Méthode de décomposition de domaine pour les équations du transport simplifié en neutronique

Bruno Lathuilière

N° d'ordre : 4001

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Bruno LATHUILIÈRE**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

Méthode de décomposition de domaine pour les équations du transport simplifié en neutronique

---

Soutenue le : 9 février 2010

Après avis des rapporteurs :

Frédéric Feyel	Maître de Recherches, ONERA
Robert Roy	Professeur, École polytechnique Montréal

Devant la commission d'examen composée de :

Maxime Barrault	Ingénieur-Chercheur, EDF R&D	Encadrant industriel
Olivier Coulaud ..	Directeur de Recherches, INRIA	Président
Frédéric Feyel	Maître de Recherches, ONERA	Rapporteur
Eric Lorentz	Pilote Stratégique Simulation, EDF R&D	Examinateur
Pierre Ramet	Maître de Conférences, IUT Bordeaux 1 .	Examinateur
Jean Roman	Professeur, ENSEIRB	Directeur de thèse
Robert Roy	Professeur, École polytechnique Montréal	Rapporteur

– 2009 –

Méthode de décomposition de domaine pour les équations du transport simplifié en neutronique

Résumé :

Les calculs de réactivité constituent une brique fondamentale dans la simulation des cœurs des réacteurs nucléaires. Ceux-ci conduisent à la résolution de problèmes aux valeurs propres généralisées *via* l'algorithme de la puissance inverse. A chaque itération, on est amené à résoudre un système linéaire de manière approchée via un algorithme d'itérations imbriquées.

Il est difficile de traiter les modélisations très fines avec le solveur développé à EDF, au sein de la plate-forme *Cocagne*, en raison de la consommation mémoire et du temps de calcul. Au cours de cette thèse, on étudie une méthode de décomposition de domaine de type Schur dual. Plusieurs placements de l'algorithme de décomposition de domaine au sein du système d'itérations imbriquées sont envisageables. Deux d'entre eux ont été implémentés et les résultats analysés. Le deuxième placement, utilisant les spécificités des éléments finis de Raviart-Thomas et de l'algorithme des directions alternées, conduit à des résultats très encourageants. Ces résultats permettent d'envisager l'industrialisation de la méthodologie associée.

Mots clés :

Equations SPn, HPC, parallélisme, décomposition de domaine, méthode de Schur dual, formulation mixte duale, Raviart-Thomas, directions alternées.

Discipline :

Informatique

Equipe-Projet Bacchus commune
à l'INRIA Bordeaux Sud-Ouest,
au PRES de Bordeaux et au CNRS
(LaBRI UMR 5800 et IMB UMR 5251)
351, cours de la libération
33405 Talence Cedex, FRANCE

EDF Recherche et Développement
Département SINETICS
Groupe I23
1, avenue du Général De Gaulle
92140 Clamart, FRANCE

Domain decomposition method for the Simplified Transport Equation in neutronic

Abstract :

The reactivity computations are an essential component for the simulation of the core of a nuclear plant. These computations lead to generalized eigenvalue problems solved by the inverse power iteration algorithm. At each iteration, an algebraic linear system is solved through an inner/outer process.

With the solver *Cocagne* developed at EDF, it is difficult to take into account very fine discretisation, due to the memory requirement and the computation time. In this thesis, a domain decomposition method based on the Schur dual technique is studied. Several placements in the inner/outer process are possible. Two of them are implemented and the results analyzed. The second one, which uses the specificities of the Raviart Thomas finite elements and of the alternating directions algorithm, leads to very promising results. From these results the industrialization of the method can be considered.

Keywords :

SPn equations, HPC, parallelism, domain decomposition, dual Schur method, mixed-dual formulation, Raviart-Thomas, alternating-directions.

Discipline :

Computer science

Equipe-Projet Bacchus commune
à l'INRIA Bordeaux Sud-Ouest,
au PRES de Bordeaux et au CNRS
(LaBRI UMR 5800 et IMB UMR 5251)
351, cours de la libération
33405 Talence Cedex, FRANCE

EDF Recherche et Développement
Département SINETICS
Groupe I23
1, avenue du Général De Gaulle
92140 Clamart, FRANCE

Remerciements

Après un peu plus de trois ans de travail vient ce moment tant attendu (et redouté) de la rédaction des remerciements. En effet ce travail n'aurait pas été possible sans l'aide, les conseils et les encouragements de nombreuses personnes.

Je tiens à remercier les membres du jury et en premier lieu Olivier Coulaud pour m'avoir fait l'honneur d'accepter de présider le jury. Ma gratitude va également aux rapporteurs Frédéric Feyel et Robert Roy dont les remarques m'ont permis d'améliorer ce manuscrit. Je remercie également Eric Lorentz pour l'intérêt qu'il a porté à mon travail. Les remerciements des membres du jury seraient très incomplets sans ceux de mes trois encadrants. Je commence par Jean Roman qui m'a proposé ce sujet de thèse en collaboration avec EDF et qui au cours de cette thèse a toujours trouvé le temps (non sans difficulté) pour nos réunions. Je continue par Pierre Ramet avec qui j'ai eu plaisir à travailler depuis un projet de deuxième année d'école qu'il co-encadrait. Il a toujours été réactif pour me faire des retours sur mon travail et magnanime en m'évitant le fanny au squash. Enfin je tiens particulièrement à remercier Maxime Barrault pour m'avoir fait confiance, pour avoir toujours été très disponible, pour avoir toujours pris le temps de m'expliquer de nombreux aspects de disciplines nouvelles pour moi, pour m'avoir toujours encouragé (aux moments où les résultats sont décevants, ça compte beaucoup), pour le détail de ses corrections, pour les moments hors travail ...

J'ai passé la majorité de mon temps au sein du département Sinetics et du groupe I23. Je suis reconnaissant de l'accueil chaleureux ainsi que de la richesse scientifique dont j'ai bénéficié. Ce travail doit beaucoup aux discussions que j'ai pu avoir avec Angélique, Laurent, Pierre et Wilfried sur différents aspects du solveur SPn, ainsi qu'aux échanges avec Frank et Olivier sur les aspects solveurs et décomposition de domaine. Je remercie également la Coffee Team I23 et assimilé (Solène, Véronique, François, Wilfried, Marc, Thierry, David, Guy, Sana, Pierre, Bruno, Christophe, Guillaume...) pour ses pauses café lors desquelles on a refait le monde, des appartements ou des lampes... tout en prenant quelques calories. Un grand merci également à François, qui détient le record de longévité dans mon bureau, pour m'avoir supporté lorsque mes jobs ne voulaient décidément pas passer sur le cluster (le reste du temps aussi), pour sa bonne humeur, ses conseils et autres astuces... Je remercie aussi Hervé, dont j'ai été un des encadrants pour son stage de fin d'étude, et avec qui j'ai eu plaisir à travailler. Je n'oublierais pas non plus l'aide que j'ai pu recevoir des autres groupes du département. Je pense en particulier à David, Matthieu, Fabrice et Tanguy pour les aspects neutroniques ainsi qu'à Hugues et Antoine pour les aspects plus informatiques.

Malgré ma présence épisodique à Bordeaux j'ai toujours été très bien accueilli dans les différents locaux de l'équipe (feu-)Scallaplix. Un grand merci donc à toute l'équipe. Outre les discussions techniques, je retiendrai les bon moments passés ensemble (ping-pong, squash, soirée, le café bleu ...). Un grand merci à Mathieu pour m'avoir fourni ma dose quotidienne de café, et pour avoir accéléré mon intégration dans l'équipe.

Merci à Marie-Agnès et Josy pour avoir résolu de nombreux problèmes administratifs. Sans elles je serais devenu fou...

Merci à l'équipe EDF des chargés de TD à la Sorbonne (Guillaume, Marie, Matthieu, Pierre et Solène) pour les échanges de coups de main et les nombreuses discussions.

Merci aux doctorants avec qui j'ai beaucoup discuté de la *dure vie* de thésard. Je me risque à une liste (*a fortiori* non-exhaustive) : Mathieu, Adeline, Jérémie, Wilfried, David, Ludovic, Angélique, Guy, Alexandra, Charlotte, Fabrice, Nicolas, Cédric, Sébastien, Robin, Adam, Algiane,

Guillaume et Cyril.

Merci à tous ceux avec qui je suis parti en conférence ou en école d'été, pour les bons moments passés ensemble, avec une dédicace spéciale à l'équipe du Picotin (François, Jérémie, Jun Ho, Mathieu, Pascal, Pierre et Xavier).

Merci à tous les bordelais chez qui j'ai squatté (Sébastien, Jérémie et Mathieu).

Un grand merci à Adeline et Mathieu car vous êtes hors catégorie pour le point précédent.

Merci au groupe i2pfa1 et à ses diverses extensions (Ad, Antoine, Jérémie, Kévin, Laure, Lolo, Mathieu, Mathieu, Morgane et Tom) avec qui j'ai commencé le parallélisme puis séquentialisé les vacances, soirées et week-end.

Merci à tous ceux qui sont venus à ma soutenance, et tout particulièrement à ceux qui sont venus de loin (Maman, Papa, Cyril, Martine, Laurent et Mathieu).

Merci à tous ceux qui m'ont aidé à organiser le pot de thèse (Maman, Papa, Martine, Laurent, Mathieu, François et Wilfried). Merci à ceux qui m'ont aidé à attaquer les restes (et merci d'avance à ceux qui m'aideront à finir).

Merci à ma famille qui m'a donné l'envie d'étudier et m'a permis de le faire avec un soutien sans faille.

Enfin étant embauché à EDF à l'issue de ma thèse, je tiens à remercier tous ceux qui ont permis que cette aventure se poursuive.

Merci à ceux que j'aurais pu oublier.

Table des matières

Liste des tableaux

Liste des Algorithmes

Table des figures

Introduction

Dans le cadre de l'exploitation de ses centrales nucléaires de type Réacteur à Eau Pressurisée (REP), le groupe EDF développe et met en œuvre des logiciels de calcul permettant de reproduire et de prévoir les phénomènes neutroniques régissant le cœur des centrales. Ces phénomènes sont modélisés par une équation de transport établissant un bilan de neutrons, appelée équation de Boltzmann.

Pour les applications industrielles visées, l'équation du transport neutronique est trop complexe à résoudre directement. Dans la future chaîne industrielle *Cocagne*, il a donc été choisi d'utiliser un schéma à deux étapes. La première étape consiste à calculer des sections homogènes par crayon ou par assemblage. Lors de la seconde étape, l'ensemble du cœur 3D est simulé en utilisant un modèle simplifié à partir de ces sections homogénéisées. C'est cette deuxième étape appelée *solveur de cœur* à laquelle on s'intéresse au cours de cette thèse. Dans la chaîne de calcul *Cocagne*, le modèle utilisé pour cette étape est basé sur les équations de transport simplifié (dites SPn). Afin d'augmenter la précision des résultats, on souhaite augmenter la finesse de discrétisation du solveur de cœur. La taille des problèmes traités devenant importante, la consommation mémoire et les temps de calcul ne permettent plus d'utiliser les ordinateurs classiques.

Jusqu'à récemment, il était possible de se contenter de l'augmentation de la puissance des processeurs pour améliorer les performances des codes de simulation. Maintenant, avec les nouvelles architectures matérielles, le développeur de codes scientifiques doit utiliser des algorithmes parallèles. Les super-calculeurs offrent de très larges possibilités, mais pour pouvoir les utiliser on doit mettre au point des algorithmes parallèles adaptés à la mémoire distribuée. Dans l'optique de la réalisation de calculs de très grande taille en des temps les plus courts possibles, la parallélisation du solveur SPn est une voie naturelle. Au cours de cette thèse, on étudie donc une méthode de décomposition domaine, par nature facilement parallélisable, pour les équations SPn.

Dans ce manuscrit, le chapitre 1 est consacré aux équations mises en jeu dans le solveur de cœur et à la méthode de résolution utilisée. Dans le chapitre 2, on décrit les précédentes tentatives de parallélisation du solveur SPn (EDF) et du solveur Minos (CEA) après une présentation succincte des différentes méthodes de décomposition de domaine nécessaires à la compréhension de ces tentatives. Au cours du chapitre 3, on présente les fondements de la méthode de décomposition de domaine choisie. Cette méthode conduit à plusieurs approches, dont deux sont étudiées en détail dans les chapitres 4 et 5. Pour chacune de ces approches, on présente des résultats numériques sur un cas test académique et sur un cas industriel. Dans le chapitre 5, on aborde également les problématiques liées à l'intégration dans la plate-forme industrielle.

Cette thèse s'est déroulée dans le cadre d'un contrat CIFRE chez EDF R&D et en collaboration avec les équipes Bacchus et HiePACS communes à l'INRIA Bordeaux Sud Ouest, au PRES de Bordeaux et au CNRS (au titre du LaBRI UMR 5800 et de l'IMB UMR 5251).

Chapitre 1

Présentation des équations de la neutronique

Sommaire

Dans ce chapitre, on a pour objectif d'introduire le solveur SPn développé à EDF [1, 2], solveur reprenant la méthodologie du solveur Minos développé au CEA [3, 4]. On commence par un bref rappel du fonctionnement d'une centrale nucléaire. Puis les équations de Boltzmann, qui régissent la population neutronique au sein du réacteur, sont décrites, ainsi que le problème aux valeurs propres qui en dérive dans le cas stationnaire. Ensuite les approximations en énergie et en angle du problème aux valeurs propres sont détaillées. Pour l'approximation en angle, on ne présente que l'approximation de la diffusion. On renvoie à l'annexe B, pour la description des équations de transport simplifié dites SPn. L'approximation en espace et le solveur du problème algébrique associé sont ensuite présentés. Enfin on mettra en évidence quelques propriétés du solveur SPn à l'aide des deux benchmarks que l'on a utilisés pour valider les méthodes de décomposition de domaine proposées.

Cette présentation se base par ailleurs sur [5, 6] ainsi que sur des présentations similaires dans des thèses portant sur la méthode du solveur Minos [7, 8, 9, 10] :

- dans [7], l'auteur détaille les différentes approximations angulaires (PN, SPn, Diffusion) ;
- dans [8], le lecteur trouvera de nombreux détails sur la discrétisation spatiale et les éléments finis de Raviart-Thomas ;
- dans [10], une attention particulière est donnée au traitement de la variable temporelle.

1.1 Concepts de base de la physique des réacteurs

Le fonctionnement d'une centrale nucléaire est basé sur une réaction en chaîne exothermique dans la cuve du réacteur. La chaleur produite est transférée via le circuit primaire à un générateur de vapeur. La vapeur d'eau, circulant dans le circuit secondaire entraîne, via une turbine, l'alternateur qui produit l'électricité. La vapeur d'eau doit ensuite être refroidie avant de retourner dans le générateur de vapeur. Cette opération est effectuée grâce au circuit tertiaire qui est soit relié à une rivière, soit à la mer, soit à une tour aéroréfrigérante. La figure Fig. 1.1 représente les trois circuits du REP (Réacteur à Eau Pressurisée) qui constitue l'essentiel du parc français.

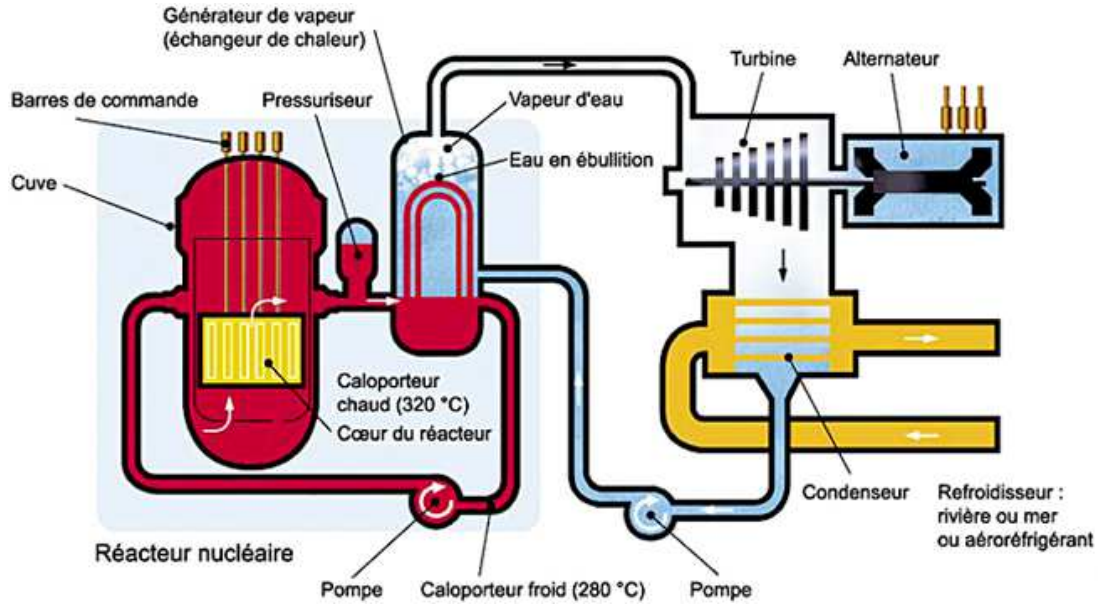


Fig. 1.1 – Fonctionnement d'une centrale nucléaire de type REP (illustration CEA)

La réaction en chaîne est la réaction de fission des atomes lourds (essentiellement l'Uranium 235). Cette réaction est engendrée par un neutron entrant en collision avec un atome lourd. Celle-ci peut générer un ou plusieurs neutrons (en moyenne 2 ou 3). Les neutrons issus d'une réaction de fission sont extrêmement rapides (de l'ordre de $20\,000\,km.s^{-1}$). Or la probabilité qu'une collision engendre une fission est plus grande avec des neutrons de plus faible énergie. Pour provoquer suffisamment de fissions, il y a deux possibilités : soit on enrichit fortement le combustible pour augmenter la concentration d'Uranium 235, soit on ralentit les neutrons. Dans le cas des REPs c'est la deuxième voie qui est choisie pour permettre l'établissement d'une réaction en chaîne (cf. figure FIG. 1.2). Pour ralentir les neutrons, le cœur doit contenir un modérateur. Dans le cas du REP, l'eau qui sert à évacuer la chaleur (fluide caloporteur) sert également de modérateur.

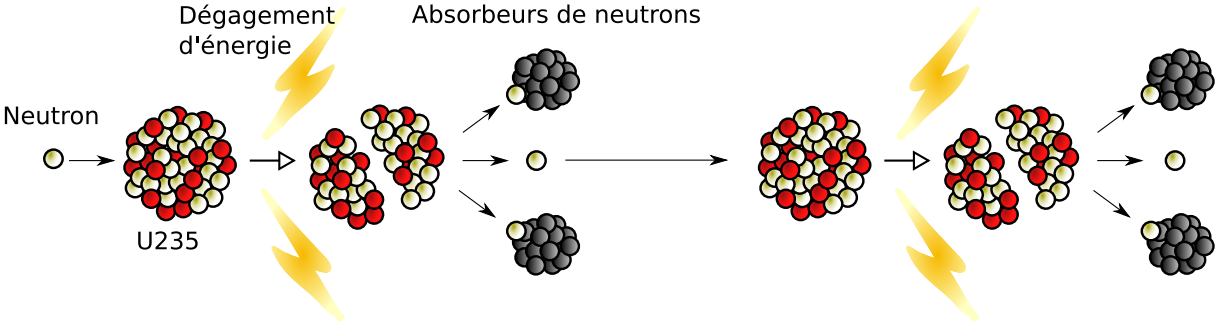


FIG. 1.2 – Réaction en chaîne de fissions de noyaux lourds (illustration inspirée du site web du CEA)

Le cœur d'un REP 900MW (cf. figure FIG. 1.3) est constitué de 157 assemblages à section carrée. La largeur d'un assemblage est de $21,4\,cm$ et sa hauteur de $4\,m$. Chacun de ces assemblages est constitué d'un réseau 17×17. Parmi ces 289 emplacements, on trouve :

- 264 crayons de combustible qui sont constitués d'un tube métallique contenant des pastilles d'oxyde d'uranium faiblement enrichi (entre 3 et 5%). Dans le cas d'assemblages MOX, du plutonium est ajouté ;
- un emplacement central utilisé pour l'instrumentation du cœur ;
- 24 emplacements utilisés pour les barres de contrôle. Ces barres constituées de puissants poisons à neutrons, permettent de contrôler la réaction. Pour ce contrôle, l'opérateur peut également réguler la concentration en Bore de l'eau, le Bore constituant un poison neutronique.

Pour simuler la population neutronique, au sein du cœur, il existe deux classes de méthodes permettant de résoudre l'équation de transport :

1. les méthodes probabilistes (dites de *Monte-Carlo*). On construit à l'aide de variables aléatoires, pour un grand nombre de neutrons, un historique. Ensuite on compte les événements d'intérêt (comme le nombre de fissions, nombre de neutrons, ...) pour une zone donnée du cœur. Cette méthode est très précise mais très coûteuse. Elle est essentiellement utilisée pour obtenir des résultats de référence.
2. les méthodes dites *déterministes* qui sont issues d'approximations et de discrétisations de l'équation du transport dite équation de Boltzmann.

