^2 N_e^2$, $t = 2\ ps$ et $t = 5\ ps$

critique. Le faisceau a un angle d'indidence de 25°. L'intérêt de cette simulation est d'observer la déflection du faisceau lorsque le plasma a une vitesse tranverse importante. En effet, les méthodes classiques de *ray tracing* ne peuvent prendre en compte la vitesse du fluide. On peut ainsi observer l'erreur de ce type de méthode dans ces cas réalistes. On considère le domaine de simulation $\mathcal{D}$ de longueur $L_x = 2000\lambda_0$ et de hauteur $L_y = 2000\lambda_0$. Le domaine de simulation du modèle Helmholtz est $\Omega = [x_f, L_x] \times [0, L_y]$ où $x_f = 1000\lambda_0$. Dans le domaine $\mathcal{D} \setminus \Omega$ où on y utilise le modèle paraxial, la densité est sous-dense, constante égale à 10% de la densité critique. On y applique le modèle Helmholtz avec un coefficient d'absorption $\nu = \alpha N_0^2(x)$ où $\alpha = 2 \times 10^{-5}$. La vitesse transverse est de l'ordre de $4 \times 10^7 cm.s^{-1}$. En posant $\Delta_x = \Delta_y = \lambda_0/2$, le maillage *fluide* grossier est alors composée de 16 millions de mailles. Pour le maillage *Helmholtz* fin, en posant $\delta_y = \delta_x = \lambda_0/10$, on est amené à considérer 200 millions d'inconnues. Le temps calcul d'une itération de Krylov est de 348 s sur 256 processeurs. La simulation dure plusieurs dizaines heures effectuées grâce à un mécanisme de protection-reprise.

Remarque On sait que la simulation doit aller jusqu'à des temps très longs, de l'ordre de 20 ps, pour permettre l'observation de la déflection du faisceau à l'echelle macroscopique. Notons qu'en début de simulation, il est simplement nécessaire que le faisceau creuse la densité. Pour gagner du temps calcul, on peut donc ne calculer la solution du problème Helmholtz qu'un pas de temps sur trois jusqu'à 10 ps.

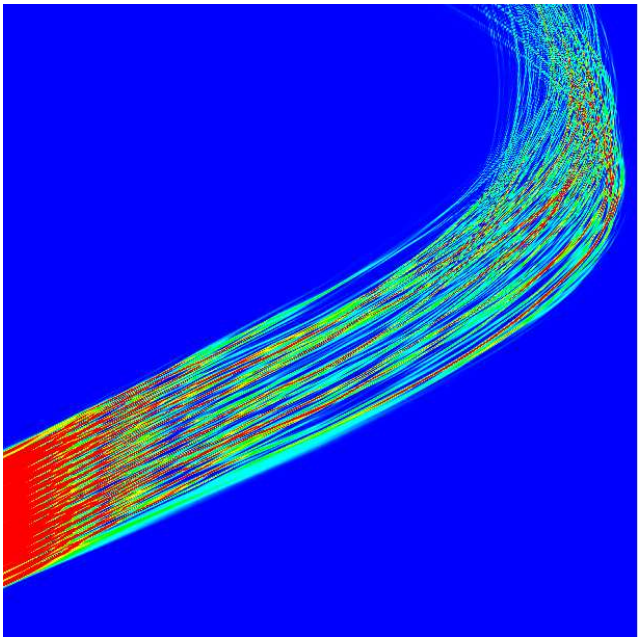


FIG. 7.53 – Cas avec vitesse transverse, intensité laser $|\psi|^2$, $t = 22\ ps$