Les équations SPn et de la diffusion étant des méthodes déterministes, nous allons décrire l'obtention de l'équation de Boltzmann et les approximations permettant d'aboutir à ces deux méthodes.

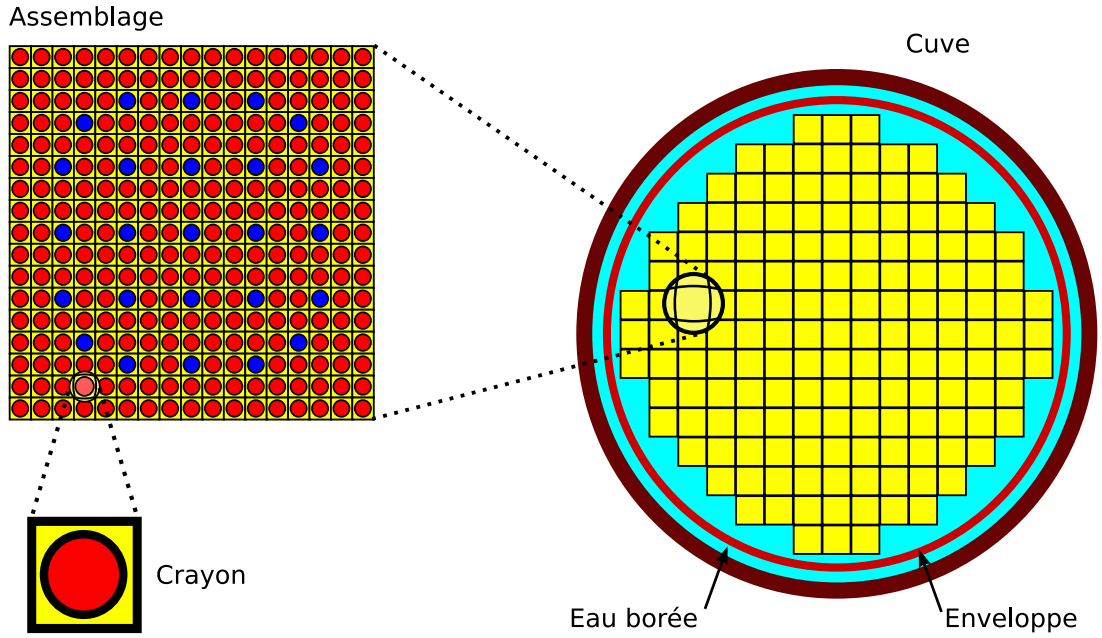


FIG. 1.3 – Géométrie du REP 900MW (inspirée d'une illustration CEA)

1.2 Équation de Boltzmann

L'équation de Boltzmann exprime le bilan neutronique dans un volume élémentaire. La variation du nombre de neutrons provient des migrations, des chocs, et des réactions de fission. On présente l'obtention de cette équation, puis le problème aux valeurs propres en découlant en régime stationnaire. Enfin on présente l'algorithme, dit *de la puissance*, généralement utilisé pour résoudre ce problème.

1.2.1 Présentation de l'équation

Au sein d'un réacteur, l'état d'un neutron se caractérise par les variables suivantes :
- $\overrightarrow{r}$ le vecteur position localisant le neutron ;
- $\overrightarrow{\Omega}$ le vecteur indiquant la direction du neutron ;
- E donnant l'énergie cinétique du neutron ;
- t donnant l'instant d'observation.

Pour caractériser la population neutronique, on peut utiliser deux grandeurs :
- la densité neutronique notée $\mathcal{N}(\overrightarrow{r}, \overrightarrow{\Omega}, E, t)$. Le nombre de neutrons présents dans le volume élémentaire $(d\overrightarrow{r}, d\overrightarrow{\Omega}, dE, dt)$ autour du point $(\overrightarrow{r}, \overrightarrow{\Omega}, E, t)$ s'écrit alors :

$$\mathcal{N}(\overrightarrow{r}, \overrightarrow{\Omega}, E, t) d\overrightarrow{r} d\overrightarrow{\Omega} dE dt;$$

- le flux angulaire défini par $\psi(\overrightarrow{r}, \overrightarrow{\Omega}, E, t) = v\mathcal{N}(\overrightarrow{r}, \overrightarrow{\Omega}, E, t)$. v correspond à la vitesse du neutron et est directement liée à son énergie cinétique E par $E = \frac{1}{2}mv^2$.

L'équation de Boltzmann résulte du bilan neutronique dans un volume élémentaire. Plusieurs phénomènes physiques peuvent provoquer une variation du nombre de neutrons pendant l'intervalle de temps dt :

les disparitions par migration (ou terme de fuite) :

$$div\big(\overrightarrow{\Omega}\psi(\overrightarrow{r},\overrightarrow{\Omega},E,t)\big)d\overrightarrow{r}\,d\overrightarrow{\Omega}\,dEdt\,; \tag{1.1}$$

les disparitions par choc (ou absorption) :

$$\Sigma_t(\overrightarrow{r},\overrightarrow{\Omega},E,t)\psi(\overrightarrow{r},\overrightarrow{\Omega},E,t)d\overrightarrow{r}\,d\overrightarrow{\Omega}\,dEdt\,, \tag{1.2}$$

où Σ_t, appelée *section efficace totale*, représente la probabilité de disparition d'un neutron à la suite d'un choc ;

les arrivées par transfert suite à un choc :

$$\left(\int_0^{+\infty}\!\!\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'\!\to\!\overrightarrow{\Omega},E'\!\to\!E,t\big)\psi(\overrightarrow{r},\overrightarrow{\Omega},E',t)\;d\overrightarrow{\Omega}\;dE'\right)d\overrightarrow{r}\,d\overrightarrow{\Omega}\,dEdt\,, \tag{1.3}$$

où $\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'\!\to\!\overrightarrow{\Omega},E'\!\to\!E,t\big)$, appelée *section efficace de diffusion (scattering)*, est la probabilité pour un neutron d'état $(\overrightarrow{r},\overrightarrow{\Omega}',E',t)$ de passer dans un état $(\overrightarrow{r},\overrightarrow{\Omega},E,t)$;

les sources extérieures permettant l'initialisation de la réaction en chaîne :

$$S_{ext}(\overrightarrow{r},\overrightarrow{\Omega},E,t)d\overrightarrow{r}\,d\overrightarrow{\Omega}\,dEdt\,; \tag{1.4}$$

les sources de fission issues de la fission des atomes fissiles présents dans le combustible :

$$\frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}.\left(\int_0^{+\infty}\!\!\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},\overrightarrow{\Omega}',E',t)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E',t)\;d\overrightarrow{\Omega}'\;dE'\right)d\overrightarrow{r}\,d\overrightarrow{\Omega}\,dEdt\,, \tag{1.5}$$

où :
- $\Sigma_f(\overrightarrow{r},\overrightarrow{\Omega},E,t)$, appelée *section efficace de fission*, représente la probabilité qu'un neutron provoque une fission ;
- $\nu(\overrightarrow{r},E)$ représente le nombre moyen de neutrons émis par une fission ;
- $\dfrac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}$ représente le spectre normalisé de fission, c'est-à-dire la part des neutrons émis par fission dans le niveau d'énergie E.

En faisant l'hypothèse que les milieux présents dans le réacteur sont isotropes, les sections Σ_f et Σ_t ne dépendent plus de la variable angulaire $\overrightarrow{\Omega}$ et les sections de scattering Σ_s ne dépendent plus que de l'angle entre $\overrightarrow{\Omega}$ et $\overrightarrow{\Omega}'$. Le bilan de toutes les contributions (1.1), (1.2), (1.3), (1.4), et (1.5) conduit à l'équation bilan suivante :

$$\begin{aligned}\frac{1}{v}\frac{\partial\psi}{\partial t}(\overrightarrow{r},\overrightarrow{\Omega},E,t) \;=\; &-div\big(\overrightarrow{\Omega}\psi(\overrightarrow{r},\overrightarrow{\Omega},E,t)\big)\\ &-\Sigma_t(\overrightarrow{r},E,t)\psi(\overrightarrow{r},\overrightarrow{\Omega},E,t)\\ &+S_{ext}(\overrightarrow{r},\overrightarrow{\Omega},E,t)\\ &+\left(\int_0^{+\infty}\!\!\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E'\!\to\!E,t\big)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E',t)\;d\overrightarrow{\Omega}'\;dE'\right)\\ &+\frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}.\left(\int_0^{+\infty}\!\!\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},E',t)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E',t)\;d\overrightarrow{\Omega}'dE'\right).\end{aligned} \tag{1.6}$$

1.2.2 Problème aux valeurs propres

En se plaçant dans le cadre d'un régime stationnaire sans source extérieure, l'équation (1.6) devient :

$$div\big(\overrightarrow{\Omega}\psi(\overrightarrow{r},\overrightarrow{\Omega},E)\big)+\Sigma_t(\overrightarrow{r},E)\psi(\overrightarrow{r},\overrightarrow{\Omega},E)=\int_0^{+\infty}\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E'\to E\big)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE' + \frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}.\int_0^{+\infty}\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},E')\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE'.$$

L'équation en régime stationnaire n'admet de solution non nulle que si le cœur est dans une configuration, dite critique, correspondant à une réaction stable et auto-entretenue. Pour étudier l'existence d'un état stationnaire, on introduit le problème aux valeurs propres généralisées suivant en ajoutant un facteur de normalisation λ :

Problème 1
Trouver λ le plus grand réel positif tel qu'il existe une fonction ψ strictement positive solution de :

$$div\big(\overrightarrow{\Omega}\psi(\overrightarrow{r},\overrightarrow{\Omega},E)\big)+\Sigma_t(\overrightarrow{r},E)\psi(\overrightarrow{r},\overrightarrow{\Omega},E)=\int_0^{+\infty}\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E'\to E\big)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE' + \frac{1}{\lambda}\frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}.\int_0^{+\infty}\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},E')\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'dE'. \tag{1.7}$$

Cette valeur propre λ est appelée *facteur effectif de multiplication* et est notée k_{eff}. Physiquement k_{eff} caractérise l'état du cœur :

- si $k_{eff}=1$, l'état est stationnaire : la réaction en chaîne est stable et auto-entretenue. Le réacteur est dit critique ;
- si $k_{eff}>1$, la production de neutrons dépasse l'absorption : la réaction en chaîne s'emballe. Le réacteur est dit sur-critique ;
- si $k_{eff}<1$, l'absorption de neutrons dépasse la production : la réaction en chaîne s'éteint. Le réacteur est dit sous-critique.

1.2.3 Algorithme de résolution

Pour résoudre le problème aux valeurs propres généralisés (1.7), on utilise l'algorithme itératif de la puissance inverse (**Algorithme 1**), qui dans le contexte neutronique est appelé algorithme de la puissance. Celui-ci requiert de calculer :

- des images par l'opérateur de fission multigroupe, noté $\mathcal{F}$, défini par :

$$\mathcal{F}(\psi)= \quad (\overrightarrow{r},\overrightarrow{\Omega},E)\longrightarrow\frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}.\left(\int_0^{+\infty}\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},E')\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'dE'\right);$$

- des antécédents de l'opérateur de transport multigroupe, noté $\mathcal{H}$, défini par :

$$\mathcal{H}(\psi)= \quad (\overrightarrow{r},\overrightarrow{\Omega},E)\longrightarrow div\big(\overrightarrow{\Omega}\psi(\overrightarrow{r},\overrightarrow{\Omega},E)\big)+\ \Sigma_t(\overrightarrow{r},E)\psi(\overrightarrow{r},\overrightarrow{\Omega},E) - \int_0^{+\infty}\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E'\to E\big)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE'.$$

Le calcul de la valeur propre utilise la connaissance du fait que la valeur propre est positive. Dans le cas général, λ serait calculée par : $\lambda = \frac{< \psi, \psi_{old} >}{< \psi_{old}, \psi_{old} >}$. Cette version permet d'économiser un produit scalaire (coûteux car impliquant deux vecteurs différents). Dans la pratique, la convergence de cet algorithme est améliorée par l'ajout d'une technique d'accélération de Tchebychev [11, 12].

Algorithme 1 : Puissance inverse (version continue)

$\psi = 1$;
tant que *!Convergence* **faire**
  $\psi_{old} = \psi$;
  ▷ *Calcul de l'image par l'opérateur de fission*
  $S = \mathcal{F}\,\psi$;
  ▷ *Calcul de l'antécédent par l'opérateur de transport*
  $\psi = \mathcal{H}^{-1}S$;
  ▷ *Calcul de la valeur propre*
  $\lambda = \frac{\|\psi\|}{\|\psi_{old}\|}$;
  ▷ *Normalisation*
  $\psi = \frac{1}{\lambda}.\psi$;
fin

Afin de procéder à la mise en œuvre de l'algorithme de la puissance, on discrétise les opérateurs $\mathcal{H}$ et $\mathcal{F}$ suivant les trois variables suivantes :

- la variable d'énergie E ;
- la variable angulaire $\overrightarrow{\Omega}$;
- la variable d'espace $\overrightarrow{r}$.

1.3 Discrétisation en énergie

1.3.1 Présentation de la discrétisation

On considère que l'espace énergétique est borné. On partitionne cet espace $[E_{inf}, E_{sup}]$ en N_g sous-intervalles :

$$[E_{inf}, E_{sup}] = [E_{N_g}, E_{N_g-1}] \cup [E_{N_g-1}, E_{N_g-2}] \cup \cdots \cup [E_1, E_0].$$

En intégrant l'équation (1.7) sur chaque intervalle $[E_g, E_{g-1}]$ et en posant

$$\psi^g(\overrightarrow{r}, \overrightarrow{\Omega}) = \int_{E_g}^{E_{g-1}} \psi(\overrightarrow{r}, \overrightarrow{\Omega}, E)dE,$$

on obtient N_g équations couplées par Σ_s de la forme :

$$\begin{aligned} &div\big(\overrightarrow{\Omega}\psi^g(\overrightarrow{r},\overrightarrow{\Omega})\big)+\int_{E_g}^{E_{g-1}}\Sigma_t(\overrightarrow{r},E)\psi(\overrightarrow{r},\overrightarrow{\Omega},E)dE= \\ &\int_{E_g}^{E_{g-1}}\sum_{g'}\left(\int_{E_{g'}}^{E_{g'-1}}\int_0^{4\pi}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E'\to E\big)\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE'\right)dE \\ &+\frac{1}{\lambda}\left(\int_{E_g}^{E_{g-1}}\frac{\mathcal{X}(\overrightarrow{r},E)}{4\pi}dE\right).\left(\sum_{g'}\int_{E_{g'}}^{E_{g'-1}}\int_0^{4\pi}\nu(\overrightarrow{r},E')\Sigma_f(\overrightarrow{r},E')\psi(\overrightarrow{r},\overrightarrow{\Omega}',E')\,d\overrightarrow{\Omega}'\,dE'\right). \end{aligned} \tag{1.8}$$

En supposant le flux séparable, on peut écrire sur chaque intervalle $[E_g, E_{g-1}]$ le flux angulaire sous la forme :

$$\psi(\overrightarrow{r},\overrightarrow{\Omega},\overrightarrow{E})=f(E)\psi^g(\overrightarrow{r},\overrightarrow{\Omega})\quad\text{avec}\int_{E_g}^{E_{g-1}}f(E)dE=1.$$

On définit pour chaque groupe g les constantes suivantes :

- la section efficace totale : $\Sigma_t^g(\overrightarrow{r})=\displaystyle\int_{E_g}^{E_{g-1}}\Sigma_t(\overrightarrow{r},E)f(E)dE$;
- la section efficace de fission : $\nu\Sigma_f^g(\overrightarrow{r})=\displaystyle\int_{E_g}^{E_{g-1}}\nu(\overrightarrow{r},E)\Sigma_f(\overrightarrow{r},E)f(E)dE$;
- le spectre de fission : $\mathcal{X}^g(\overrightarrow{r})=\displaystyle\int_{E_g}^{E_{g-1}}\mathcal{X}(\overrightarrow{r},E)dE$.

On définit également pour chaque couple de groupes g et g' la section différentielle de scattering du groupe g vers le groupe g' :

$$\Sigma_s^{g\to g'}(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega})=\int_{E_g}^{E_{g-1}}\int_{E_{g'}}^{E_{g'-1}}\Sigma_s\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega},E\to E'\big)f(E')dE'dE.$$

Avec ces notations, (1.8) devient :

$$\begin{aligned} div\big(\overrightarrow{\Omega}\psi^g(\overrightarrow{r},\overrightarrow{\Omega})\big)+\Sigma_t^g(\overrightarrow{r})\psi^g(\overrightarrow{r},\overrightarrow{\Omega})=\sum_{g'}\left(\int_0^{4\pi}\Sigma_s^{g'\to g}\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega}\big)\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\right) \\ +\frac{1}{\lambda}\left(\frac{\mathcal{X}^g(\overrightarrow{r})}{4\pi}\right).\left(\sum_{g'}\int_0^{4\pi}\nu\Sigma_f^{g'}(\overrightarrow{r})\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\right). \end{aligned} \tag{1.9}$$

1.3.2 Méthode de résolution du problème aux valeurs propres multigroupe

On définit les opérateurs suivants :

- $\mathcal{H}^{g'\to g}$ l'opérateur de scattering[1] qui au flux $\psi^{g'}$ associe le terme :
$$-\int_0^{4\pi}\Sigma_s^{g'\to g}\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega}\big)\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\,;$$
- $\mathcal{H}^{g=g}$ l'opérateur de transport monogroupe qui au flux ψ^g associe le terme :
$$div\big(\overrightarrow{\Omega}\psi^g(\overrightarrow{r},\Omega)\big)+\Sigma_t^g(\overrightarrow{r})\psi^g(\overrightarrow{r},\Omega)\big)-\int_0^{4\pi}\Sigma_s^{g\to g}\big(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega}\big)\psi^g(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\,;$$
- $\mathcal{F}^{g'\to g}$ l'opérateur de fission du groupe g' au groupe g qui au flux $\psi^{g'}$ associe le terme :
$$\frac{\mathcal{X}^g(\overrightarrow{r})}{4\pi}\int_0^{4\pi}\nu\Sigma_f^{g'}(\overrightarrow{r})\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'.$$

[1] Quand $g'>g$, respectivement $g>g'$, on utilise la dénomination *down-scattering* (descente en énergie), respectivement d'*up-scattering* (remontée en énergie).

(1.9) se récrit alors :

$$\mathcal{H}^{g=g}\psi^g + \sum_{g' \neq g} \mathcal{H}^{g' \to g}\psi^{g'} = \frac{1}{\lambda} \sum_{g'} \mathcal{F}^{g' \to g}\psi^{g'}.$$

A chaque itération de puissance, le produit par l'opérateur de fission ($\mathcal{F}\psi$) dans l'**Algorithme 1** s'écrit donc :

$$S^g = \sum_{g' \neq g} \mathcal{F}^{g' \to g}\psi^{g'}.$$

A chaque itération de puissance, le système $\mathcal{H}\psi = S$ dans l'**Algorithme 1** s'écrit donc pour tout $1 \leq g \leq N_g$:

$$\mathcal{H}^{g=g}\psi^g = S^g - \sum_{g' \neq g} \mathcal{H}^{g' \to g}\psi^{g'}. \tag{1.10}$$

La résolution de (1.10) s'effectue à l'aide d'un algorithme de Gauss-Seidel par bloc. A chaque itération, on résout successivement N_g problèmes monogroupes avec g décroissant[2]. En faisant intervenir ces itérations emboîtées, l'**Algorithme 1** devient l'**Algorithme 2**.

Algorithme 2 : Puissance inverse (version multigroupe continue)

$(\psi^1, \ldots, \psi^{N_g}) = 1$;
tant que *!Convergence* **faire**
  $(\psi^1_{old}, \ldots, \psi^{N_g}_{old}) = (\psi^1, \ldots, \psi^{N_g})$;

  ▷ *Calcul de l'image par l'opérateur de fission*
  pour chaque g **faire**
    $S^g = \sum_{g'} \mathcal{F}^{g' \to g}\psi^{g'}$;
  fin

  ▷ *Résolution du problème multigroupe par l'algorithme de Gauss-Seidel par bloc*
  tant que *!Convergence* **faire**
    pour chaque g **faire**
      ▷ *Calcul du terme source monogroupe*
      $L = S^g - \sum_{g' \neq g} \mathcal{H}^{g' \to g}\psi^{g'}$;
      ▷ *Résolution du problème monogroupe*
      $\psi^g = (\mathcal{H}^{g=g})^{-1} L$;
    fin
  fin

  ▷ *Calcul de la valeur propre*
  $\lambda = \dfrac{\|(\psi^1, \ldots, \psi^{N_g})\|}{\|(\psi^1_{old}, \ldots, \psi^{N_g}_{old})\|}$;
  ▷ *Normalisation*
  $(\psi^1, \ldots, \psi^{N_g}) = \dfrac{1}{\lambda}.(\psi^1, \ldots, \psi^{N_g})$;
fin

[2] On choisit g décroissant car l'up-scattering est plus faible numériquement que le down-scattering.

1.4 Approximation angulaire

Après la discrétisation en énergie, il est nécessaire d'effectuer une discrétisation angulaire. Pour ce faire il existe plusieurs techniques. Les méthodes Sn (ou méthodes des ordonnées discrètes [5, chapitre 4]) utilisent une discrétisation de la variable angulaire en $n(n+2)$ directions. Elles nécessitent le calcul de $n(n+2)$ fonctions scalaires indépendantes. Le flux angulaire est ensuite obtenu en additionnant les contributions de chaque direction par une formule de quadrature angulaire. Les méthodes décrites par la suite sont issues d'un développement à un ordre fini du flux et des sections de scattering dépendant de la variable angulaire :

- Pour la méthode Pn [7, p.67], ces fonctions sont développées sur une base d'harmoniques sphériques à l'ordre n. La méthode nécessite de calculer $(n+1)^2$ fonctions scalaires couplées ;
- Pour la méthode SPn [7, p.73], décrite en annexe B, on fait l'approximation d'une solution localement plane, puis on développe à l'ordre n (avec n impair) le flux et les sections de scattering sur une base de polynômes de Legendre. La méthode nécessite de calculer $2(n+1)$ fonctions scalaires. Cette approximation est moins coûteuse que la méthode Pn et plus précise que l'approximation de la diffusion décrite ci-dessous.

Bien que nous intéressant aux équations SPn, pour simplifier le propos, seules les équations de la diffusion sont présentées. En effet, les équations de transport simplifiées SPn conduisent à la résolution de $(n+1)/2$ systèmes de diffusion couplés[3]. La mise en œuvre des méthodes de décomposition de domaine proposées dans les chapitres 4 et 5 est similaire pour ces deux modèles.

Pour chaque groupe g, on définit le flux scalaire ϕ^g et le courant $\overrightarrow{J^g}$ par :

$$\phi^g(\overrightarrow{r}) = \int_0^{4\pi} \psi^g(\overrightarrow{r}, \overrightarrow{\Omega}) d\overrightarrow{\Omega}, \quad \overrightarrow{J^g}(\overrightarrow{r}) = \int_0^{4\pi} \overrightarrow{\Omega}.\psi^g(\overrightarrow{r}, \overrightarrow{\Omega}) d\overrightarrow{\Omega}.$$

On fait l'hypothèse de la loi de Fick qui s'exprime pour chaque groupe g par

$$\overrightarrow{J^g} = -\mathcal{D}^g \overrightarrow{\nabla} \phi^g$$

où $\mathcal{D}^g$ est appelé coefficient de diffusion[4]. Le **Problème 1** devient :

Problème 2
Trouver λ le plus grand réel positif tel qu'il existe N_g fonctions $(\phi^1, \ldots, \phi^{N_g})$ strictement positives solutions des systèmes couplés suivants : pour tout $1 \leq g \leq N_g$

$$\begin{cases} div\left(\overrightarrow{J^g}(\overrightarrow{r})\right) + \Sigma_t^g(\overrightarrow{r})\phi^g(\overrightarrow{r}) = \displaystyle\sum_{g'} \left(\Sigma_s^{g\to g'}(\overrightarrow{r})\phi^{g'}(\overrightarrow{r})\right) + \frac{1}{\lambda}\frac{\mathcal{X}^g}{4\pi}.\sum_{g'}\left(\nu\Sigma_f^{g'}(\overrightarrow{r})\phi^{g'}(\overrightarrow{r})\right) \\ \overrightarrow{J^g}(\overrightarrow{r}) + D^g(\overrightarrow{r})\overrightarrow{\nabla}\phi^g(\overrightarrow{r}) = \overrightarrow{0}. \end{cases}$$

L'utilisation des équations de la diffusion implique donc de calculer quatre fonctions scalaires.

[3] n est impair.

[4] Dans le cas du modèle SP1, ce coefficient est égal à $\dfrac{1}{3\left(\Sigma_t^g - \Sigma_s^{g\to g}\right)}$.

1.5 Approximation en espace

Les choix faits dans le solveur SPn développé à EDF [1, 2] pour l'approximation en espace sont les mêmes que pour le solveur Minos [3, 4]. La résolution du problème aux valeurs propres multigroupe (**Problème 2**) conduit, à chaque itération de puissance, à résoudre un système linéaire multigroupe. Ce système est résolu via un algorithme de Gauss-Seidel par bloc. A chaque itération de cet algorithme, on résout N_g systèmes linéaires monogroupes : pour tout $1 \leq g \leq N_g$

$$\left\{ \begin{array}{rcrcl} div\big(\overrightarrow{J^g}(\overrightarrow{r})\big) & + & \big(\Sigma_t^g(\overrightarrow{r}) - \Sigma_s^{g\to g}(\overrightarrow{r})\big)\phi^g(\overrightarrow{r}) & = & S^g(\overrightarrow{r}) \\ \overrightarrow{J^g}(\overrightarrow{r}) & + & D^g(\overrightarrow{r})\overrightarrow{\nabla}\phi^g(\overrightarrow{r}) & = & \overrightarrow{0} \end{array} \right. \tag{1.11}$$

avec des conditions aux limites de type *"flux nul"* au bord[5]. Les seconds membres S^g désignent la source multigroupe calculée à l'aide de l'opérateur de fission multigroupe. Le terme $\overrightarrow{0}$ peut ne pas être nul[6] lorsque que les sections de scattering sont développées à un ordre strictement positif (cf. annexe B). On formalise donc mathématiquement le problème monogroupe (1.11) par le problème suivant :

Problème 3
Trouver $(\phi, \overrightarrow{J}) \in L^2(\Omega) \times H(div, \Omega)$ *tel que*

$$\left\{ \begin{array}{rcrcll} div\,\overrightarrow{J} & + & \Sigma\phi & = & f & dans\ \Omega \\ \dfrac{\overrightarrow{J}}{D} & + & \overrightarrow{\nabla}\phi & = & \overrightarrow{0} & dans\ \Omega \\ & & \phi & = & 0 & sur\ \partial\Omega \end{array} \right. \tag{1.12}$$

avec Σ, $D \in C^0(\Omega)$ *bornées et positives et* f *dans* $L^2(\Omega)$.

1.5.1 Formulation variationnelle

En multipliant par v la première ligne de (1.12), par $\overrightarrow{w}$ la seconde, en intégrant sur le domaine Ω et enfin en appliquant la formule de Green, on obtient le problème variationnel mixte dual suivant [2, 13] :

Problème 4
Trouver $(\phi, \overrightarrow{J}) \in L^2(\Omega) \times H(div, \Omega)$ *tel que*

$$\left\{ \begin{array}{lll} \displaystyle\int_\Omega div\Big(\overrightarrow{J}(\overrightarrow{r})\Big)v(\overrightarrow{r})\,d\overrightarrow{r} + \int_\Omega \Sigma(\overrightarrow{r})\phi(\overrightarrow{r})v(\overrightarrow{r})\,d\overrightarrow{r} = \int_\Omega f(\overrightarrow{r})v(\overrightarrow{r})\,d\overrightarrow{r} & & \forall v \in L^2(\Omega) \\ \displaystyle\int_\Omega \frac{1}{D(\overrightarrow{r})}\overrightarrow{J}(\overrightarrow{r}).\overrightarrow{w}(\overrightarrow{r})\,d\overrightarrow{r} - \int_\Omega \phi(\overrightarrow{r})div\Big(\overrightarrow{w}(\overrightarrow{r})\Big)\,d\overrightarrow{r} = 0 & & \forall \overrightarrow{w} \in H(div, \Omega). \end{array} \right.$$

Les espaces $L^2(\Omega)$ et $H(div, \Omega)$ sont approchés par des espaces de dimension finie V_h et W_h. On se ramène donc à résoudre le problème :

[5] Il existe d'autres conditions aux limites :
- les conditions de réflexion : elles permettent de gérer le cas des réacteurs comportant des axes de symétrie ;
- les conditions d'Albédo qui sont comparables à des conditions de type Robin pour la modélisation des réflecteurs.

[6] Cette hypothèse simplifie la présentation qui peut aisément se généraliser.

Problème 5
Trouver $(\phi_h, \overrightarrow{J_h}) \in V_h \times W_h$ *tel que*

$$\begin{cases} \displaystyle\int_\Omega div\big(\overrightarrow{J_h}(\overrightarrow{r})\big) v_h(\overrightarrow{r})\, d\overrightarrow{r} \quad + \int_\Omega \Sigma(\overrightarrow{r})\phi_h(\overrightarrow{r}) v_h(\overrightarrow{r})\, d\overrightarrow{r} = \int_\Omega f(\overrightarrow{r}) v_h(\overrightarrow{r})\, d\overrightarrow{r} \quad \forall v_h \in V_h \\ \displaystyle\int_\Omega \frac{1}{D(\overrightarrow{r})} \overrightarrow{J_h}(\overrightarrow{r}) \overrightarrow{w}_h(\overrightarrow{r})\, d\overrightarrow{r} \; - \int_\Omega \phi_h(\overrightarrow{r}) div\big(\overrightarrow{w}_h(\overrightarrow{r})\big)\, d\overrightarrow{r} = 0 \qquad\qquad \forall \overrightarrow{w}_h \in W_h. \end{cases}$$

De par la géométrie parallélépipédique des assemblages, on modélise le cœur par un parallélépipède rectangle, auquel on associe un maillage cartésien défini par N_x, N_y et N_z mailles dans chacune des trois directions de l'espace. On note $N = N_x.N_y.N_z$ le nombre total de mailles. Pour chaque maille indicée par n, on note K_n son volume. On considère que sur chaque maille $D(\overrightarrow{r})$ et $\Sigma(\overrightarrow{r})$ sont des fonctions constantes. On note par D_n et Σ_n leur valeur dans chaque maille. On se ramène alors au problème discrétisé suivant :

Problème 6
Trouver $(\phi_h, \overrightarrow{J_h}) \in V_h \times W_h$ *tel que*

$$\begin{cases} b(v_h, \overrightarrow{J_h}) + t(\phi_h, v_h) &= s(v_h) & \forall v_h \in V_h \\ a(\overrightarrow{J_h}, \overrightarrow{w}_h) - b(\phi_h, \overrightarrow{w}_h) &= 0 & \forall \overrightarrow{w}_h \in W_h \end{cases} \tag{1.13}$$

avec

$$\left| \begin{array}{rcl} a\big(\overrightarrow{J_h}, \overrightarrow{w}_h\big) &=& \displaystyle\sum_n \frac{1}{D_n} \int_{K_n} \overrightarrow{J_h}(\overrightarrow{r}).\overrightarrow{w}_h(\overrightarrow{r}) d\overrightarrow{r} \\ b(\phi_h, \overrightarrow{w}_h) &=& \displaystyle\int_\Omega \phi_h(\overrightarrow{r}) div\big(\overrightarrow{w}_h(\overrightarrow{r})\big) d\overrightarrow{r} \\ t(\phi_h, v_h) &=& \displaystyle\sum_n \Sigma_n \int_{K_n} \phi_h(\overrightarrow{r}) v_h(\overrightarrow{r}) d\overrightarrow{r} \\ s(v_h) &=& \displaystyle\int_\Omega f(\overrightarrow{r}) v_h(\overrightarrow{r}) d\overrightarrow{r}. \end{array} \right.$$

1.5.2 L'élément fini de Raviart-Thomas

On utilise l'élément fini de Raviart-Thomas-Nedelec [14, 15], adapté à la formulation mixte duale, qui permet d'obtenir un système matriciel très creux. Pour pouvoir définir les espaces d'approximation V_h et W_h, on introduit $P_{l,m,n}$ l'ensemble des polynômes à trois variables de degré l, m et n par rapport respectivement aux variables x, y et z. L'élément fini RTk d'ordre k conduit aux espaces V_h^k et W_h^k définis par :

- V_h^k est l'ensemble des fonctions telles que leur restriction dans chaque maille soit dans $P_{k,k,k}$;
- W_h^k est l'ensemble des fonctions $\overrightarrow{w}$ telles que leur restriction dans chaque maille soit dans $P_{k+1,k,k} \times P_{k,k+1,k} \times P_{k,k,k+1}$ et telles que les fonctions $\overrightarrow{w}.\overrightarrow{n}$ soient continues au niveau des interfaces entre les mailles.

▷ Les éléments finis de Raviart-Thomas ne sont pas définis sur les mailles K_n mais sur l'élément de référence $\hat{K} = [-1,1]^3$:

– En 3D les fonctions scalaires de référence $\hat{v}_h$ sont développées sur une base de $(k+1)^3$ fonctions

$$\left\{\hat{v}_{(i_x,i_y,i_z)} \quad | \quad i_d \in [\![0,k]\!] \text{ et } d \in \{x,y,z\}\right\}$$

dont les degrés de liberté associés sont représentés par des points sur la figure Fig. 1.4. $\hat{v}_{(i_x,i_y,i_z)}$ est définie comme le produit des polynômes de Lagrange aux points de Gauss (définis en annexe A.2) :

$$\hat{v}_{(i_x,i_y,i_z)}(x,y,z) = l_{i_x}(x).l_{i_y}(y).l_{i_z}(z).$$

– Pour définir la base des fonctions vectorielles, on définit les polynômes $\{p_i \mid i \in [\![0,k+1]\!]\}$ sur $[-1,1]$ à partir des polynômes de Lagrange associés aux points de Gauss d'ordre k par

$$\begin{cases} p_0(x) = 1 - \dfrac{1}{w_0}\displaystyle\int_{-1}^{x} l_0(t)dt \\ p_i(x) = 1 - \dfrac{1}{w_{i-1}}\displaystyle\int_{-1}^{x} l_{i-1}(t)dt \\ p_{k+1} = \dfrac{1}{w_k}\displaystyle\int_{-1}^{x} l_k(t)dt \end{cases} , \qquad \begin{array}{r}\text{avec } \{w_i\} \text{ les points de Gauss d'ordre k} \\ \text{(cf. annexe A.2).}\end{array}$$

$\overrightarrow{w}_h$ sont définies sur une base de $3(k+1)^2(k+2)$ fonctions

$$\left\{\overrightarrow{\hat{w}^x}_{i_x,i_y,i_z} = \begin{pmatrix} \hat{w}^x_{i_x,i_y,i_z} \\ 0 \\ 0 \end{pmatrix} \quad | \quad i_x \in [\![0,k+1]\!], \quad (i_y,i_z) \in [\![0,k]\!]^2\right\}$$
$$\cup \left\{\overrightarrow{\hat{w}^y}_{i_x,i_y,i_z} = \begin{pmatrix} 0 \\ \hat{w}^y_{i_x,i_y,i_z} \\ 0 \end{pmatrix} \quad | \quad i_y \in [\![0,k+1]\!], \quad (i_x,i_z) \in [\![0,k]\!]^2\right\}$$
$$\cup \left\{\overrightarrow{\hat{w}^z}_{i_x,i_y,i_z} = \begin{pmatrix} 0 \\ 0 \\ \hat{w}^z_{i_x,i_y,i_z} \end{pmatrix} \quad | \quad i_z \in [\![0,k+1]\!], \quad (i_x,i_y) \in [\![0,k]\!]^2\right\}$$

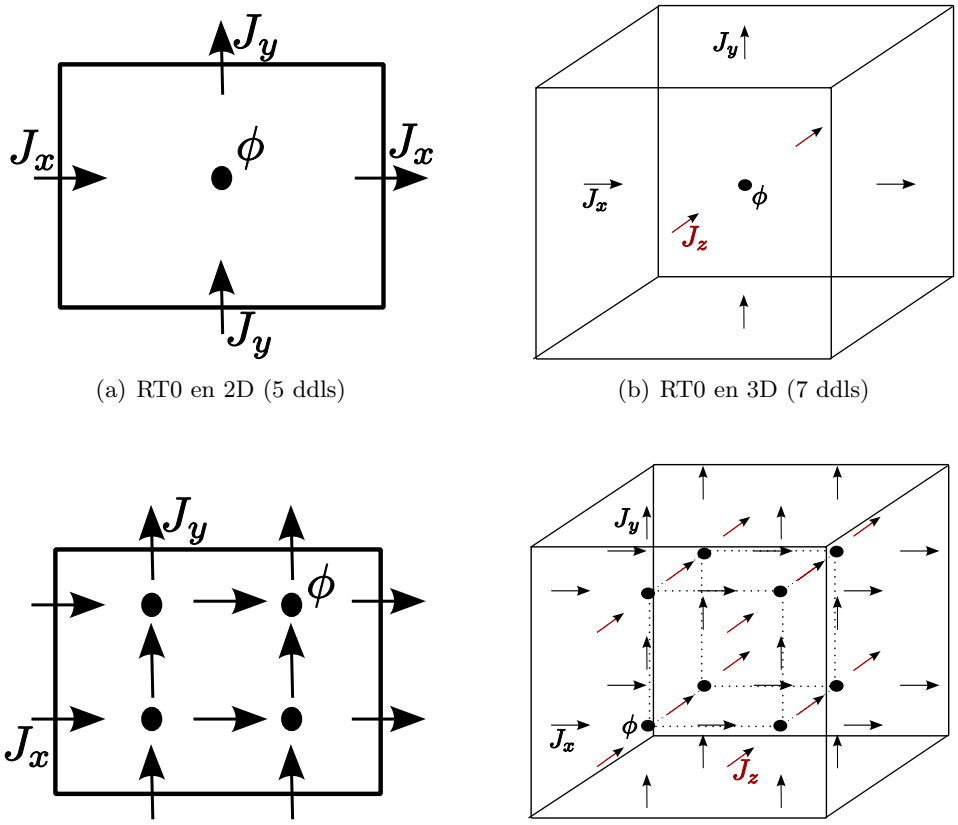
avec $\begin{cases} \hat{w}^x_{i_x,i_y,i_z}(x,y,z) = p_{i_x}(x)l_{i_y}(y)l_{i_z}(z) \\ \hat{w}^y_{i_x,i_y,i_z}(x,y,z) = l_{i_x}(x)p_{i_y}(y)l_{i_z}(z) \\ \hat{w}^z_{i_x,i_y,i_z}(x,y,z) = l_{i_x}(x)l_{i_y}(y)p_{i_z}(z). \end{cases}$

On remarque que $p_i(-1) = \delta_{i,0}$ et $p_i(1) = \delta_{i,k+1}$ ce qui permet d'assurer la continuité aux interfaces en plaçant les degrés de liberté associés aux fonctions $\overrightarrow{\hat{w}^x}_{-1,i_y,i_z}$ sur l'interface gauche de $\hat{K}$ et sur l'interface droite ceux associés à $\overrightarrow{\hat{w}^x}_{1,i_y,i_z}$. Les degrés de liberté associés aux fonctions $\overrightarrow{\hat{w}^d}$ sont représentés par des flèches sur la figure Fig. 1.4.

▷ Pour obtenir les fonctions de base sur une maille K_n définie par$[x_n, x_n + h^x_n] \times [y_n, y_n + h^y_n] \times [z_n, z_n + h^z_n]$, on applique le changement de variable suivant :

$$\begin{array}{rccl} F_n : & [x_n, x_n + h^x_n] \times [y_n, y_n + h^y_n] \times [z_n, z_n + h^z_n] & \to & [-1,\,1]^3 \\ & (x,y,z) & \to & (2.\frac{x-x_n}{h^x_n} - 1,\; 2.\frac{y-y_n}{h^y_n} - 1,\; 2.\frac{z-z_n}{h^z_n} - 1) \end{array} .$$

Une fonction de base v (respectivement w) s'obtient à partir de $\hat{v}$ (respectivement $\hat{w}$) par la relation $v = \hat{v}\, o\, F_n$ (respectivement $w = \hat{w}\, o\, F_n$).



(a) RT0 en 2D (5 ddls)

(b) RT0 en 3D (7 ddls)

(c) RT1 en 2D (16 ddls)

(d) RT1 en 3D (44 ddls)

FIG. 1.4 – Représentation des degrés de liberté des éléments finis de Raviart-Thomas-Nedelec sur une maille

1.5.3 Formulation matricielle

On note :
- A_D la matrice associée à la forme bilinéaire a ;
- B_D la matrice associée à la forme bilinéaire b ;
- T la matrice associée à la forme bilinéaire t ;
- S le vecteur associé à la forme linéaire s ;
- J_D le vecteur contenant les degrés de liberté du courant $\overrightarrow{J_h}$;
- ϕ le vecteur contenant les degrés de liberté du flux ϕ_h.