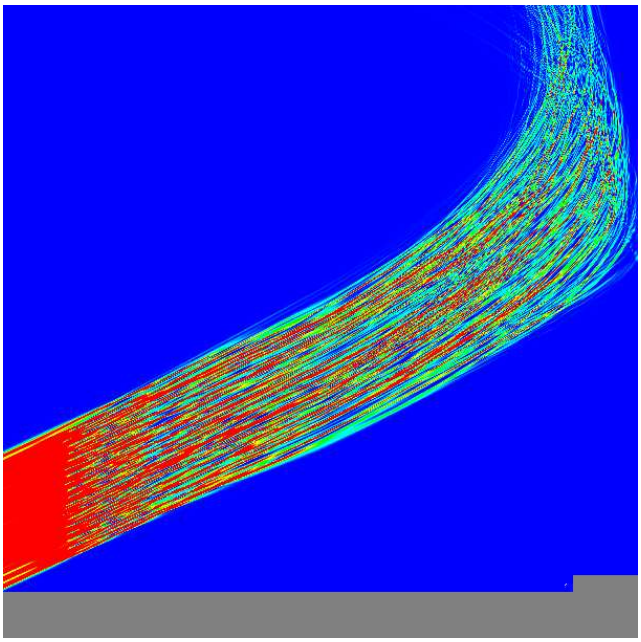


FIG. 7.54 – Cas sans vitesse transverse, intensité laser $|\psi|^2$, $t = 22\ ps$

On constate que dans le cas avec vitesse transverse, le faisceau est légérement dévié dans le sens de déplacement du fluide. C'est le résultat attendu. Cependant, le phénomène n'est pas très marqué. Il faudrait donc pousser la simulation

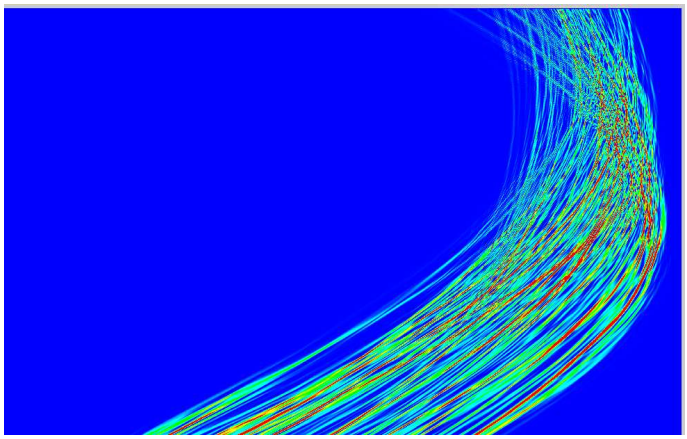


FIG. 7.55 – Cas avec vitesse transverse, zoom, intensité laser $|\psi|^2$, $t = 22\ ps$

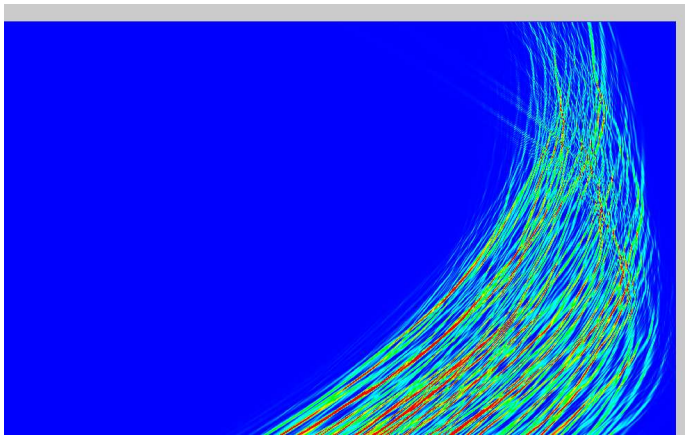


FIG. 7.56 – Cas sans vitesse transverse, zoom, intensité laser $|\psi|^2$, $t = 22\ ps$

encore plus loin pour quantifier l'erreur commise sur le dépôt d'énergie par des méthodes type *ray-tracing*. Une simulation réaliste aussi longue demande énormément de moyen de calcul et on atteint les limites de la résolution sur 256 processeurs. Il devient alors nécessaire d'utiliser 512 processeurs pour être plus efficace. Ces simulations sont maintenent réalisables et actuellement en cours avec de telles ressources.