L'équation (1.13) s'écrit alors sous la forme matricielle

$$\begin{pmatrix} A_D & -B_D \\ {}^tB_D & T \end{pmatrix} \begin{pmatrix} J_D \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ S \end{pmatrix}.$$

J_D est décomposé en trois segments, un pour chaque direction de l'espace. Le vecteur J_D et la matrice B_D s'écrivent donc

$$J_D = \begin{pmatrix} J_x \\ J_y \\ J_z \end{pmatrix}, \quad B_D = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}.$$

Comme pour $d \neq d'$, on a $\displaystyle\int_K \overrightarrow{\hat{w}^d}_{i_x,i_y,i_z}(x,y,z).\overrightarrow{\hat{w}^{d'}}_{j_x,j_y,j_z}(x,y,z)dxdydz = 0$, la matrice A_D ne contient pas de couplage entre les directions de l'espace et s'écrit donc sous la forme d'une matrice bloc diagonale

$$A_D = \begin{pmatrix} A_x & & \\ & A_y & \\ & & A_z \end{pmatrix}.$$

L'équation (1.13) s'écrit alors sous forme matricielle

$$\begin{pmatrix} A_x & & & -B_x \\ & A_y & & -B_y \\ & & A_z & -B_z \\ {}^tB_x & {}^tB_y & {}^tB_z & T \end{pmatrix} \begin{pmatrix} J_x \\ J_y \\ J_z \\ \phi \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ S \end{pmatrix}. \tag{1.14}$$

Lorsque cela est nécessaire, les sous-matrices possèdent un exposant indiquant le niveau d'énergie auquel elles correspondent[7]. Pour le groupe d'énergie g', on note par exemple

$$H^{g=g'} = \begin{pmatrix} A_x^{g=g'} & & & -B_x \\ & A_y^{g=g'} & & -B_y \\ & & A_z^{g=g'} & -B_z \\ {}^tB_x & {}^tB_y & {}^tB_z & T^{g=g'} \end{pmatrix}.$$

On remarque que les sous-matrices B_d (d étant une direction x, y ou z) ne dépendent pas de l'indice de groupe car la forme bilinéaire b ne dépend pas des données physiques. Dans la suite de ce paragraphe, nous déterminons les propriétés des blocs de la matrice $H^{g=g'}$ qui permettent d'obtenir un solveur efficace.

[7] Lorsqu'il n'y a pas d'ambiguïté possible, on omet l'écriture $g = g'$ à l'exposant. La matrice H sera définie comme la matrice multigroupe (cf. page 22) et l'omission de l'exposant $g = g'$ ne peut donc être faite dans ce cas.

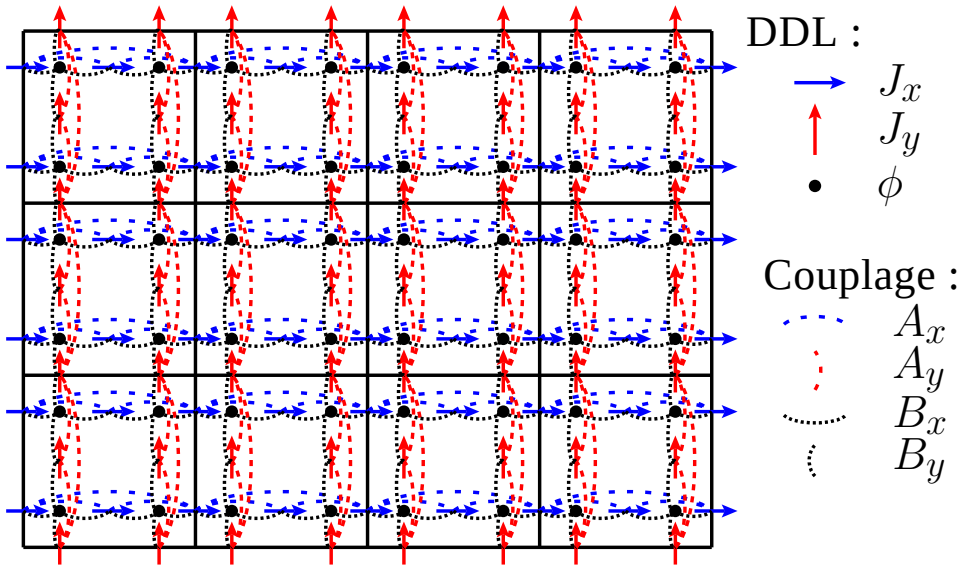


FIG. 1.5 – Représentation des termes de couplage avec les éléments finis RT1

▷ La matrice T est diagonale définie positive car la base de fonctions de V_h est orthogonale.

▷ Comme la matrice A_D dérive directement des produits scalaires des fonctions de base de W_h ; elle est donc symétrique définie positive ainsi que chacun de ses trois blocs diagonaux (A_x, A_y, et A_z). Considérons la matrice A_x. On observe que

$$\begin{aligned} & \int_{\hat{K}} \overrightarrow{\hat{w}^x}_{i_x,i_y,i_z}(x,y,z).\overrightarrow{\hat{w}^x}_{j_x,j_y,j_z}(x,y,z)dx\,dy\,dz \\ = & \int_{-1}^{1} p_{i_x}(x)p_{j_x}(x)dx.\int_{-1}^{1} l_{i_y}(y)l_{j_y}(y)dy.\int_{-1}^{1} l_{i_z}(z)l_{j_z}(z)dz \\ = & \left(\int_{-1}^{1} p_{i_x}(x)p_{j_x}(x)dx\right).w_{i_y}.w_{i_z}.\delta_{i_y,j_y}.\delta_{i_z,j_z}. \end{aligned} \tag{1.15}$$

Le terme $\delta_{i_y,j_y}.\delta_{i_z,j_z}$ permet de définir les lignes de courant, puisque la matrice A_x ne contient des termes de couplage qu'entre degrés de liberté appartenant à la même ligne définie par les indices i_y et i_z (cf. figure FIG. 1.5).

▷ Pour déterminer le profil de la matrice B_d, on calcule

$$\begin{aligned} & \int_{\hat{K}} \hat{v}_{i_x,i_y,i_z}(x,y,z)div\big(\overrightarrow{\hat{w}^x}_{j_x,j_y,j_z}(x,y,z)\big)dx\,dy\,dz \\ = & \int_{-1}^{1} l_{i_y}(y)l_{j_y}(y)dy.\int_{-1}^{1} l_{i_z}(z)l_{j_z}(z)dz.\int_{-1}^{1} \frac{\partial p_{i_x}(x)}{\partial x} l_{i_x}(x)dx \\ = & \delta_{i_y,j_y}.\delta_{i_z,j_z}.w_{i_y}.w_{i_z}.\int_{-1}^{1} \frac{\partial p_{i_x}(x)}{\partial x} l_{j_x}(x)dx \\ = & \begin{cases} -\delta_{i_y,j_y}.\delta_{i_z,j_z}.\delta_{0,i_x}.w_{i_y}.w_{i_z} & \text{pour } j_x = 0 \\ \delta_{i_y,j_y}.\delta_{i_z,j_z}(\delta_{j_x,i_{x-1}} - \delta_{j_x,i_{x+1}}).w_{i_y}.w_{i_z} & \text{pour } j_x \in [\![1,k]\!] \\ \delta_{i_y,j_y}.\delta_{i_z,j_z}\delta_{k,i_x}.w_{i_y}.w_{i_z} & \text{pour } j_x = k+1. \end{cases} \end{aligned} \tag{1.16}$$

Cette expression signifie que chaque degré de liberté de flux n'est couplé qu'avec les degrés de liberté de courant situés juste avant et juste après dans chaque direction (cf. figure FIG. 1.5).

Pour obtenir les matrices A_d et B_d, il reste à numéroter l'ensemble des degrés de liberté. Les degrés de liberté de courant dans la direction d sont numérotés suivant la direction d, puis la

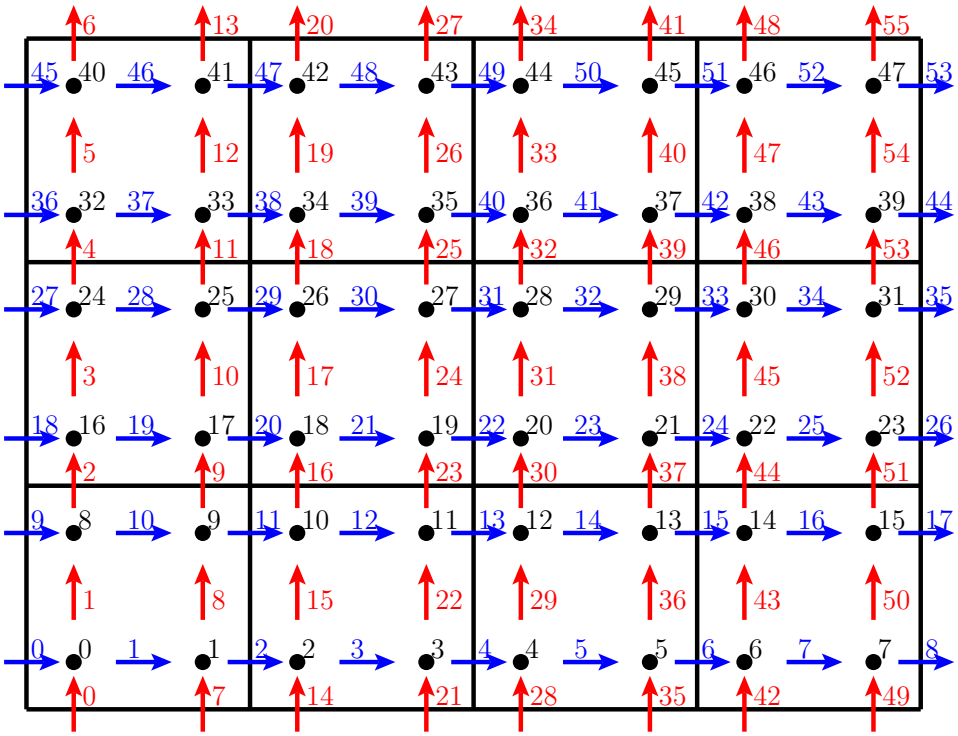


FIG. 1.6 – Numérotation des degrés de liberté avec les éléments finis RT1

direction $d+1$ et enfin $d+2$. Pour le flux, on ordonne les degrés de liberté suivant la direction x puis y et enfin z. Sur la figure FIG. 1.6, on représente pour un cas 2D avec l'élément fini RT1, la numérotation des flux (en noir), des courants J_x (en bleu) et des courants J_y (en rouge). Observons l'impact de cette numérotation sur la structure des matrices A_d et B_d.

▷ Ce choix permet de respecter la structure des lignes de courant (cf. (1.15)). Les matrices A_d sont donc blocs diagonales. Chaque bloc corrrespondant à une ligne de courant a une structure bande dont la largeur dépend de l'ordre de l'élément fini. Ces blocs sont tridiagonaux en RT0 et pentadiagonaux en RT1.

▷ Comme la numérotation des flux suit la même logique que la numérotation des courants J_x, la matrice B_x est une matrice rectangulaire avec une bande de coefficients valant 1 et une bande de coefficients valant -1[8]. La matrice n'est donc pas stockée en mémoire. Les courants J_y ou J_z ne suivent pas la même numérotation que le flux. On perd donc la continuité des accès mémoire lors d'un produit matriciel par B_y ou B_z.

Sur la figure FIG. 1.7, on représente la matrice $H^{g=g'}$ obtenue avec ce choix de numérotation (représentée sur la figure FIG. 1.6).

1.5.4 Algorithme de résolution

La matrice $H^{g=g'}$ n'est pas symétrique. Afin d'obtenir un système symétrique, il suffit de multiplier par -1 les trois premières lignes du système (1.14). Mais comme $-A_D$ est définie négative et T définie positive, la matrice obtenue n'est ni positive ni négative. Afin de se ramener

[8] En effet dans la pratique, on effectue un changement de variable pour supprimer les poids de Gauss, et les pas de maillage dans l'expression des matrices B_d (cf. [16])

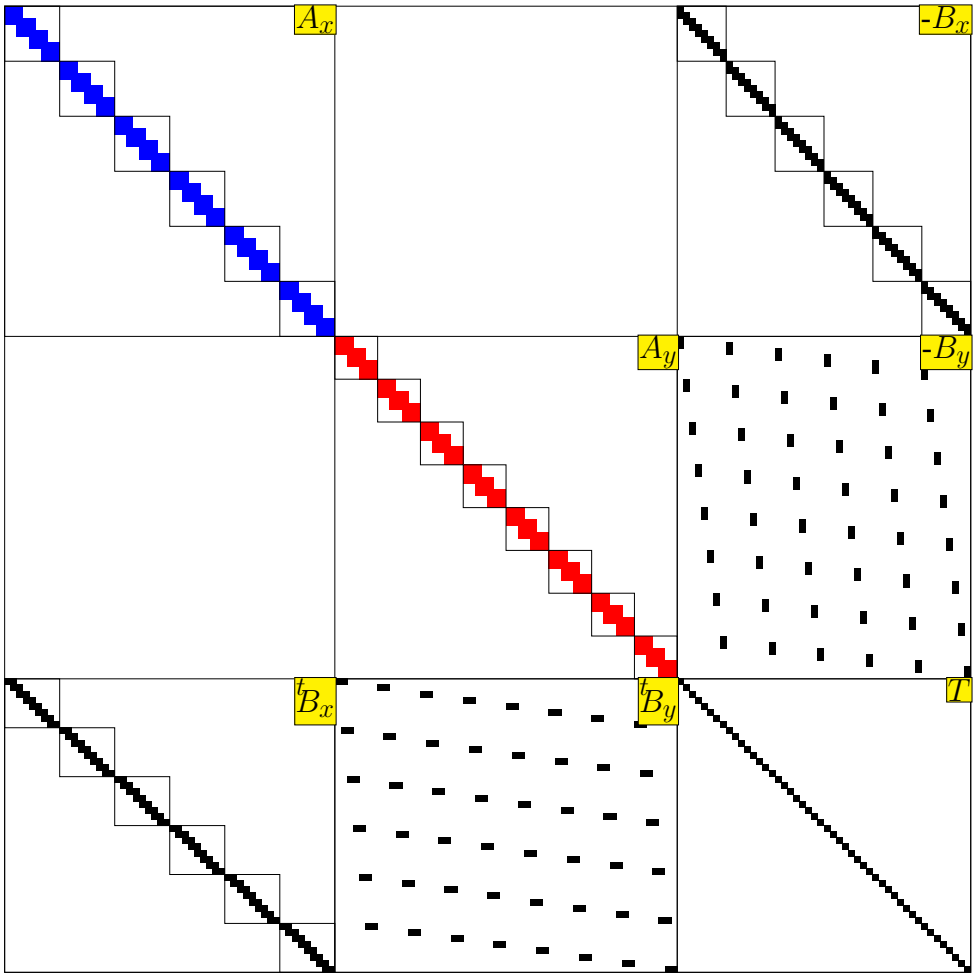


FIG. 1.7 – Profil de la matrice $H^{g=g'}$ pour le maillage défini par la figure FIG. 1.5

à la résolution d'un système linéaire défini positif, on élimine le flux ϕ à l'aide de la relation suivante

$$\phi = T^{-1}\Big(S - B_D J_D\Big).$$

Le système (1.14) devient après élimination des flux

$$\Big(A_D + B_D {T^{-1}}^t B_D\Big)\, J_D = B_D T^{-1} S\,.$$

La matrice $W = \Big(A_D + B_D {T^{-1}}^t B_D\Big)$ est alors symétrique définie positive.

Preuve :

- W est symétrique : car A_D et T le sont ;
- W est définie positive : car A_D et T^{-1} le sont. En effet :
  Soit $X \neq 0$, on a $< WX, X >=< A_D X, X > + < {T^{-1}}^t B_D X, {}^t B_D X >$.
  - comme A_D est symétrique définie positive : $\forall X \neq 0 \quad < A_D X, X > > 0$;
  - comme T^{-1} est symétrique définie positive : $\forall Y \neq 0 \quad < T^{-1} Y, Y > > 0$.

  En posant $Y = {}^t B_D X$, on obtient $< {T^{-1}}^t B_D X, {}^t B_D X > \geq 0$ et donc le caractère défini positif de W.

  [CQFD]

Cette opération d'élimination des flux effectuée avec des notations où les directions de l'espace sont détaillées aboutit aux équations suivantes

$$\phi = T^{-1}\Big(S - \sum_{d\in\{x,y,z\}} {}^t B_d J_d\Big),$$

$$\begin{pmatrix} W_x & B_x {T^{-1}}^t B_y & B_x {T^{-1}}^t B_z \\ B_y {T^{-1}}^t B_x & W_y & B_y {T^{-1}}^t B_z \\ B_z {T^{-1}}^t B_x & B_z {T^{-1}}^t B_y & W_z \end{pmatrix} \begin{pmatrix} J_x \\ J_y \\ J_z \end{pmatrix} = \begin{pmatrix} B_x T^{-1} S \\ B_y T^{-1} S \\ B_z T^{-1} S \end{pmatrix}, \tag{1.17}$$

avec

$$\forall d \in \{x, y, z\} \quad W_d = A_d + B_d {T^{-1}}^t B_d.$$

De par la nature du couplage flux/courant dans la direction d, la matrice $B_d {T^{-1}}^t B_d$ est bloc-diagonale avec des blocs tridiagonaux (correspondant chacun à une ligne de courant). Le profil de $B_d {T^{-1}}^t B_d$ est donc inclus dans celui A_d. Il est donc possible de stocker les matrices W_d en lieu et place des matrices A_d.

Le système (1.17) est résolu à l'aide d'un algorithme de Gauss-Seidel par bloc. A chaque itération, on résout un système linéaire impliquant chaque matrice W_d. La structure des matrices W_d conduit naturellement à résoudre ces systèmes de façon exacte *via* une factorisation de Cholesky. Cette factorisation ne crée pas de termes de remplissage car la struture des facteurs est identique à celle de A_d.

Cet algorithme où les flux sont éliminés et les courants sont résolus par bloc peut être vu comme un algorithme de directions alternées pour des éléments finis mixtes [17, 18, 19].

1.6 Résolution globale via des itérations imbriquées

Pour décrire l'**Algorithme 2** au niveau matriciel, on utilise les notations suivantes :
- X^g désigne le vecteur ${}^t\left(J_x^g, J_y^g, J_z^g, \phi^g\right)$;
- X désigne le vecteur ${}^t\left(X^1, \ldots, X^{N_g}\right)$;
- $\|X\|$ désigne la norme du vecteur ${}^t\left(\phi^1, \ldots, \phi^{N_g}\right)$;
- F est la matrice de l'opérateur $\mathcal{F}$;
- H est la matrice de l'opérateur $\mathcal{H}$;
- $H^{g=g'}$ est la matrice de l'opérateur $\mathcal{H}^{g=g'}$;
- $H^{g\to g'}$ est la matrice de l'opérateur $\mathcal{H}^{g\to g'}$.

Avec ces notations, on a :

$$H = \begin{pmatrix} H^{g=1} & H^{2\to 1} \\ H^{1\to 2} & H^{g=2} \end{pmatrix}.$$

Lorsque l'on développe la résolution des systèmes matriciels à tous les niveaux (groupes d'énergie et directions de l'espace), on obtient l'**Algorithme 3**.

Dans le cas plus général des équations SPn, on ajoute un niveau d'itérations entre les groupes et les directions de l'espace (cf. **Algorithme 4**). Avec la figure Fig. 1.8, on représente le profil de la matrice H pour deux groupes d'énergie, deux harmoniques[9] (indicées par l) et deux dimensions de l'espace (maillage 5×3). Celle-ci permet de visualiser au niveau matriciel l'ajout de cette boucle : chaque matrice monogroupe ($H^{g=1}$ et $H^{g=2}$) est une matrice bloc 2×2 dont chaque bloc diagonal correspondant à une harmonique est un système de diffusion (indicé par g et l). L'**Algorithme 4** dépend de quatre critères de convergence (ligne 1, 4, 8 et 12). La littérature sur les itérations imbriquées provenant d'algorithmes de Newton, de problèmes aux valeurs propres, de compléments de Schur, de problèmes de point-selle, ou de préconditionnements de solveurs itératifs *flexibles* [20], est abondante mais donne pas de méthodes systématiques pour le calage optimal de ces critères dans le contexte d'algorithmes itératifs emboîtés à plusieurs niveaux. On peut néanmoins dégager des règles générales. Un algorithme externe de type point fixe, requiert une précision élevée du solveur interne lors des dernières itérations. Pour un solveur de type sous-espace, les premières itérations nécessitent un solveur interne précis. La précision du solveur interne pouvant être progressivement relâchée par la suite.

La plupart de travaux de la littérature essaient de contrôler la convergence de l'algorithme interne en fonction du résidu de l'algorithme externe de manière à ne pas modifier le nombre d'itérations externes. Ainsi dans [21], une stratégie concernant les algorithmes de Krylov (GMRES et BiCGStab) a été développée. Cette même stratégie a été utilisée pour le calcul de valeurs propres [22, 23, 24], et adaptée pour des problèmes issues de compléments de Schur [23]. Dans [25, 26], les auteurs fournissent un cadre théorique à ces travaux basés sur des constatations empiriques. Dans [27], les auteurs proposent pour l'algorithme de point fixe des puissances inverses sans accélération, deux critères permettant de relâcher la précision des itérations internes en conservant le même nombre d'itérations externes. Dans [28], les auteurs cherchent à réduire le nombre totale d'itérations internes, sans garder de contraintes sur le nombre d'itérations externes.

Pour notre application, les critères de convergence des solveurs internes ne sont plus basés sur les résidus mais sur un nombre d'itérations, économisant ainsi le calcul des résidus. Il a été choisi de bloquer toutes les boucles internes (lignes 4, 8 et 12) à une seule itération. Cette stratégie peut conduire à une augmentation du nombre d'itérations externes (ligne 1). Toutefois, cette augmentation du nombre d'itérations est largement compensée par le gain réalisé à chaque

[9] Calcul SP3.

Algorithme 3 : Puissance inverse (version multigroupe discrète)

▷ *Initialisation des flux à 1 et des courants à 0*
pour chaque g **faire**
  ▷ X^g *désigne* $(J_x^g, J_y^g, J_z^g, \phi^g)$
  $X^g = (0, 0, 0, 1)$;
fin
▷ *Précalcul*
pour chaque g **faire**
  pour chaque $d \in \{x, y, z\}$ **faire**
    Factorisation de W_d^g;
  fin
fin

▷ *Boucle de la puissance*
tant que *!Convergence* **faire**
  ▷ X *désigne* $(X^1, \ldots, X^{N_g})$
  $X_{old} = X$;
  ▷ *Produit par la matrice de fission*
  $S = FX$;
  ▷ *Résolution itérative du système linéaire :* $X = H^{-1}S$
  tant que *!Convergence* **faire**
    pour chaque g **faire**
      ▷ *Calcul du terme source*
      $L = S^g - \sum_{g' \neq g} H^{g' \to g} X^{g'}$;
      ▷ *Résolution itérative du système linéaire monogroupe :* $X^g = (H^{g=g})^{-1} L$
      tant que *!Convergence* **faire**
        ▷ L *désigne* (L_x, L_y, L_z, L_ϕ)
        pour chaque $d \in \{x, y, z\}$ **faire**
          ▷ *Résolution par descente-remontée*
          $J_d^g = (W_d^g)^{-1}\Big(L_d + B_d (T^g)^{-1}\big(L_\phi - \sum_{d' \neq d} {}^t B_{d'} J_{d'}^g\big)\Big)$;
        fin
      fin
      ▷ *calcul du flux*
      $\phi^g = (T^g)^{-1}\Big(L - \sum_d {}^t B_d J_d^g\Big)$;
    fin
  fin
  ▷ *Calcul de la valeur propre*
  $\lambda = \dfrac{\|X\|}{\|X_{old}\|}$;
  ▷ *Normalisation*
  $X = \dfrac{1}{\lambda}.X$;
fin

itération de l'algorithme de la puissance. Ce phénomène numérique n'est pas surprenant car l'algorithme de la puissance s'apparente à une résolution de type point fixe qui nécessite une précision croissante des calculs effectués au cours de l'algorithme. Celle-ci est obtenue par une meilleure initialisation des algorithmes itératifs internes.

Algorithme 4 : Itérations emboîtées

```
tant que !Convergence faire
    (...)
    ▷ Boucle de Gauss-Seidel sur les groupes d'énergie
    tant que !Convergence faire
        pour chaque groupe faire
            (...);
            ▷ Boucle de Gauss-Seidel sur les harmoniques
            tant que !Convergence faire
                (...);
                pour chaque harmonique faire
                    ▷ Boucle de Gauss-Seidel sur les directions de l'espace
                    tant que !Convergence faire
                        pour chaque d ∈ {x, y, z} faire
                            ▷ Résolution par descente-remontée
                        fin
                    fin
                    ▷ calcul du flux
                fin
            fin
        fin
    fin
    (...)
fin
```

1.7 Présentation des cas tests considérés

Pour tester les méthodes de décomposition de domaine proposées (cf. chapitre 4 et 5), un cas test académique et un cas test industriel sont utilisés. Au cours de la présentation du premier cas test, quelques propriétés du solveur sont mises en évidence.

1.7.1 Cas test BenchMarkAIEA

Le cas test académique *BenchMarkAIEA* [29] représente un cœur 3D avec deux types de combustible et deux hauteurs d'insertion de barre. Le cœur est constitué de cinq matériaux placés suivant la configuration définie par la figure Fig. 1.9. Ces matériaux sont :

- Combustible 1 ;
- Combustible 2 ;
- Combustible 1 barré (avec des barres de contrôle) ;
- Réflecteur ;
- Réflecteur barré.

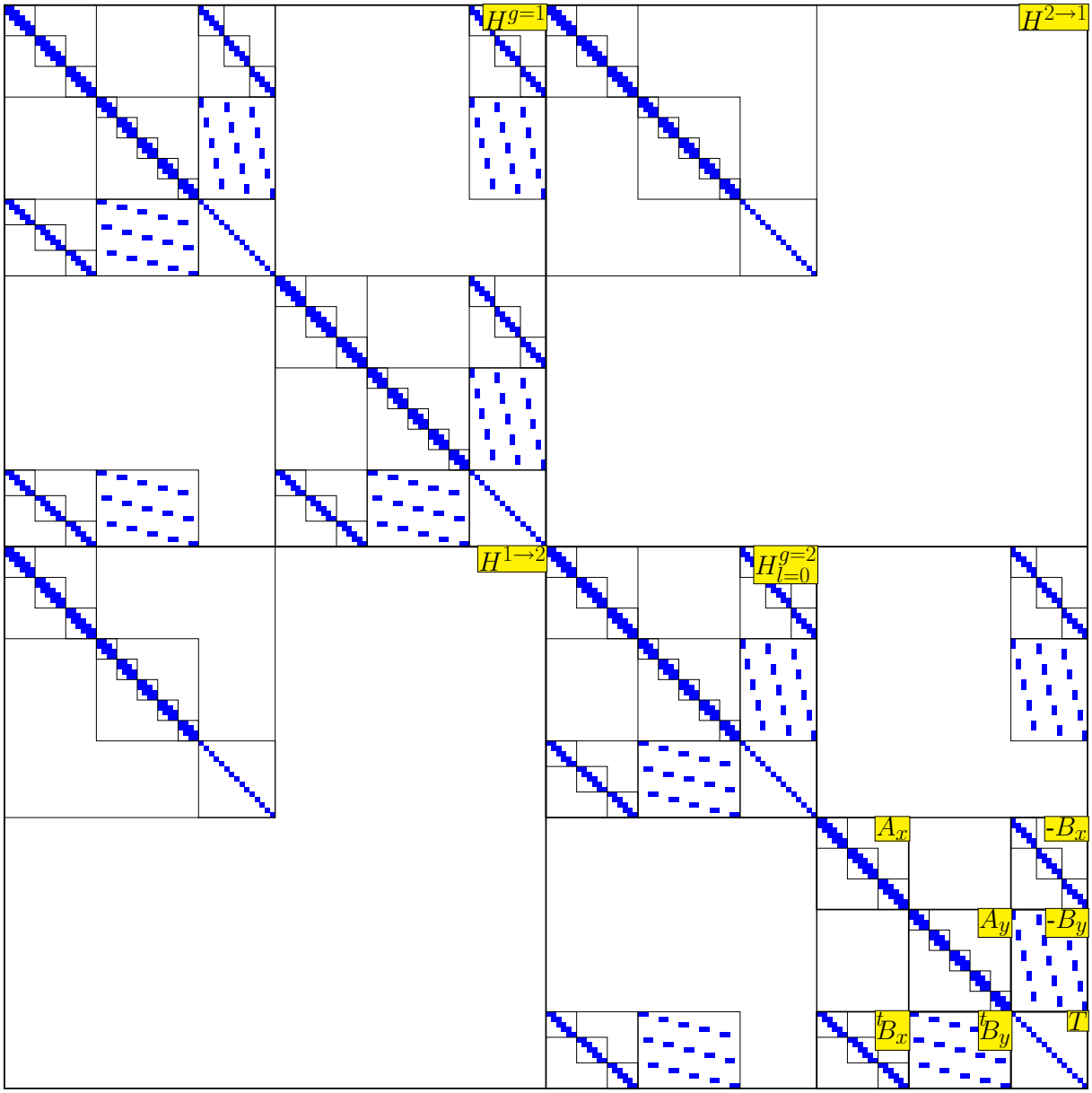


FIG. 1.8 – Profil de la matrice SP3 à 2 groupes d'énergie

équations	élément fini	itérations externes	k_{eff}	temps (en s)	nombre de ddl
SP1	RT0	78	1.02898	145.37	25 601 354
SP3	RT0	79	1.02946	335.285	51 202 708
SP1	RT1	78	1.02903	1173.49	203 966 952
SP3	RT1	X	X	X	407 933 904

TAB. 1.1 – BenchmarkAIEA : résultats avec le solveur séquentiel. X signifie que l'on n'est pas capable de faire tourner ce cas sur un nœud du cluster `rendvous` (cf. annexe D).

Les valeurs des sections pour chacun des matériaux sont données dans le tableau TAB. 1.2. Comme le solveur ne permet de gérer que des maillages cartésiens sur une géométrie parallélépipédique, la géométrie est complétée avec des assemblages de type réflecteur (cf. figure FIG. 1.10). On considère un maillage crayon par crayon[10] : $289 \times 289 \times 38$. Comme tous les algorithmes de Gauss-Seidel sont bloqués à une itération le seul critère de convergence concerne l'algorithme de la puissance

$$\frac{\|S^n - S^{n-1}\|}{\|S^n\|} < \epsilon \quad \text{et} \quad \left|\frac{\lambda^n - \lambda^{n-1}}{\lambda^n}\right| < \epsilon, \tag{1.18}$$

où

- S^n est la source de fission $(F\psi)$ à l'itération n de la puissance ;
- λ^n est l'estimation de la valeur propre à l'itération n de la puissance ;
- ϵ est fixé à 10^{-6}.

Les résultats obtenus pour les valeurs de k_{eff}, le nombre d'itérations de puissance et le temps d'exécution sont reportés dans le tableau TAB. 1.1. Les flux intégrés par maille dans le cas SP1/RT0 sont représentés sur la figure FIG. 1.11.

En termes de complexité, le solveur est linéaire. En effet, sur la figure FIG. 1.12, on reporte le nombre d'itérations de puissance pour les discrétisations suivantes :

$$\{(17.i) \times (17.i) \times (19.j) \quad | \quad i \in [\![1, 17]\!], \quad j \in [\![1, 2]\!]\}.$$

On remarque que le nombre d'itérations externes peut être considéré comme constant pour un problème physique donné, en négligeant les discrétisations très grossières (nombre de mailles inférieur à 300 000). Le nombre d'opérations est donc une fonction linéaire du nombre de mailles du maillage, car l'inversion approchée de la matrice H par les algorithmes de Gauss-Seidel bloqués à une itération est linéaire. En termes temps d'exécution, on retrouve cette complexité linéaire sauf pour les domaines de petite taille où l'on bénéficie d'effets de cache (cf. figure FIG. 1.13).

[10] On considère 17×17 assemblages, chacun constitué de 17×17 emplacements essentiellement occupés par des crayons (cf. page 5).

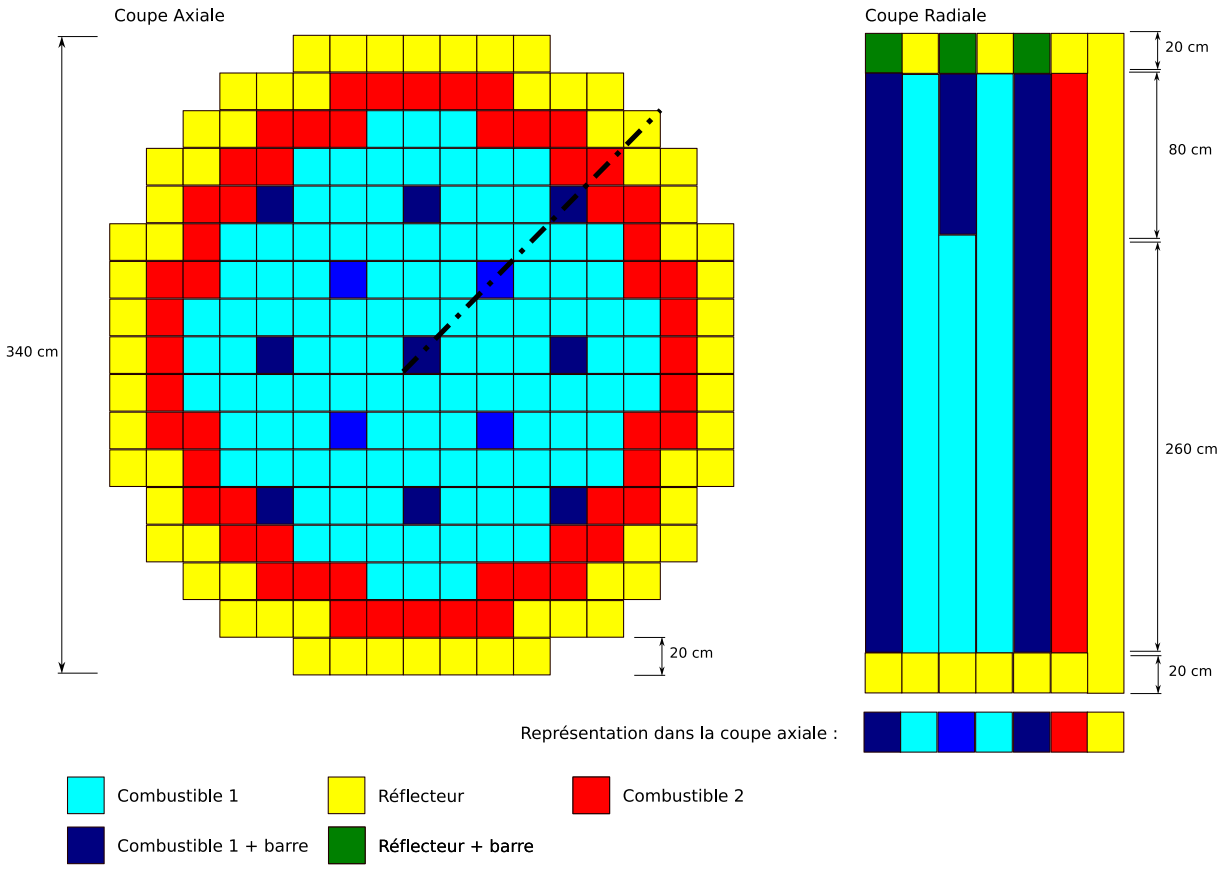


FIG. 1.9 – BenchMarkAIEA : géométrie réelle

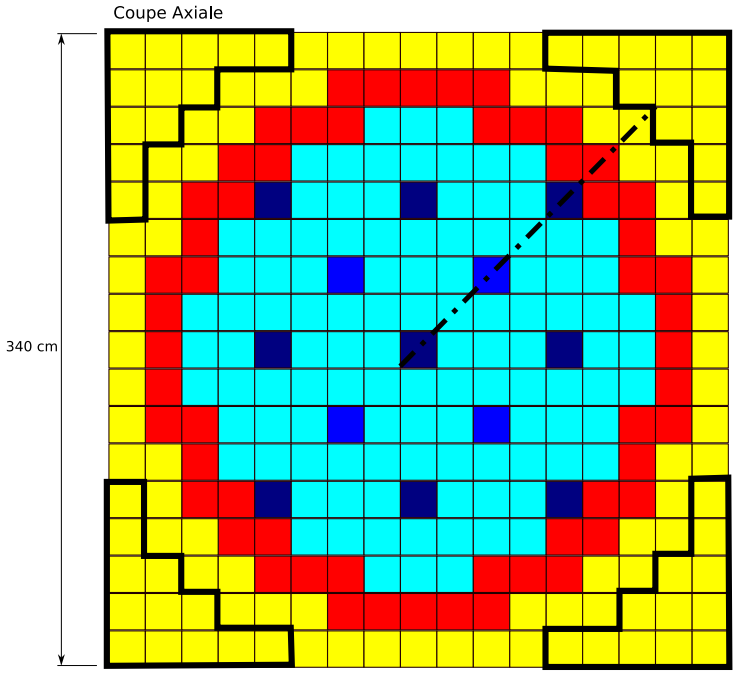
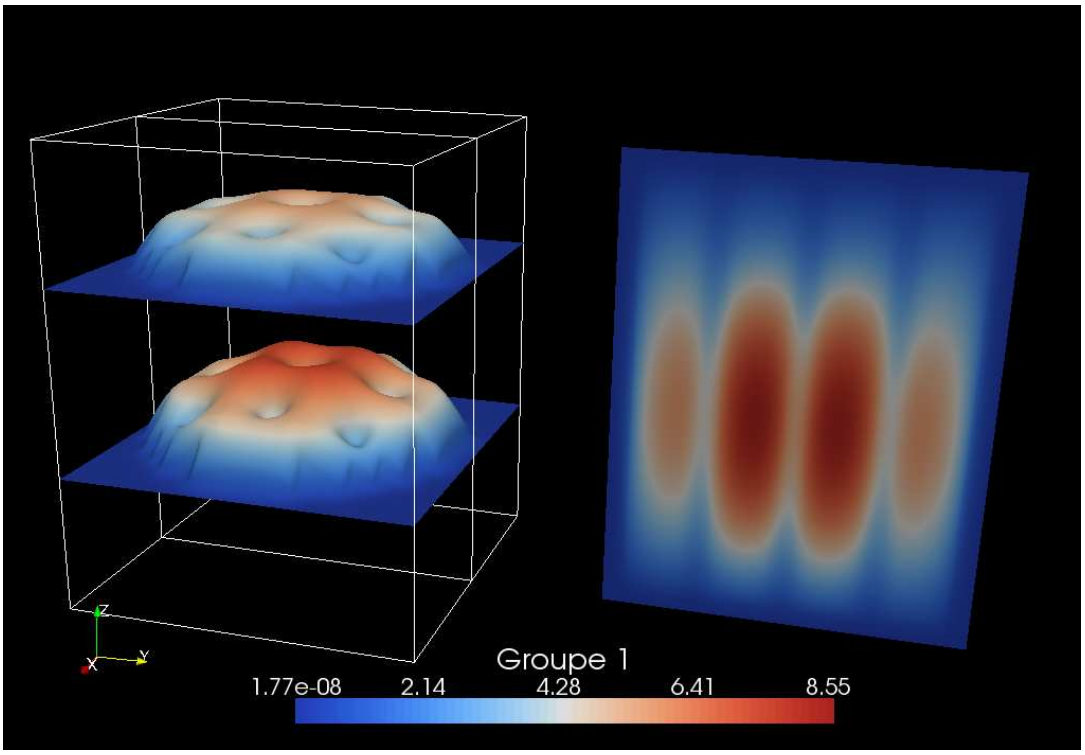
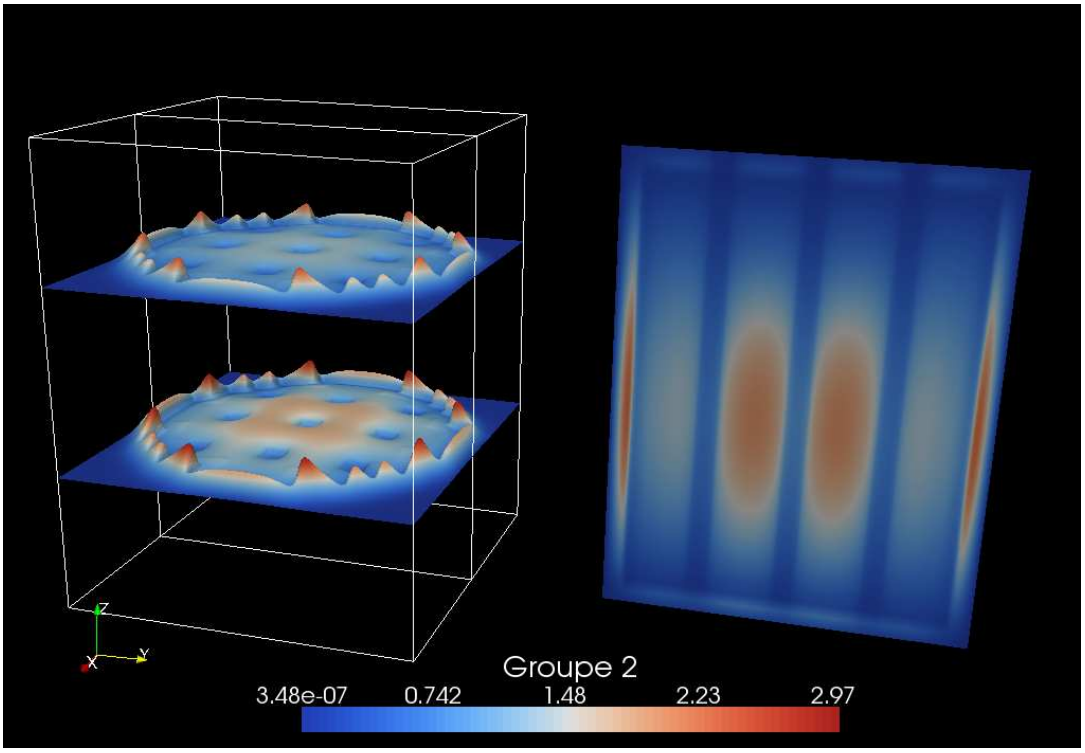