Conclusion et perspectives

On a présenté dans cette thèse une méthode pour la résolution d'un problème de propagation d'ondes avec couches absorbantes *PML* dans un plasma dense présentant de fortes variations dans une direction privilégiée. Les contraintes numériques étant très fortes, on a mis en évidence le besoin d'avoir une stratégie basée sur l'utilisation d'un solveur rapide de type Réduction Cyclique. De plus, afin d'éviter les problèmes de réflexion aux bords, on a choisi d'utiliser des couches absorbantes *PML*. Une stratégie particulière de préconditionnement pour les variations de densité a été alors employée et intégrée dans une méthode de Krylov. Tout ceci représente finalement un arsenal d'outils performants qu'il a fallu faire cohabiter avec la prise en compte d'une décomposition de domaine. Cette méthode de résolution fut intégrée dans le code de production *HERA*. La parallélisation est assurée par le standard *MPI* et le *multithreading*. Les résultats obtenus sont nouveaux et permettent maintenant l'étude des phénomènes de perturbation et dépôt d'énergie près de la caustique.

Pour gagner du temps calcul, une première perspective peut être d'intégrer une décomposition de domaine plus générale permettant de diviser le domaine aussi dans les deux directions. On réduirait ainsi le facteur de stockage mémoire et la taille du produit matrice-matrice à effectuer. En contre partie, on augmenterait le nombre d'itérations de la méthode de Krylov. Il pourrait être intéressant de le quantifier. Remarquons aussi qu'avec l'arrivée des prochaines machines dans le cadre du projet *Tera*, l'espace mémoire n'est plus vraiment un problème. On pourrait aussi envisager une méthode afin d'isoler les zones où la solution est nulle afin de ne pas les calculer. Ceci peut être intégré directement dans la méthode de Krylov ou alors par une décomposition de domaine générale. Toutefois, cette perspective pose le problème d'équilibrage des charges entre les processus. Disposant d'un solveur rapide pour l'équation d'Helmholtz, il est maintenant possible d'étudier les phénomènes de retrodiffusion Brillouin nécessitant la résolution d'un couplage avec une équation des ondes temporelle supplémentaire. Pour finir, le passage au 3D devra également s'envisager.

A

Rappels de différences finies

On considère l'équation d'Helmholtz

$$\epsilon^2 \Delta u + (1 - N)u = 0 \quad \text{dans} \quad \Omega \tag{A.1}$$

On discrétise cette équation par différences finies en numérotant les inconnues de gauche à droite et de bas en haut sur un maillage régulier à (n_x, n_y) points dans les directions (x, y). On indice par i les lignes du maillage et par j les colonnes. On applique un schéma centré classique pour discrétiser l'opérateur Laplacien. Pour u suffisamment régulière, ce schéma est obtenu en sommant les développements de Taylor suivants

$$u(x + \delta x) = u(x) + \delta x \frac{\partial u}{\partial x} + \frac{\delta x^2}{2} \frac{\partial^2 u}{\partial x^2} + \frac{\delta x^3}{6} \frac{\partial^3 u}{\partial x^3} + O(\delta x^4) \tag{A.2}$$

et

$$u(x - \delta x) = u(x) - \delta x \frac{\partial u}{\partial x} + \frac{\delta x^2}{2} \frac{\partial^2 u}{\partial x^2} - \frac{\delta x^3}{6} \frac{\partial^3 u}{\partial x^3} + O(\delta x^4). \tag{A.3}$$

En notant $u(x + \delta x) = u_{i+1}$, on trouve alors

$$\left. \frac{\partial^2 u}{\partial x^2} \right|_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\delta x^2} + O(\delta x^2).$$

En deux dimensions, l'équation (A.1) peut se discrétiser ainsi :

$$\epsilon^2 \left[\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta y^2} \right] + (1 - N_{i,j})u_{i,j} = 0 \tag{A.4}$$

pour $1 \leq i \leq n_x$ et $1 \leq j \leq n_y$. On obtient alors une matrice tridiagonale par blocs dont chaque bloc correspond à la discrétisation d'une ligne du maillage.