FIG. 1.10 – BenchMarkAIEA : géométrie de calcul



(a) Groupe rapide



(b) Groupe thermique

FIG. 1.11 – BenchMarkAIEA : flux intégré

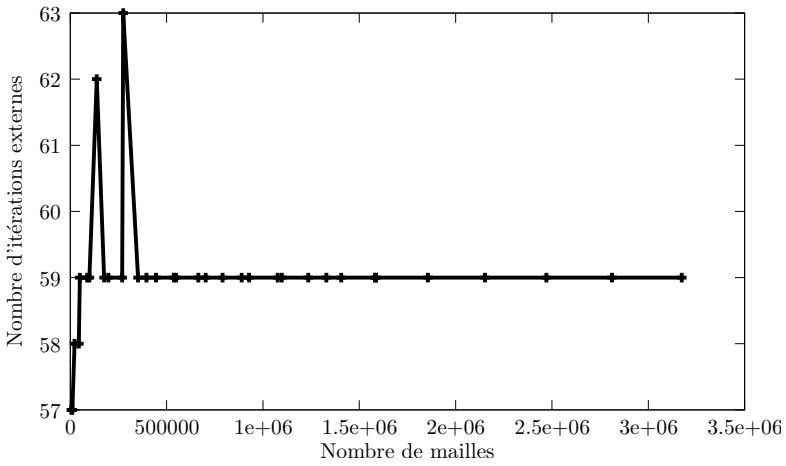


FIG. 1.12 – BenchMarkAIEA : nombre d'itérations en fonction de la discrétisation

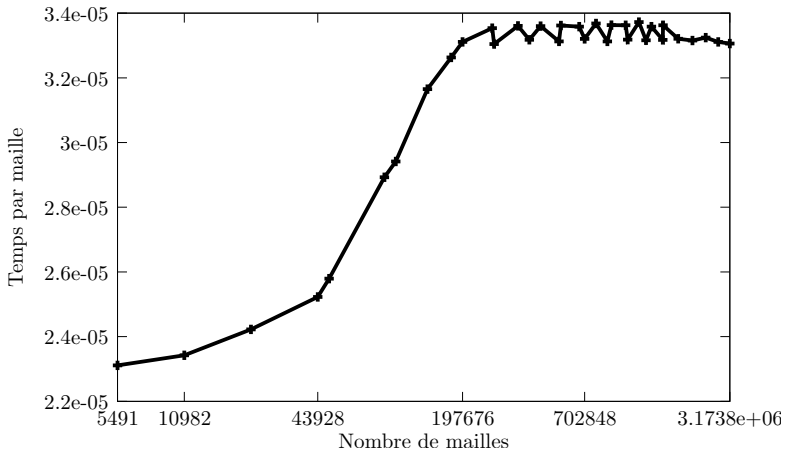


FIG. 1.13 – BenchMarkAIEA : temps par itération de puissance rapporté au nombre de mailles. L´axe des abscisses est en échelle logarithmique, pour mettre en évidence les effets de cache avec les problèmes de petite taille.

	Groupe 1 (rapide)			Groupe 2 (thermique)		
Matériau	Σ_t	$\nu\Sigma_f$	χ	Σ_t	$\nu\Sigma_f$	χ
Combustible 1	0.222222222	0	1	0.833333333	0.135	0
Combustible 1 + barre	0.222222222	0	1	0.833333333	0.135	0
Combustible 2	0.222222222	0	1	0.833333333	0.135	0
Réflecteur	0.166666667	0	1	1.111111111	0	0
Réflecteur+barre	0.166666667	0	1	1.111111111	0	0

	Scattering*			
Matériau	$\Sigma_s^{1\to1}$	$\Sigma_s^{1\to2}$	$\Sigma_s^{2\to1}$	$\Sigma_s^{2\to2}$
Combustible 1	0.192222222	0.02	0	0.7483333333
Combustible 1 + barre	0.192222222	0.02	0	0.7033333333
Combustible 2	0.192222222	0.02	0	0.7533333333
Réflecteur	0.126666667	0.04	0	1.1011111111
Réflecteur+barre	0.126666667	0.04	0	1.0561111111

* On considère des sections de scattering développées à l'ordre 0.

TAB. 1.2 – BenchMarkAIEA : propriétés des matériaux

1.7.2 Cas test basé sur un REP 900MW

Afin de pouvoir tester la méthode sur un cas associé à des données industrielles, on considère un REP 900MW avec les données de la campagne Gravelines 514 [30, 31]. Elle contient 3 types d'assemblages enrichis en U235 à 3.70% :

- 152 assemblages de type ANFH dont 8 stockés en piscine pendant 1.3 ans ;
- 4 assemblages AFG stockés en piscine pendant 2.5 ans ;
- 1 assemblage FRAM stockés en piscine pendant 7.7 ans.

La figure FIG. 1.14 donne la disposition de ces assemblages. Chacun de ces assemblages est maillé radialement en 17 par 17. Le maillage axial est constitué de 40 mailles :

- le réflecteur en haut et en bas de cœur est traité avec 2×4 mailles : la taille des mailles (donnée en cm) est plus petite lorsqu'elle est proche d'un assemblage (`[75, 10, 10, 5]`) ;
- les assemblages sont maillés avec 32 mailles de haut en bas :

```
[1.909, 3.810, 14.526, 14.526, 14.526, 14.526,
 3.81, 16.192, 16.192, 16.192,   3.81, 16.192, 16.192, 16.192,
 3.81, 16.192, 16.192, 16.192,   3.81, 16.192, 16.192, 16.192,
 3.81, 16.192, 16.192, 16.192,   3.81, 12.065, 12.065, 05.815, 3.75, 2.5]
```

  Les mailles de taille `3.81` correspondent aux grilles de mélange.

A partir de ces données, on a effectué une recherche de Bore critique[11] avec la plate-forme Cocagne. A la sixième et dernière itération de Bore, on a stocké, lors de l'entrée dans le solveur SPn, les sections dans des fichiers. Malgré une opération d'homogénéisation sur les assemblages, les données de chaque crayon sont *a priori* différentes suite aux contre-réactions et aux gradients d'irradiation des combustibles.

Le calcul de Bore ayant été effectué en SP1/RT0, il est attendu que les résultats obtenus (cf. tableau TAB. 1.3) soient plus proches de la criticité en SP1/RT0. On remarque qu'avec ces

[11] On calcule par un algorithme itératif, la concentration en Bore du réacteur permettant d'obtenir la criticité ($k_{eff} = 1$) : cf. page 96

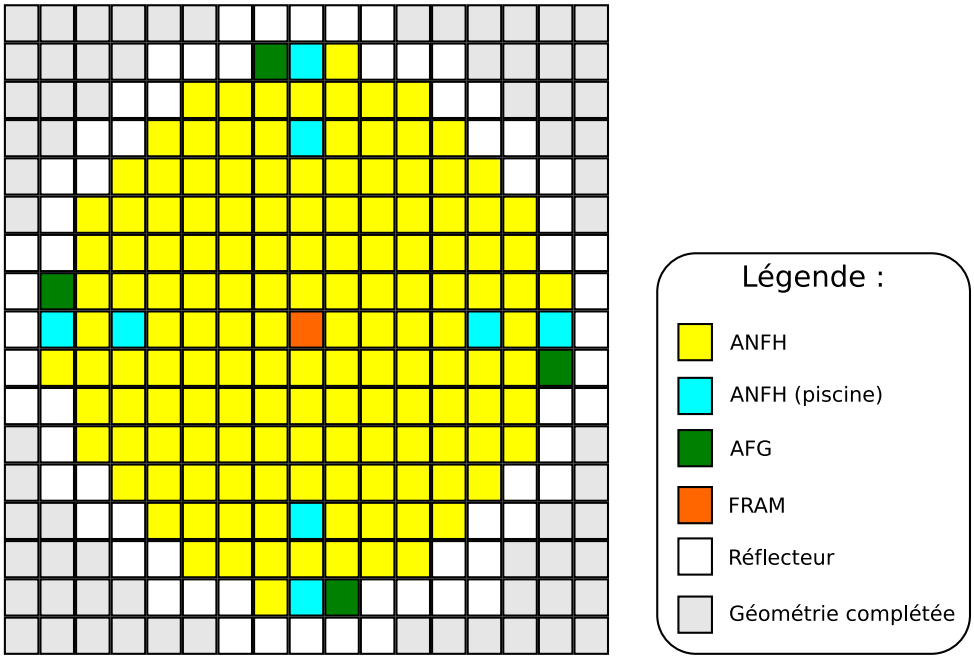
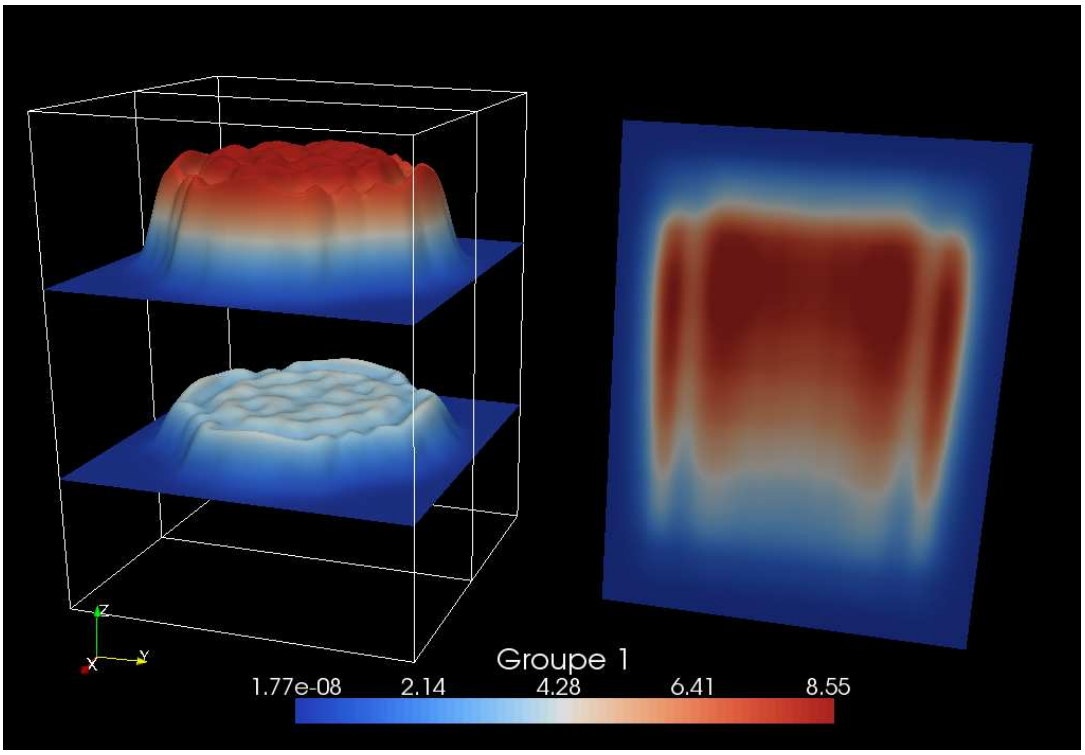


FIG. 1.14 – REP 900MW : disposition des assemblages

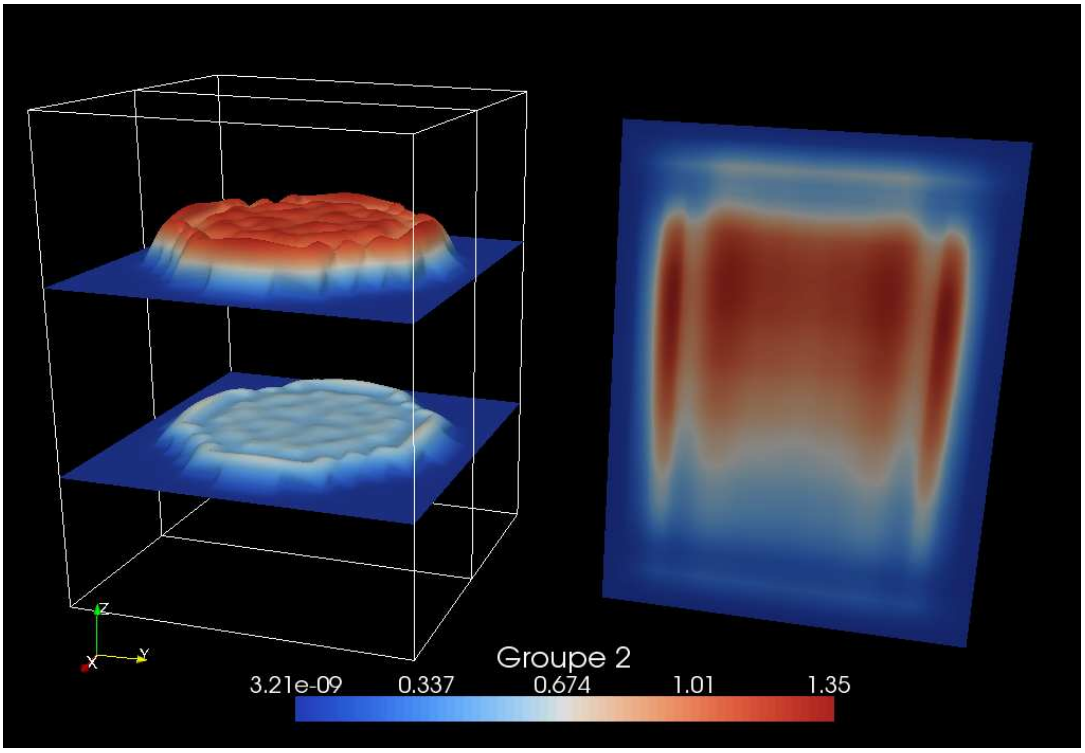
équations	élément fini	itérations externes	k_{eff}	temps (en s)	nombre de ddl
SP1	RT0	88	0.999994	248.577	26 940 002
SP3	RT0	90	1.00102	525.929	53 880 004
SP1	RT1	88	0.999951	2198.18	214 666 888
SP3	RT1	X	X	X	429 333 776

TAB. 1.3 – REP 900MW : résultats avec le solveur séquentiel

données plus complexes, le solveur séquentiel nécessite plus d'itérations de puissance que pour le cas test BenchMarkAIEA, alors que le critère de convergence est moins stricte ($\epsilon = 10^{-5}$). On représente les flux intégrés par maille sur la figure FIG. 1.15.



(a) Groupe rapide



(b) Groupe thermique

FIG. 1.15 – REP 900MW : flux intégré

Chapitre 2

État de l'art des méthodes de décomposition de domaine autour des équations de transport simplifié

Sommaire

Les méthodes de décomposition de domaine expriment un problème global à l'aide plusieurs problèmes locaux plus petits. Elles répondent donc à un ou plusieurs des objectifs suivants :

1. diminuer l'empreinte mémoire d'un solveur séquentiel en utilisant des motifs se répétant ;
2. diminuer le temps de résolution séquentiel lorsque la complexité du solveur séquentiel est surlinéaire ;
3. permettre de passer en mémoire, en utilisant des machines à mémoire distribuée, des cas trop importants sur une machine à mémoire partagée ;
4. diminuer le temps de résolution en utilisant une algorithmique parallèle ;
5. obtenir plus de souplesse dans la discrétisation du problème. En effet, elles autorisent pour certaines d'utiliser des tailles de maillage différentes, des ordres d'éléments finis différents, ou encore des modèles physiques différents dans chaque sous-domaine.

Malgré la géométrie des assemblages, il est *a priori* difficile de trouver au niveau des sections efficaces des motifs se répétant. Comme la complexité du solveur SPn est linéaire, on ne peut tirer parti des deux premiers objectifs. Au cours de cette thèse, on cherche à mettre en œuvre une méthode de décomposition de domaine pour les équations SPn répondant aux objectifs 3 et 4. Dans un deuxième temps, postérieur à cette thèse, on souhaite également faire évoluer cette méthode pour pouvoir choisir différents paramètres de discrétisation dans différentes parties du cœur.

Ce chapitre n'a pas pour objectif de faire un état de l'art complet des méthodes de décomposition de domaine. Le lecteur souhaitant une vue plus complète de ces méthodes est invité à se référer à [32, 33, 34, 35, 36, 37]. Afin de permettre la compréhension des méthodes existantes en neutronique et des méthodes proposées dans cette thèse, on présente succinctement les principales méthodes de décomposition de domaine. Enfin on présente l'ensemble des tentatives de parallélisation du solveur SPn basé sur les éléments finis de Raviart-Thomas.

2.1 Présentation succincte des méthodes de décomposition de domaine

On considère un domaine connexe noté Ω et l'équation aux dérivées partielles suivante :

$$\begin{cases} Lu = f & \text{dans} \quad \Omega \\ u = 0 & \text{sur} \quad \partial\Omega. \end{cases}$$

Les méthodes de décomposition de domaine sont basées sur un partionnement du domaine Ω en n ($n \geq 2$) sous-domaines Ω_i tel que :

$$\Omega = \bigcup_{1 \leq i \leq n} \Omega_i.$$

Ces méthodes visent à résoudre le problème global sur Ω en résolvant des problèmes plus petits sur les sous-domaines Ω_i. On distingue deux types de partitionnement :

- **sans recouvrement :** l'intersection entre les sous-domaines se limite aux interfaces (cf. figure Fig. 2.1) :
$$\forall\, 1 \leq i < j \leq n\,, \quad \Omega_i \cap \Omega_j = \partial\Omega_i \cap \partial\Omega_j.$$
On note $\Gamma_{i,j}$ la frontière entre les sous-domaines Ω_i et Ω_j. Dans le cas à deux sous-domaines, l'interface $\Gamma_{1,2}$ est notée Γ ;
- **avec recouvrement :** chaque sous-domaine recouvre une partie de ses sous-domaines voisins (cf. figure Fig. 2.2).

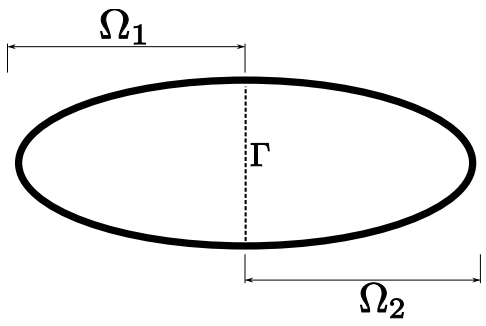


FIG. 2.1 – Décomposition sans recouvrement

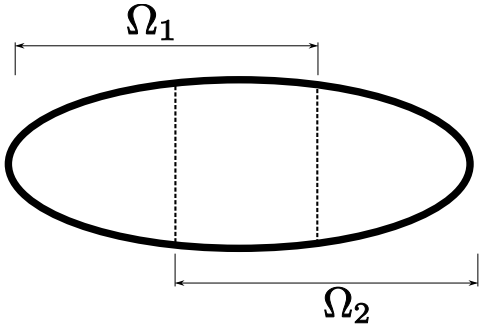


FIG. 2.2 – Décomposition avec recouvrement

Algorithme 5 : Schwarz additif

tant que *!Convergence* **faire**

Résoudre :

$$\begin{cases} Lu_1^{n+1} = f & \text{dans } \Omega_1 \\ u_1^{n+1} = 0 & \text{sur } \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} = u_2^n & \text{sur } \Gamma; \end{cases}$$

Résoudre :

$$\begin{cases} Lu_2^{n+1} = f & \text{dans } \Omega_2 \\ u_2^{n+1} = 0 & \text{sur } \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} = u_1^n & \text{sur } \Gamma; \end{cases}$$

fin

Algorithme 6 : Schwarz multiplicatif

tant que *!Convergence* **faire**

Résoudre :

$$\begin{cases} Lu_1^{n+1} = f & \text{dans } \Omega_1 \\ u_1^{n+1} = 0 & \text{sur } \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} = u_2^n & \text{sur } \Gamma; \end{cases}$$

Résoudre :

$$\begin{cases} Lu_2^{n+1} = f & \text{dans } \Omega_2 \\ u_2^{n+1} = 0 & \text{sur } \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} = u_1^{n+1} & \text{sur } \Gamma; \end{cases}$$

fin

2.1.1 Méthodes de type Schwarz

Les algorithmes dits de Schwarz appartiennent à la famille des méthodes de décomposition de domaine avec recouvrement. Ils consistent en une résolution alternée de l'équation différentielle dans chacun des sous-domaines avec des conditions aux limites de type de Dirichlet provenant des sous-domaines voisins[12]. On distingue deux versions :

- l'algorithme dit additif (cf. **Algorithme 5**) : on prend comme conditions de Dirichlet pour un sous-domaine Ω_i les valeurs données par le domaine voisin à l'itération précédente ;
- l'algorithme dit multiplicatif (cf. **Algorithme 6**) : on prend comme conditions de Dirichlet les dernières valeurs d'interface calculées par le sous-domaine voisin.

On peut faire une analogie directe de ces deux méthodes avec les algorithmes de Jacobi et de Gauss-Seidel pour la résolution d'un système linéaire. A l'instar de l'algorithme de Jacobi, l'algorithme de Schwarz additif est directement parallélisable car les résolutions dans chaque sous-domaine sont indépendantes.

Pour améliorer la convergence des algorithmes de type Schwarz, on généralise les conditions de transfert entre les sous-domaines : les conditions de type Dirichlet $u_i^{n+1} = u_j^n$ sont remplacées par un opérateur de type Robin aboutissant à l'**Algorithme 7**. Pour $\alpha > 0$, l'algorithme ne nécessite plus de recouvrement pour converger. Par ailleurs, cet algorithme converge plus rapidement que les algorithmes de Schwarz standards pour de bonnes valeurs de α. Pour des cas académiques, on détermine de manière analytique la valeur optimale de α [38]. Dans les autres

[12] Ces conditions viennent en plus des conditions dites *externes* concernant $\partial\Omega \cap \partial\Omega_i$.

cas, la détermination d'une bonne valeur de α s'effectue de manière empirique.

Algorithme 7 : Schwarz additif avec conditions de Robin ($\alpha > 0$)

tant que *!Convergence* **faire**

Résoudre :
$$\begin{cases} Lu_1^{n+1} = f & \text{dans } \Omega_1 \\ u_1^{n+1} = 0 & \text{sur } \partial\Omega_1 \cap \partial\Omega \\ \dfrac{\partial u_1^{n+1}}{\partial n_1} + \alpha u_1^{n+1} = \dfrac{\partial u_2^n}{\partial n_1} + \alpha u_2^n & \text{sur } \Gamma; \end{cases}$$

Résoudre :
$$\begin{cases} Lu_2^{n+1} = f & \text{dans } \Omega_2 \\ u_2^{n+1} = 0 & \text{sur } \partial\Omega_2 \cap \partial\Omega \\ \dfrac{\partial u_2^{n+1}}{\partial n_2} + \alpha u_2^{n+1} = \dfrac{\partial u_1^n}{\partial n_2} + \alpha u_1^n & \text{sur } \Gamma; \end{cases}$$

fin

2.1.2 Méthodes basées sur le complément de Schur

Il s'agit de méthodes sans recouvrement. Dans les méthodes basées sur le complément de Schur on cherche à déterminer une grandeur au niveau de l'interface. La connaissance de cette grandeur permet ensuite de calculer aisément la solution du système global. On se ramène donc à la résolution d'un problème plus petit en termes de nombre de degrés de liberté. On distingue deux types de méthodes dites de Schur :

la méthode de Schur primal : on choisit comme variable d'interface l'inconnue u à calculer. On résout dans chaque sous-domaine le problème suivant ($i \in \{1, 2\}$) :

$$\begin{cases} Lu_i = f & \text{dans } \Omega_i \\ u_i = 0 & \text{sur } \partial\Omega_i \cap \partial\Omega \\ u_i = u_\Gamma & \text{sur } \Gamma. \end{cases}$$

La méthode consiste à trouver u_Γ vérifiant la condition supplémentaire : $\dfrac{\partial u_1}{\partial n_1} + \dfrac{\partial u_2}{\partial n_2} = 0$.

la méthode de Schur dual : on résout dans chacun des sous-domaines un problème avec des conditions limites de Neumann ($i \in \{1, 2\}$) :

$$\begin{cases} Lu_i = f & \text{dans } \Omega_i \\ u_i = 0 & \text{sur } \partial\Omega_i \cap \partial\Omega \\ \dfrac{\partial u_i}{\partial n_i} = (-1)^i \Lambda & \text{sur } \Gamma. \end{cases}$$

Le terme $(-1)^i$ provient du fait que les normales sortantes des sous-domaines Ω_1 et Ω_2 sont de signe contraire sur l'interface Γ. L'ajout de la condition $u_1 = u_2$ sur Γ permet de déterminer Λ.

Nous présentons maintenant ces deux méthodes sous forme matricielle.

La méthode de Schur primal

On recherche la valeur de u au niveau de l'interface Γ. Cette méthode consiste simplement à réordonner les inconnues en regroupant les degrés de liberté internes au sous-domaine Ω_1, puis

au sous-domaine Ω_2 et enfin ceux de l'interface Γ. On aboutit ainsi au système matriciel suivant :

$$\begin{pmatrix} A_{11} & 0 & A_{1\Gamma} \\ 0 & A_{22} & A_{2\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_\Gamma \end{pmatrix}.$$

Si A_{11} et A_{22} sont inversibles, on se ramène à la résolution d'un système linéaire sur u_Γ

$$Su_\Gamma = b \quad \text{avec} \quad \begin{cases} S = A_{\Gamma\Gamma} - A_{\Gamma 1}A_{11}^{-1}A_{1\Gamma} - A_{\Gamma 2}A_{22}^{-1}A_{2\Gamma} \\ b = f_\Gamma - A_{\Gamma 1}A_{11}^{-1}f_1 - A_{\Gamma 2}A_{22}^{-1}f_2 \end{cases}.$$

Une fois ce système résolu (généralement par une méthode itérative préconditionnée), on calcule la solution sur chacun des sous-domaines Ω_i *via* la relation suivante

$$u_i = A_{ii}^{-1}(f_i - A_{i\Gamma}u_\Gamma). \tag{2.1}$$

Les appels aux solveurs locaux A_{ii}^{-1} utilisés lors des produits par la matrice d'interface S provenant de la méthode itérative ou du calcul des solutions locales (2.1) peuvent être résolus par une méthode directe ou par une méthode itérative.

La méthode de Schur dual

On décompose $A_{\Gamma\Gamma}$ et f_Γ en une somme provenant des contributions de chacun des sous-domaines : $A_{\Gamma\Gamma} = A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2$ et $f_\Gamma = f_\Gamma^1 + f_\Gamma^2$. On cherche à résoudre le système suivant

$$\begin{pmatrix} A_{11} & A_{1\Gamma} & \\ A_{\Gamma 1} & A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2 & A_{2\Gamma} \\ & A_{\Gamma 2} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_\Gamma \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_\Gamma^1 + f_\Gamma^2 \\ f_2 \end{pmatrix}. \tag{2.2}$$

On duplique les degrés de liberté au niveau de l'interface en posant $u_\Gamma^1 = u_\Gamma^2 = u_\Gamma$, puis on découple les contributions venant de chacun des sous-domaines

$$\begin{pmatrix} A_{11} & A_{1\Gamma} & & \\ A_{\Gamma 1} & A_{\Gamma\Gamma}^1 & & \\ & & A_{\Gamma\Gamma}^2 & A_{2\Gamma} \\ & & A_{\Gamma 2} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_\Gamma^1 \\ u_\Gamma^2 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_\Gamma^1 + f_\Gamma^2 - A_{\Gamma\Gamma}^2 u_\Gamma - A_{2\Gamma}u_2 \\ f_\Gamma^2 + f_\Gamma^1 - A_{\Gamma\Gamma}^1 u_\Gamma - A_{\Gamma 1}u_1 \\ f_2 \end{pmatrix}. \tag{2.3}$$

On définit le vecteur d'interface Λ par deux expressions[13]

$$\begin{aligned} \Lambda &= -(f_\Gamma^2 - A_{\Gamma\Gamma}^2 u_\Gamma - A_{2\Gamma}u_2), \\ &= f_\Gamma^1 - A_{\Gamma\Gamma}^1 u_\Gamma - A_{1\Gamma}u_1. \end{aligned}$$

L'équation (2.3) s'écrit dans chaque sous-domaine Ω_i

$$\begin{pmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} u_i \\ u_\Gamma^i \end{pmatrix} = \begin{pmatrix} f_i \\ f_\Gamma^i \end{pmatrix} - \begin{pmatrix} 0 \\ (-1)^i \Lambda \end{pmatrix}.$$

Pour simplifier les notations, on pose

$$A_i = \begin{pmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma}^i \end{pmatrix}, \quad F_i = \begin{pmatrix} f_i \\ f_\Gamma^i \end{pmatrix}, \quad U_i = \begin{pmatrix} u_i \\ u_\Gamma^i \end{pmatrix}, \quad C_i = \begin{pmatrix} 0 \\ (-1)^i I_d \end{pmatrix}$$

[13] La justification de l'égalité de ces deux expressions provient de la deuxième équation de (2.2) : $f_\Gamma^1 - A_{\Gamma\Gamma}^1 u_\Gamma - A_{1\Gamma}u_1 + (f_\Gamma^2 - A_{\Gamma\Gamma}^2 u_\Gamma - A_{2\Gamma}u_2) = f_\Gamma - A_{\Gamma\Gamma}u_\Gamma - A_{1\Gamma}u_1 - A_{2\Gamma}u_2 = 0$.

où I_d désigne la matrice identité. On remarque que résoudre $AU_i = F$ revient à résoudre une équation du même type que le problème global dans chaque sous-domaine Ω_i

$$A_i U_i = F_i - C_i \Lambda. \tag{2.4}$$

Par ailleurs, l'égalité $u_\Gamma^1 = u_\Gamma^2$ s'écrit

$$^tC_1 U_1 + {}^tC_2 U_2 = 0. \tag{2.5}$$

Les équations (2.4) et (2.5) conduisent donc à la résolution d'un problème dit de point-selle

$$\begin{pmatrix} A_1 & & C_1 \\ & A_2 & C_2 \\ {}^tC_1 & {}^tC_2 & \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \Lambda \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ 0 \end{pmatrix}. \tag{2.6}$$

Ce système est le plus souvent résolu en traitant dans un premier temps le problème d'interface[14]

$$\Big(\sum {}^tC_i A_i^{-1} C_i\Big)\Lambda = \sum {}^tC_i A_i^{-1} F_i.$$

Le calcul des U_i s'effectue ensuite en utilisant la relation (2.4).

Pour résoudre le problème d'interface, on utilise souvent des méthodes itératives ne nécessitant que savoir faire le produit par la matrice. Ainsi il n'est pas nécessaire de construire le complément de Schur $\sum {}^tC_i A_i^{-1} C_i$. Le fait qu'il suffise de faire appel aux solveurs locaux constitue un avantage certain, puisque le code du solveur local est le même que le solveur séquentiel global (méthode directe ou itérative). Cet avantage est à modérer par le fait qu'il faille souvent développer un préconditionneur pour le système d'interface.

Lorsque que les matrices A_i^{-1} ne sont pas inversibles[15], on est amené à introduire des pseudo-inverses comme c'est le cas pour les méthodes FETI [36, 39].

2.2 Parallélisation de la méthode Minos

2.2.1 Première tentative

Afin de paralléliser le solveur de diffusion (voir [40]), l'auteur décompose la matrice de diffusion en trois parties, une pour l'intérieur (indicé par o) du sous-domaine Ω_1, une pour l'intérieur du sous-domaine Ω_2 et une pour l'interface

$$\begin{pmatrix} A_1^o & B_1 & & & A_1^\Gamma \\ {}^tB_1 & T_1 & & & \\ & & A_2^o & B_2 & A_2^\Gamma \\ & & {}^tB_2 & T_2 & \\ {}^tA_1^\Gamma & & {}^tA_2^\Gamma & & A_{\Gamma\Gamma} \end{pmatrix} \cdot \begin{pmatrix} J_1^o \\ \phi_1 \\ J_2^o \\ \phi_2 \\ J_\Gamma \end{pmatrix} = \begin{pmatrix} 0 \\ S_1 \\ 0 \\ S_2 \\ 0 \end{pmatrix}. \tag{2.7}$$

Ce système est résolu par des algorithmes de Jacobi par bloc avec

- 2 blocs : un pour le sous-domaine Ω_1 et l'interface Γ et un pour le sous-domaine Ω_2 ;
- 3 blocs : un par sous-domaine et un pour l'interface.

[14] On obtient ce système en injectant (2.4) dans (2.5).

[15] Dans notre cas elles le sont.

L'auteur s'est limité à des partitions respectant les axes de symétrie du cœur[16] pour que l'initialisation de l'algorithme de Jacobi par $J_\Gamma = 0$ soit bonne. Des cas 2D avec 2 et 4 sous-domaines ont été testés. Les résultats obtenus en séquentiel ne permettaient pas d'envisager un algorithme parallèle plus rapide que le code séquentiel suite à une augmentation du nombre d'itérations.

2.2.2 Parallélisation sans modification algorithmique

Dans [7], la parallélisation du solveur Minos pour les équations de la diffusion est étudiée[17]. On utilise le fait que les matrices W_d sont diagonales par bloc, chaque bloc correspondant à une ligne de courant. Les degrés de liberté de courant sont donc distribués par groupes de lignes sur les différents processeurs (cf. figure FIG. 2.3). Les produits par les matrices B_d impliquent des communications de données homogènes à des flux[18](cf. figure FIG. 2.4). Une étude précise du coût des communications, pour plusieurs types de distribution 3D, conduit au même constat : la méthode ne peut être efficace que pour un faible nombre de processeurs. La méthode étant purement algébrique, elle ne permet pas d'adopter des tailles de maillage ou des degrés d'approximation différents selon les régions du cœur.

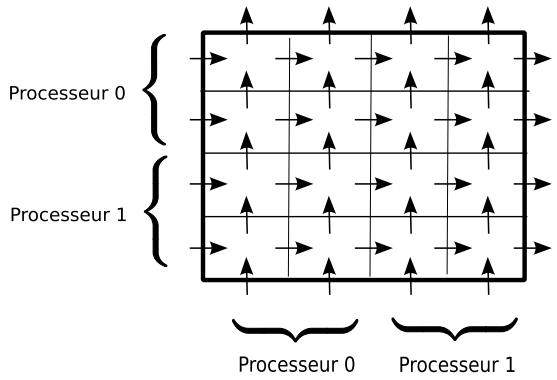


FIG. 2.3 – Distribution des degrés de liberté de courant dans chacune des directions

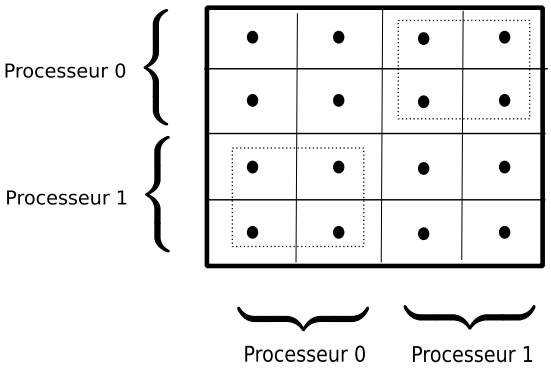


FIG. 2.4 – Schéma de communication : les données impliquant des communications sont encadrées

2.2.3 Méthode de synthèse modale

La méthode de synthèse modale consiste à choisir comme espace d'approximation l'espace engendré par les modes propres locaux de l'opérateur L. On note u_k^i le i-ème mode propre de L sur Ω_k :

$$\begin{cases} Lu_k^i &= \lambda_k^i u_k^i & \text{sur } \Omega_k \\ u_k &= 0 & \text{sur } \partial\Omega_k \end{cases} .$$

On note $\tilde{u}_k^i$ le prolongement de u_k^i par la fonction nulle sur Ω. On choisit N_k modes propres dans chaque sous-domaine Ω_k afin de définir l'espace de dimension finie V_δ dans lequel on discrétise

[16] Et avec des sections *presque* symétriques
[17] Ce travail peut s'appliquer directement aux équations SPn.
[18] Il s'agit du vecteur de fuite.

le problème modal global

$$V_\delta = Vect\left(\bigcup_k \bigcup_{1\leq i\leq N_k} \{\tilde{u}_k^i\}\right).$$

Si la partition n'admet aucun recouvrement, la solution obtenue est nulle aux interfaces suite aux calculs des modes propres avec des conditions de flux nul aux bords. Pour résoudre ce problème, deux solutions ont été mises en œuvre :

- dans [7], on ajoute des modes d'interface à l'espace d'approximation ;
- dans [9, 41], on utilise une partition avec recouvrement. On constate que le choix du partitionnement a un impact sur la qualité de la solution.

Dans cette méthode, la décomposition de domaine se place au dessus de la boucle de la puissance inverse. L'algorithme se décompose en 3 étapes :

1. on calcule en parallèle les modes propres dans chaque sous-domaine Ω_k. Cette partie nécessite une modification de l'algorithme de la puissance inverse afin de calculer plusieurs modes propres[19] ;
2. on discrétise le problème aux valeurs propres global dans l'espace V_δ construit. Le problème ainsi obtenu est de petite taille et se résout donc sur un seul processeur ;
3. on reconstruit la solution dans chaque sous-domaine Ω_k.

Dans la pratique, ces méthodes s'avèrent coûteuses et ont finalement été abandonnées au profit de méthodes de décomposition de domaine se plaçant sous l'algorithme de la puissance inverse. Plus simplement, on se ramène préférentiellement à la résolution d'un système linéaire plutôt que d'un problème aux valeurs propres et ce par décomposition de domaine.

2.2.4 Méthode de Schwarz avec conditions de Robin

Dans [9, 42], est étudiée une méthode de décomposition de domaine sans recouvrement pour les équations de la diffusion basée sur l'algorithme de Schwarz additif avec des conditions de transmission au niveau des interfaces de type Robin. La décomposition de domaine est jointe à la boucle de la puissance en bloquant à 1 le nombre d'itérations de l'algorithme de décomposition de domaine. A l'itération n de la puissance, on note ϕ_i^n et $\overrightarrow{J_i^n}$ le flux et le courant dans chaque sous-domaine Ω_i. On doit résoudre dans chaque sous-domaine Ω_i le problème suivant :

Problème 7
Trouver $(\phi_i^n, \vec{J_i^n}) \in L^2(\Omega_i) \times H(div, \Omega_i)$ *tel que*

$$\left\{\begin{array}{rcrcll} div\vec{J_i^n} & + & \Sigma_i\phi_i^n & = & f_i^n & \text{dans } \Omega_i \\ \dfrac{\vec{J_i^n}}{D_i} & + & \vec{\nabla}\phi_i^n & = & \vec{0} & \text{dans } \Omega_i \\ -\overrightarrow{J_i^n}.\overrightarrow{n_i} & + & \alpha_{i,j}\phi_i^n & = & \overrightarrow{J_j^{n-1}}.\vec{n_j} + \alpha_{i,j}\phi_j^{n-1} & \text{sur } \Gamma_{i,j} \quad \forall j \text{ voisin de } i \\ & & \phi_i^n & = & 0 & \text{sur } \partial\Omega_i \cap \partial\Omega \end{array}\right. ,$$

avec Σ_i, $D_i \in C^0(\Omega_i)$ *bornées et positives et* f_i^n *dans* $L^2(\Omega_i)$.

[19] Une deuxième méthode développée dans [9] permet de remplacer les modes non fondamentaux par des fonctions dépendant du mode propre fondamental et donc de s'affranchir de ce problème.

Le principal problème de la méthode réside dans le choix des paramètres $\alpha_{i,j}$. Pour l'instant, c'est à l'utilisateur de tâtonner pour trouver une bonne valeur de ces paramètres. Une autre difficulté survient lors de la discrétisation. En effet, on ne peut pas considérer de valeurs ponctuelles pour le flux aux interfaces. Dans [9], pour l'élément fini RT0, le choix a été fait de faire un recouvrement d'une seule maille et d'approximer le flux sur chacune des deux interfaces par la valeur du flux au sein de la maille.

Les résultats obtenus par cette méthode sont très bons : on observe que l'ajout de la décomposition de domaine influe peu sur le nombre d'itérations de la boucle de la puissance et la masse de calcul ajoutée par rapport à la version séquentielle est très faible (le recouvrement d'une seule maille est négligeable). On obtient même dans certains cas une efficacité parallèle supérieure à 100%. La méthode ne permet pas encore de traiter des maillages non-coïncidents. Elle peut toutefois être adaptée en ajoutant des matrices de projection. Malgré son principal inconvénient, qu'est le choix des paramètres $\alpha_{i,j}$ pour chaque groupe d'énergie, cette méthode est à ce jour la plus convaincante en vue d'une application industrielle.