En utilisant (A.2) et (A.3), on a également les relations à l'ordre un pour la dérivée

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{u_{i+1} - u_i}{\delta x} + O(\delta x) \tag{A.5}$$

et

$$\left. \frac{\partial u}{\partial x} \right|_i = \frac{u_i - u_{i-1}}{\delta x} + O(\delta x). \tag{A.6}$$

On a également l'ordre deux

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+1} - u_{i-1}}{2\delta x} + O(\delta x^2). \tag{A.7}$$

En utilisant (A.2), l'ordre deux peut également s'écrire

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+1} - u_i}{\delta x} - \frac{\delta x}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i + O(\delta x^2).$$

En insérant l'équation (A.1), on obtient

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_{i+1} - u_i}{\delta x} + \frac{\delta x}{2}\left(\left.\frac{\partial^2 u}{\partial y^2}\right|_i + \frac{1 - N_i}{\epsilon^2}\right) + O(\delta x^2). \tag{A.8}$$

On peut obtenir des relations similaires en utilisant le développement (A.3)

$$\left.\frac{\partial u}{\partial x}\right|_i = \frac{u_i - u_{i-1}}{\delta x} - \frac{\delta x}{2}\left(\left.\frac{\partial^2 u}{\partial y^2}\right|_i + \frac{1 - N_i}{\epsilon^2}\right) + O(\delta x^2). \tag{A.9}$$

B

Calcul analytique de valeurs et vecteurs propres

On est intéressé ici par l'obtention analytique des valeurs et vecteurs propres d'une matrice d'ordre $n+1$

$$T = \begin{bmatrix} \alpha & 1 & & & \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ & & & 1 & \alpha \end{bmatrix},$$

où α peut être complexe. Pour α réel, voir par exemple [47]. On s'intéresse ici aux cas particuliers $\alpha = 0$ et $\alpha = 1$.

Soient λ une valeur propre de T associée au vecteur propre v,

$$Tv = \lambda v.$$

B.1 Valeurs propres

On cherche les vecteurs propres sous la forme

$$v_j = Ae^{(j-1)\Phi} + Be^{-(j-1)\Phi}.$$

Pour $j \neq 1$ et $j \neq n+1$, on a

$$v_{j-1} + v_{j+1} = \lambda v_j,$$

ce qui donne facilement

$$\lambda = e^{\Phi} + e^{-\Phi}.$$

Alors, en regardant pour $j = 1$ et $j = n+1$, on a

$$\alpha(A + B) = Ae^{-\Phi} + Be^{\Phi}$$

et

$$A(\alpha e^{n\Phi} - e^{(n+1)\Phi}) + B(\alpha e^{-n\Phi} - e^{-(n+1)\Phi}) = 0.$$

On a alors un système linéaire 2×2 homogène

$$\begin{pmatrix} \alpha - e^{-\Phi} & \alpha - e^{\Phi} \\ e^{n\Phi}(\alpha - e^{\Phi}) & e^{-n\Phi}(\alpha - e^{-\Phi}) \end{pmatrix} \begin{pmatrix} A \\ B \end{pmatrix} = 0.$$

Pour avoir une solution non triviale, le déterminant doit être nul. Ainsi,

$$(\alpha - e^{-\Phi})^2 e^{-n\Phi} - (\alpha - e^{\Phi})^2 e^{n\Phi} = 0. \tag{B.1}$$

1. Si on a $\alpha = 0$, on obtient immédiatement

   $$e^{-(n+2)\Phi} = e^{(n+2)\Phi}.$$

   Alors, $\Phi = 0$ est une solution menant à $\lambda = 2$. Autrement,

   $$e^{2(n+2)\Phi} = 0,$$

   et Φ est imaginaire pur, $\Phi = \mathrm{i}\theta$, ce qui donne

   $$\sin(2(n+2)\theta) = 0, \quad \cos(2(n+2)\theta) = 1.$$

   La solution est alors

   $$\theta = \frac{j\pi}{n+2}, \quad j = 1, \ldots$$

   Les valeurs propres de T sont alors

   $$\lambda_j = 2\cos\left(\frac{(j+1)\pi}{n+2}\right), \quad j = 0, \ldots$$

2. Si on a $\alpha = 1$ alors

   $$(1 - e^{-\Phi})^2 e^{-n\Phi} - (1 - e^{\Phi})^2 e^{n\Phi} = 0,$$

   ce qui donne

   $$e^{-(n+2)\Phi}(e^{\Phi} - 1)^2 - e^{n\Phi}(e^{\Phi})^2 = 0.$$

   Ainsi, on a $e^{\Phi} = 1$ ou $e^{-(n+2)\Phi}) - e^{n\Phi} = 0$ ce qui donne

   $$e^{-(n+1)\Phi}) = e^{(n+1)\Phi}).$$

   Φ est imaginaire pur et

   $$\theta = \frac{j\pi}{n+1}, \quad j = 1, \ldots$$

   Les valeurs propres de T sont alors

   $$\lambda_j = 2\cos\left(\frac{j\pi}{n+1}\right), \quad j = 0, \ldots$$

3. Dans le cas général, on peut exprimer la solution de (B.1), α, comme une fonction de Φ. Toutefois, c'est l'inverse de la fonction qui nous intéresse. On a deux solutions

   $$\alpha_+ = \frac{e^{-(\frac{n}{2}+1)\Phi} - e^{(\frac{n}{2}+1)\Phi}}{e^{-\frac{n}{2}\Phi} - e^{\frac{n}{2}\Phi}} \alpha_- = \frac{e^{-(\frac{n}{2}+1)\Phi} + e^{(\frac{n}{2}+1)\Phi}}{e^{-\frac{n}{2}\Phi} + e^{\frac{n}{2}\Phi}}$$

   Il est clair que quand α est complexe alors Φ doit être complexe et non réel ou purement imaginaire.

B.2 Vecteurs propres

On recherche maintenant les vecteurs propres. On peut choisir $A = 1$ ce qui donne

$$B = \frac{\alpha - e^{-\Phi}}{e^{\Phi} - \alpha}$$

Ainsi, les composantes du vecteur propre sont

$$v_j = e^{(j-1)\Phi} + \frac{\alpha - e^{-\Phi}}{e^{\Phi} - \alpha} e^{-(j-1)\Phi}.$$

A un facteur près, on peut réécrire comme

$$e^{(j-\frac{1}{2})\Phi} + \frac{\alpha e^{\Phi} - 1}{e^{\Phi} - \alpha} e^{-j+\frac{1}{2})\Phi}.$$

1. Quand $\alpha = 0$, à un facteur près, on obtient

   $$e^{j\Phi} - e^{-j\Phi},$$

   et $\Phi = \mathrm{i}\theta$, ainsi les composantes des vecteurs propres sont proportionnelles à

   $$v_j = \sin(j\theta).$$

2. Quand $\alpha = 1$, on a

   $$e^{(j-\frac{1}{2})\Phi} - e^{-j+\frac{1}{2})\Phi}.$$

   Ceci donne

   $$v_j = \cos\left((2j-1)\frac{\theta}{2}\right).$$

3. Quand α est complexe, on a

   $$v_j = e^{(j-\frac{1}{2})\Phi} + \beta e^{-j+\frac{1}{2})\Phi},$$

   où β est un nombre complexe.

C

Encapsulation du *multithreading* en C++

Tout d'abord, rappelons que les très connus outils de parallélisation *MPI* et *OpenMP* reposent sur le procédé de *multithreading*. Pour *MPI* de gros efforts de performance ont été fait par les constructeurs. C'est ainsi devenu un standard incountournable de la programmation parallèle. Toutefois, reposant sur la qualité du réseau d'interconnexion, on peut vouloir limiter son utilisation dans des configuration non propriétaire par le *multithreading*. Pour *OpenMP*, sa simplicité d'utilisation est un argument qui l'a rendu très populaire au sein de la communauté scientifique. Toutefois, les performances ne sont pas toujours au rendez-vous et le coté hermétique de cette bibliothèque n'aide pas à la compréhension du problème.

Le *multithreading* est un outil qu'on qualifie souvent de bas niveau. Dialoguant directement avec le système, la portabilité n'est pas assurée. C'est une raison motivant parfois l'utilisation d'alternative comme par exemple*OpenMP*. Précisons que tous les systèmes d'exploitation ont une gestion des threads. On utilise la bibliothèque *POSIX pthread* disponible sur Linux, Unix ou encore Windows.

On a montré que la programmation par *multithreading* demande une attention particuliere quant à son utilisation. La grain des tâches à paralléliser est important et quelques notions d'architecture sont parfois nécessaires pour comprendre et optimiser les performances. D'un point de vue informatique, des fonctions de *pthread* permettent de faire des appels sytème pour la gestion des threads, verrous et sémaphores. Typiquement, on fournit à un thread une fonction (file d'execution) qu'il execute à sa creation. L'execution terminée, le thread se libère. On voit bien la difficulté d'une utilisation générique.

On présente ici une manière proposée par Pascal Havé [48] particulièrement élégante et pratique d'encapsuler le *multithreading* par le langage multiparadigme C++. Le *multithreading* devient alors un concept qu'on affecte à des objets par polymorphisme. On donne le caractère générique à ce concept pour plus de souplesse. Ainsi, le *multithreading* devient transparent pour le developpeur.

```
#include <pthread.h>

//! Virtual Runnable Function
//! Used as template independent virtual class
class Runnable {
  public:

  //! Destructor
  virtual ~Runnable(void) { }

  //! Run function
  virtual void run(void) = 0;
};

extern "C" {

  //! C function called for starting a thread class
  inline void * thread_function(void * t) {

    reinterpret_cast<Runnable *>(t)->run(); // Appel par virtual
    pthread_exit(NULL);

    return NULL;
  }
}

//! Thread Proxy class
template<typename T>
class FonctionThread : public Runnable {
  private:

  T & f;

  public:

  FonctionThread(T * const t) : f(*t) { }

  ~FonctionThread(void) { }

  inline void run(void) { f.run(); }
};

//! Main Thread class
class MultiThread : public Runnable {
  private:

  pthread_t thread;
```

```
  public:

  MultiThread(void) { }

  virtual ~MultiThread(void) { join(); }

  void start() {
    pthread_create(&thread,NULL,thread_function,
    new FonctionThread<MultiThread>(this));
  }

  inline void join() const {
    void * ret;
    pthread_join(thread,&ret);
  }
};
```

Par exemple, le code suivant écrit « Hello ! » par *multithreading*

```
class A : public Thread {
  private:

  char* s;

  public:

  A(char* c) { s = c; }

  void run(){ std::cout << "Hello " << s << "!\n"; }
};

int main(int argc, char* argv[]) {

  A a(argv[1]);

  a.start();

  return 0;
}


> g++ test.cpp -pthread -D__REENTRANT
> ./a.out Sylvain
Hello Sylvain!
```

On comprend bien que par le jeu des encapsulations, surcharges et héritages, on arrive simplement à créer des objets comme des opérateurs alégbriques de

base sur des vecteurs ou des matrices. On peut par exemple encapsuler certaines fonctions de la bibliothèques *LAPACK* pour les rendent *multithread*.

Bibliographie

[1] F. Walret, G. Riazuelo, G. Bonnaud *Propagation in a Plasma of a Laser Beam smoothed by longitudinal spectral Dispersion, Phys. Plasmas, vol. 10, pp. 811, 2003*

[2] C. H. Still, R. L. Berger, A. B. Langdon, E. A. Williams *Three-Dimensional Non-linear Hydro Code to study laser-Plasma interaction, Technical Report UCRL 105821-96-4, Lawrence Livermore Nat.Lab., 1996*

[3] R. Sentis *Mathematical Models for Laser-Plasma Interaction, ESAIM : Math. Modeling and Num. Analysis, vol. 39, pp. 275, 2005*

[4] W. L. Kruer *Physics of Laser-Plasma Interaction, Addinson-Wesley, Redwood, 1988*

[5] J. D. Lindl, P. Amendt P. and al., *The Physics Basis for Ignition using Indirect Drive Targets, Phy. Plasmas, vol. 11, pp. 339, 2004*

[6] M. Doumic, F. Golse, R. Sentis, *Propagation laser paraxiale en coordonnées obliques : équation d'advection-Schrödinger, Note C. R. Ac. Sciences, Paris, série I, vol. 336, pp. 23, 2003*

[7] M. R. Dorr, F. X. Garaizar, J. A. Hittinger *Simuation of laser-Plasma filamentation, J. Comp. Phys., vol. 17, pp.233, 2002*

[8] P. Ballereau, M. Casanova, F. Duboc, D. Dureau, H. Jourdren, P.Loiseau, J. Metral, O. Morice, R. Sentis *Coupling Hydrodynamics with a Paraxial Solver for Laser Propagation, soumis à J. Scient. Comp.*

[9] S. Hüller, Ph. Mounaix, V. T. Tikhonchuk, D. Pesme *Interaction of two neighboring laser beams taking into account the effects of plasma hydrodynamics, Phys. Plasmas, vol. 4, 1997*

[10] V. V. Eliseev, W. Rozmus, V. T. Tikhonchuk, C. E. Capjack *Effect of diffraction on stimulated Brillouin scattering from a single laser hot spot, Phys. Plasmas, vol. 3, 1996*

[11] V. V. Eliseev *Parallelization of three-dimensional spectral laser-plasma interaction code using High Performance Fortran, Comp.Phys., vol. 12, no. 2, 1998*

[12] A. V. Maximov, J. Myatt, W. Seka, R. W. Short, R. S. Craxton *Modeling of stimulated Brillouin scattering near the critical-density surface in the plasmas of direct-drive inertial confinement fusion targets, Phys. Plasmas, vol. 11, no. 6, 2004*

[13] M. R. Amin, C. E. Capjack, P. Frycz, W. Rozmus V. T. Tikhonchuk *Two-dimensional studies of stimulated Brillouin scattering, filamentation and self-focusing instabilities of laser light in plasmas, Phys. Fluids, vol. 5, 1993*

[14] B. N.Parlett *Actua Numerica, pp. 459-491, 1995*

[15] I. S. Dhillon *A New $O(n^2$ Algorithm for the Symetric Tridiagonal Eigenvalue/Eigenvector Problem, PhD thesis, University of California, 1997*

[16] J. H. Wilkinson *: The algebraic eigenvalue problem, Clarendon Press, Oxford, England, 1965*

[17] H. Rutishauser *: Solution of eigenvalue problems with the LRtransformation, Nat. Bur. Standards Appl. Math., 1958*

[18] B. Engquist, A. Majda *: Absorbing boundary conditions for the numerical simulation of waves, Math. Comp. 31, 1977, 629-651*

[19] H. Jourdren, *: HERA hydrodynamics AMR plateform for multiphysics simulation, Proc. of Chicago workshop on AMR methods (sept 2003), Springer Verlag, Berlin, 2004*

[20] F. Collino, *Perfectly matched absorbing layers for paraxial equation. J. Comput. Phys. vol. 131, pp. 164-180, 1997*

[21] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys., vol. 114, no. 1, pp. 185-200, 1994.*

[22] T. Rossi, J. Toivanen *A nonstandard cyclic reduction method, its variants and stability, SIAM J. Matrix Anal. appli., vol. 20, no. 3, pp. 628-645, 1999.*

[23] T. Rossi, J. Toivanen *A parallel fast direct solver for block tridiagonal systems with separable matrices of arbitrary dimension, SIAM J. Sci. Comput., vol. 20, no. 5, pp. 1778-1796, 1999.*

[24] E. Heikkola, T. Rossi, J. Toivanen *Fast solvers for the Helmholtz equation with a perfectly matched layer / an absorbing boundary condition, Fifth World Congress on Computational Mechanics, July 7-12, 2002, Vienna, Austria.*

[25] O. Buneman *A compact non-iterative Poisson solver with sparsity, Technical Report 294, Stanford University Institute for Plasma Research, Stanford, CA, 1969.*

[26] Y. A. Kuznetsov, A. M. Matsokin *On partial solution systems of linear algebraic equations, Soviet J. Numer. Anal. Math. Modelling, no. 4, pp. 453-468, 1989.*

[27] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen *LAPACK Users' Guide, Second Edition, SIAM, Philadelphia, PA, 1995, www.netlib.org/lapack/.*

[28] M. S. Lam, E. E. Rothberg, M. E. Wolf *The cache performance and optimization of blocking algorithms, Computer Systems Laboratory, Stanford University, 1991*

[29] J. A. Bilmes, K. Asanovic, R. Vuduc, S. Iyer, J. Demmel, C. Chin, D. Lam *Optimizing Matrix Multiply using PHiPAC : a Portable, High-performance, ANSI C coding methodology, Lapack Working note, www.icsi.berkeley.edu/ bilmes/phipac/.*

[30] R. C. Whaley, A. Petitet, J. J. Dongarra *Automted empirical optimizations of software and the ATLAS project, Parallel Computing, no. 27, pp. 3-35, 2001.*

[31] K. Goto, R. Van De Geijn *On Reducing TLB Misses in Matrix Multiplication, Technical Report TR-2002-55, The University of Texas at Austin, Department of Computer Sciences, 2002.*

[32] R. W. Hockney *A fast direct solution of Poisson's equation using Fourier analysis, J. Assoc. Comput. Mach., no. 12, pp. 95-113, 1965.*

[33] D. Coppersmith, S. Winograd *On the asymptotic complexity of matrix multiplication, SIAM Journal Comp., no. 11, pp. 472-492, 1980.*

[34] P. N. Swarztrauber *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of the Poisson's equation on a rectangle, SIAM rev., no. 19, pp. 490-501, 1977.*

[35] P. N. Swarztrauber *A direct method for the discrete solution of separable elliptic equations, J. Comp. Physics, vol. 15, no. 1, pp. 46-54, 1973.*

[36] M. Doumic *Etude asymptotique et simulation numérique de la propagation laser en milieu inhomogène, thèse de l'université Paris VII, 2005.*

[37] M. Doumic, F. Golse, R. Sentis *Un modèle paraxial de propagation de la lumière : problème aux limites pour l'equation d'advection Schrödinger en coordonnées obliques, C. R. Acad. Sci. Ser. I Math, vol. 336, pp. 23-28, 2003.*

[38] B. Després *Méthodes de décomposition de domaine pour les problèmes de propagation d'ondes en régime harmonique, thèse de l'université Paris IX Dauphine, 1991.*

[39] M. Gander , F. Magoulès and F. Nataf *Optimized Schwarz Methods without Overlap for the Helmholtz Equation , SIAM J. Sci. Comp., vol. 24, no. 1, pp. 38-60, 2002.*

[40] Y. Saad, M. Schultz *GMRES : A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM J. Sci. Statist. Comput., vol 7, pp. 856-869, 1986.*

[41] R. Fletcher *Conjugate gradient methods for indefinite systems, Numerical Analysis Dundee, G. Watson, ed., Springer Verlag, Berlin, New York, pp. 73-89, 1976.*

[42] P. Sonneveld *CGS : A fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., Vol. 10, No. 1, pp. 36-52, 1989.*

[43] H. A. van der Vorst *BI-CGSTAB : A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., Vol. 13, No. 2, pp. 631-644, 1992.*

[44] B. Stroustrup, *The C++ programming language, Addison-Wesley, third edition, 1998.*

[45] M. Perache *Thèse à paraître, CEA/DAM - INRIA*

[46] V. Danjean *De la réactivité des threads, thèse de doctorat, ENS Lyon, 2004*

[47] H. Shintani *Direct solution of partial difference equation for a rectangle, J. Sci. Hiroshima Univ., ser.A-I, vol. 32, pp. 17-53, 1968.*

[48] P. Havé *EASYMSG : Tools and techniques for an adaptative overlapping in SMPD programming, Math. Mod. Num. An., vol. 36, pp. 863-882, 2002.*