2.3 Parallélisation du solveur SPn

2.3.1 Méthode de type Schur en neutronique

Une méthode de décomposition de domaine pour les équations multigroupes de la diffusion est mise en œuvre dans [43]. Chaque système monogroupe est résolu par une méthode de Schur primal. Le système d'interface est résolu par un algorithme de Gradient Conjugué muni d'un préconditionneur de type Neumann-Neumann. Les résultats sont pénalisés suite à un nombre important d'itérations de l'algorithme de Gradient Conjugué et des solveurs locaux. En effet les solveurs de Krylov sont très sensibles aux erreurs commises lors des premières itérations et les algorithmes itératifs internes doivent donc être poussés à convergence. L'algorithme parallélisé sur 5 processeurs est plus lent que le code séquentiel. Dans [44] l'auteur introduit des matrices de projection pour appliquer la méthode à des maillages avec des raffinements locaux.

2.3.2 Méthodes de Krylov

Les algorithmes de Krylov ne nécessitent que de savoir faire le produit par la matrice que l'on cherche à inverser. Un produit matriciel se parallélisant facilement, plusieurs auteurs ont essayé d'appliquer ces algorithmes.

2.3.2.1 Utilisation de l'algorithme du Gradient Conjugué préconditionné

Dans [45] est présentée une méthode de décomposition de domaine basée sur le découpage des matrices W. Au lieu d'appliquer un algorithme de Gauss-Seidel par bloc avec une factorisation de Cholesky sur chacun des trois blocs diagonaux W_d, l'algorithme de Gradient Conjugué est appliqué sur la matrice W. Ce système est préconditionné dans chaque sous-domaine par la matrice bloc-diagonale. Pour un faible nombre de sous-domaines, la méthode permet d'obtenir un solveur parallèle plus rapide que le solveur séquentiel. L'exécution séquentielle avec 27 sous-domaines est deux fois plus lente que le solveur mono-domaine. Avec 5 processeurs, le temps est réduit par deux. Avec plus de sous-domaines, le préconditionneur local se détériore et le nombre d'itérations pénalise la méthode.

2.3.2.2 Utilisation de l'algorithme GMRES préconditionné

Dans [46] une méthode parallèle basée sur l'utilisation de l'algorithme GMRES préconditionné [47] est présentée. Celui-ci est appliqué à la matrice multigroupe non symétrique. Le préconditionneur P utilisé est bloc diagonal et défini pour chaque groupe d'énergie :

$$P^{-1}_{g=g'} = \begin{pmatrix} A_x^{g=g'} & & & \\ & A_y^{g=g'} & & \\ & & A_z^{g=g'} & \\ & & & T^{g=g'} \end{pmatrix}.$$

Pour gérer les itérations imbriquées, un système de gestion de précision dynamique est utilisé. La taille de l'espace de Krylov est initialement fixée à 3 et ne peut excéder 9. Lorsque le résidu de l'algorithme de puissance augmente entre deux itérations, la taille de l'espace de Krylov est augmentée.

Les expérimentations numériques ont mis en évidence que le surcoût en nombre d'itérations ne permet pas d'obtenir un solveur parallèle efficace. Sur le cas test BenchMarkAIEA, l'exécution parallèle à 8 processeurs est 4.6 fois plus lente que l'exécution séquentielle du solveur basé sur les algorithmes de Gauss-Seidel.

2.4 Motivation de la thèse

Pour des machines à mémoire distribuée, la méthode de Schwarz avec conditions de Robin aux interfaces est la plus convaincante. Cependant, en vue d'une application industrielle, elle souffre de plusieurs limitations :

- le recouvrement d'une maille rend plus complexe l'intégration de la méthode dans un flux de données parallèles ;
- elle ne gère pas les maillages coïncidents ;
- dans le cas de maillages coïncidents, une nouvelle approximation est faite ce qui rend le problème discret multi-domaine non-équivalent au problème discret mono-domaine ;
- actuellement le seul moyen pour déterminer les paramètres α des conditions de Robin est la méthode *essai/erreur*.

L'objectif de cette thèse est de développer une méthode permettant de pallier à ces problèmes. Pour ce faire, on est prêt à perdre légèrement en performance. Notre choix s'est porté sur une méthode de type Schur Dual qui est décrite dans le chapitre suivant.

Chapitre 3

Une méthode de décomposition de domaine de type Schur dual

Sommaire

Suite aux limites des méthodes parallèles précédemment développées, on propose d'étudier une méthode sans recouvrement de type Schur dual [48] car elle a les avantages suivants :

- il s'agit d'une méthode classique pour traiter des maillages non-coïncidents ;
- dans le cas de maillages coïncidents, la méthode aboutit à un système matriciel équivalent au solveur monodomaine ;
- elle permet de réutiliser le solveur séquentiel existant sans introduire un nouveau de type de conditions aux limites ;
- une méthode sans recouvrement est plus simple à mettre en œuvre pour la gestion des données. Il s'agit d'un critère important en vue d'une intégration dans le code industriel ;
- elle n'introduit pas de nouveaux paramètres tel que le paramètre α des conditions de Robin dans le cas des méthodes de Schwarz. Dans la pratique, on est amené à ajouter de nouveaux paramètres concernant les critères de convergence de la résolution itérative du problème d'interface. Utilisant comme critère un nombre maximum d'itérations, il s'agit d'un paramètre entier, *a priori*, beaucoup plus simple à déterminer.

Le solveur monodomaine séquentiel est efficace car l'algorithme de la puissance permet d'utiliser un solveur linéaire très dégradé[20]. Pour être efficace, la méthode proposée doit se baser sur des solveurs locaux également très dégradés : ce point constitue une difficulté importante pour la mise en œuvre d'une méthode de décomposition de domaine efficace. Les gains en temps de calcul passent par une implémentation parallèle performante, car le solveur monodomaine séquentiel est de complexité linéaire avec la taille du problème traité.

Dans un premier temps, on introduit la méthode de type Schur dual pour les équations de diffusion pour deux sous-domaines. Le nom de la méthode étant sujet à discussion, un paragraphe nommé *classification* donne quelques éléments de justification de ce nom. Puis on généralise ces équations pour un nombre quelconque de sous-domaines. Enfin on discute des différents placements possibles de l'algorithme de résolution au sein du système d'itérations imbriquées qui est caractéristique de l'algorithme global.

3.1 Introduction : cas avec deux sous-domaines

On rappelle que l'on cherche à résoudre le problème de diffusion suivant :

Problème 8
Trouver $(\phi, \vec{J}) \in L^2(\Omega) \times H(div, \Omega)$ *tel que :*

$$\begin{cases} div(\vec{J}) + \Sigma\phi = f & dans\ \Omega \\ \dfrac{\vec{J}}{D} + \vec{\nabla}\phi = \vec{0} & dans\ \Omega \\ \phi = 0 & sur\ \partial\Omega \end{cases}$$

avec $\Sigma,\ D \in C^0(\Omega)$ *bornées telles que* $\Sigma(x) > \alpha,\ D(x) > \beta, \quad \alpha, \beta > 0$.

On découpe Ω en deux sous-domaines disjoints notés Ω_1 et Ω_2. On note Γ l'interface entre les deux sous-domaines et $\overrightarrow{n}$ le vecteur normal à Γ allant de Ω_1 à Ω_2. L'objectif de la méthode de décomposition de domaine est de résoudre le **Problème 8** en résolvant des problèmes locaux dans chaque sous-domaine Ω_i. La continuité des courants et des flux aux interfaces conduisent au problème équivalent :

[20] Algorithmes de Gauss-Seidel imbriqués bloqués à une itération.

Problème 9
Trouver $(\phi_1, \vec{J}_1) \in L^2(\Omega_1) \times H(div, \Omega_1)$, $(\phi_2, \vec{J}_2) \in L^2(\Omega_2) \times H(div, \Omega_2)$ *tels que :*

$$\left\{\begin{array}{rcll} div(\vec{J}_1) + \Sigma\phi_1 &=& f & \text{dans } \Omega_1 \\ \dfrac{\vec{J}_1}{D} + \vec{\nabla}\phi_1 &=& \vec{0} & \text{dans } \Omega_1 \\ \\ \phi_1 &=& \phi_\Gamma & \text{sur } \Gamma \\ \phi_1 &=& 0 & \text{sur } \partial\Omega_1 \cap \partial\Omega \end{array}\right. \quad et \quad \left\{\begin{array}{rcll} div(\vec{J}_2) + \Sigma\phi_2 &=& f & \text{dans } \Omega_2 \\ \dfrac{\vec{J}_2}{D} + \vec{\nabla}\phi_2 &=& \vec{0} & \text{dans } \Omega_2 \\ \\ \phi_2 &=& \phi_\Gamma & \text{dans } \Gamma \\ \phi_2 &=& 0 & \text{sur } \partial\Omega_2 \cap \partial\Omega \end{array}\right. \quad avec$$

$$\vec{J}_1 \cdot \vec{n} = -\vec{J}_2 \cdot \vec{n}.$$

3.1.1 Formulation variationnelle

Pour obtenir la formulation variationnelle, on introduit un multiplicateur[21] de Lagrange Λ. Le problème variationnel s'écrit donc :

Problème 10
Trouver $(\phi_1, \vec{J}_1) \in L^2(\Omega_1) \times H(div, \Omega_1)$, $(\phi_2, \vec{J}_2) \in L^2(\Omega_2) \times H(div, \Omega_2)$ *et* $\Lambda \in H^{1/2}(\Gamma)$ *tels que*

$$\left\{\begin{array}{lcll} \displaystyle\int_{\Omega_1} div(\vec{J}_1)\, v_1 \, d\vec{r} + \int_{\Omega_1} \Sigma_1\phi_1 v_1 \, d\vec{r} &=& \displaystyle\int_{\Omega_1} f_1 v_1 \, d\vec{r} & \forall\, v_1 \in L^2(\Omega_1) \\ \displaystyle\int_{\Omega_1} \frac{\vec{J}_1}{D_1} \cdot \vec{w}_1 \, d\vec{r} - \int_{\Omega_1} \phi_1\, div(\vec{w}_1)\, d\vec{r} + \int_\Gamma \Lambda \vec{w}_1 \cdot \vec{n} \, d\Gamma = 0 & & & \forall\, \vec{w}_1 \in H(div, \Omega_1) \end{array}\right.$$

$$\left\{\begin{array}{lcll} \displaystyle\int_{\Omega_2} div(\vec{J}_2)\, v_2 \, d\vec{r} + \int_{\Omega_2} \Sigma_2\phi_2 v_2 \, d\vec{r} &=& \displaystyle\int_{\Omega_2} f_2 v_2 \, d\vec{r} & \forall\, v_2 \in L^2(\Omega_2) \\ \displaystyle\int_{\Omega_2} \frac{\vec{J}_2}{D_2} \cdot \vec{w}_2 \, d\vec{r} - \int_{\Omega_2} \phi_2\, div(\vec{w}_2)\, d\vec{r} - \int_\Gamma \Lambda \vec{w}_2 \cdot \vec{n} \, d\Gamma = 0 & & & \forall\, \vec{w}_2 \in H(div, \Omega_2) \end{array}\right.$$

$$avec \int_\Gamma \left(\vec{J}_1 - \vec{J}_2\right) \cdot \vec{n}\, \mu \, d\Gamma = 0 \quad \forall\, \mu \in H^{1/2}(\Gamma).$$

La formulation variationnelle obtenue est composée de cinq équations :
- deux équations correspondant à la formulation variationnelle dans le sous-domaine Ω_1 et prenant en compte des conditions limites de type flux à l'interface ;
- deux équations pour le sous-domaine Ω_2 ;
- une équation qui permet d'assurer faiblement la continuité de la composante normale du courant sur l'interface Γ.

Dans chacun des sous-domaines ($i \in \{1, 2\}$), on considère un maillage cartésien dont les mailles sont notées K_n^i. Le **Problème 10** devient après discrétisation :

[21] Ce multiplicateur correspond au flux à l'interface entre les sous-domaines.

Problème 11
Trouver $(\phi_1, \vec{J_1}) \in V_h^1 \times W_h^1$, $(\phi_2, \vec{J_2}) \in V_h^2 \times W_h^2$ *et* $\Lambda_h \in V_h^\Gamma$ *tels que*

$$\left\{\begin{array}{rcll} a_1(\overrightarrow{J_1}, \overrightarrow{w_1}) - b_1(\phi_1, \overrightarrow{w_1}) & = & -c_1(\Lambda, \overrightarrow{w_1}) & \forall \overrightarrow{w_1} \in W_h^1 \\ b_1(v_1, \overrightarrow{J_1}) + t_1(\phi_1, v_1) & = & s_1(v_1) & \forall v_1 \in V_h^1 \\ & & & \\ a_2(\overrightarrow{J_2}, \overrightarrow{w_2}) - b_2(\phi_2, \overrightarrow{w_2}) & = & -c_2(\Lambda_h, \overrightarrow{w_2}) & \forall \overrightarrow{w_2} \in W_h^2 \\ b_2(v_2, \overrightarrow{J_2}) + t_2(\phi_2, v_2) & = & s_2(v_2) & \forall v_2 \in V_h^2 \\ & & & \\ c_1(\mu, \overrightarrow{J_1}) + c_2(\mu, \overrightarrow{J_2}) & = & 0 & \forall \mu \in V_h^\Gamma \end{array}\right.$$

avec

$$\left|\begin{array}{l} a_i(\overrightarrow{J_i}, \overrightarrow{w_i}) = \displaystyle\sum_n \frac{1}{D_n^i} \int_{K_n^i} \overrightarrow{J_i}.\overrightarrow{w_i} d\vec{r} \\ b_i(\phi_i, \overrightarrow{w_i}) = \displaystyle\int_{\Omega_i} \phi_i div\left(\overrightarrow{w_i}\right) \, d\vec{r} \\ t_i(\phi_i, v_i) = \displaystyle\sum_n \Sigma_n^i \int_{K_n^i} \phi_i v_i d\vec{r} \\ s_i(v_i) = \displaystyle\int_{\Omega_i} f_i v_i d\vec{r} \\ c_i(\Lambda_h, \overrightarrow{w_i}) = (-1)^{i+1} \displaystyle\int_\Gamma \Lambda_h \vec{w}_i \cdot \vec{n} \, d\Gamma. \end{array}\right.$$

Les espaces V_h^i et W_h^i sont définis comme les espaces V_h et W_h (cf. page 14) sur chaque sous-domaine Ω_i. Le choix de l'espace V_h^Γ est plus délicat. On choisit la trace de l'espace W_h^i ayant la discrétisation la plus fine, ce choix correspondant à l'une des deux possibilités de la méthode mortar [49, 50]. Le choix des fonctions de base de V_k^Γ n'intervient que dans l'écriture des matrices de couplage (cf. annexe C.2).

3.1.2 Formulation matricielle

On note :
- A_D^i la matrice associée à la forme bilinéaire a_i ;
- B_D^i la matrice associée à la forme bilinéaire b_i ;
- T_i la matrice associée à la forme bilinéaire t_i ;
- C_D^i la matrice associée à la forme bilinéaire c_i ;
- S_i le vecteur associé à la forme linéaire s_i ;
- J_D^i le vecteur contenant les degrés de liberté de courant $\overrightarrow{J_i}$;
- ϕ_i le vecteur contenant les degrés de liberté de flux ϕ_i.

Le **Problème 11** conduit alors au système matriciel suivant :

$$\left(\begin{array}{cc|cc|c} A_D^1 & -B_D^1 & & & C_D^1 \\ {}^tB_D^1 & T_1 & & & \\ \hline & & A_D^2 & -B_D^2 & C_D^2 \\ & & {}^tB_D^2 & T_2 & \\ \hline {}^tC_D^1 & & {}^tC_D^2 & & \end{array}\right) \left(\begin{array}{c} J_D^1 \\ \phi_1 \\ \hline J_D^2 \\ \phi_2 \\ \hline \Lambda \end{array}\right) = \left(\begin{array}{c} 0 \\ S_1 \\ \hline 0 \\ S_2 \\ \hline 0 \end{array}\right).$$

Dans le cas de maillages coïncidents, la méthode de décomposition de domaine est équivalente d'un point de vue matriciel au problème monodomaine. En 2D, le domaine (Fig. 3.1) auquel on associe le maillage 5×3 et l'élément fini RT0 est partitionné en deux sous-domaines suivant l'axe x (Fig. 3.2). La figure Fig. 3.3 permet de comparer le profil de la matrice monodomaine et celui de la matrice multidomaine. Les termes de couplage entre degrés de liberté du premier (respectivement second) sous-domaine sont colorés en bleu (respectivement rouge). Les termes en noir correspondent au couplage entre les deux sous-domaines.

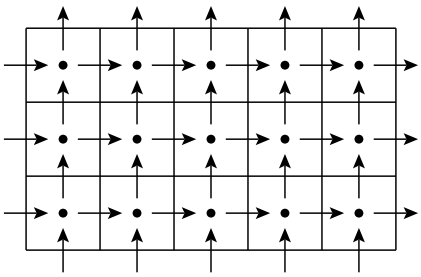


Fig. 3.1 – Maillage conforme de taille 5×3 en RT0

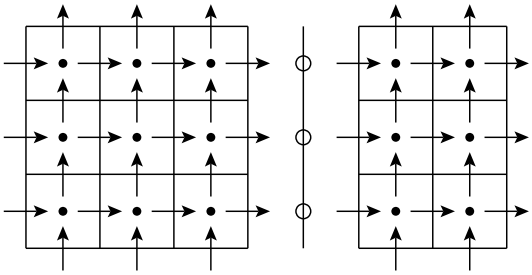


Fig. 3.2 – Décomposition en deux sous-domaines avec maillages coïncidents

De la même manière, dans le cas de maillages non coïncidents (Fig. 3.4(a)) avec le découpage en sous-domaines illustré avec la figure Fig. 3.4(b), on obtient la matrice multidomaine dont le profil est donné par la figure Fig. 3.4(c).

3.1.3 Classification de la méthode proposée

Il est difficile de donner un nom à la méthode au vue de la bibliographie existante. Celui-ci dépendra de l'axe selon lequel on la regarde.

▷ Si l'on met en avant la gestion de maillages non-coïncidents et le choix des fonctions de base du multiplicateur, la méthode peut être qualifiée de *méthode Mortar* [49, 50].

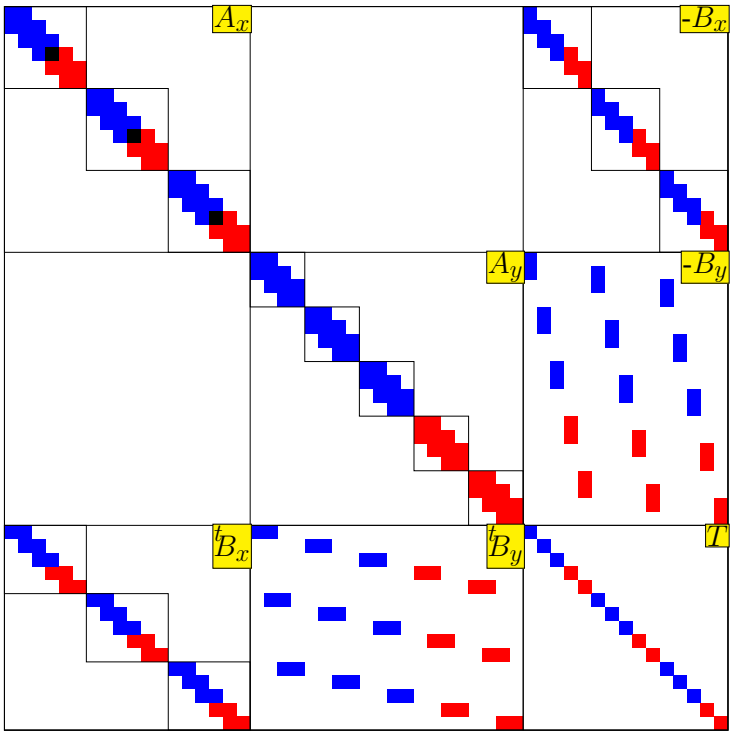
▷ Le fait que l'on applique une méthode de décomposition de domaine sur une formulation mixte [51] impose quelques spécificités. Les premières méthodes de décomposition de domaine pour des équations en formulation mixte ont été développées dans [52]. Ces méthodes se basent sur une hybridation partielle. L'hybridation est dite partielle car les sous-domaines contiennent plusieurs mailles. Cette méthode est largement utilisée en mécanique des fluides [52, 53, 54, 55, 56, 57]. La méthode d'hybridation est souvent utilisée de manière complète, car elle permet de se ramener à un système condensé à l'interface qui est symétrique défini positif [58, 59] et pouvant être résolu via une méthode de décomposition de domaine [60]. En neutronique, la méthode mixte-hybride pour les équations de la diffusion a été introduite dans [13] puis dans [61, Chapitre 4] et enfin détaillée dans [62]. Dans [8, p. 99], elle est utilisée pour relâcher la contrainte de continuité forte sur les courants et pouvoir ainsi construire aisément une base pour les courants pour des éléments de référence trapézoïdaux.

Pour ces mêmes équations, sans lien direct avec la neutronique, d'autres méthodes de décomposition de domaine ont été mises en œuvre [63, 64]. On peut donc appeler la méthode proposée *méthode de décomposition de domaine basée sur la formulation mixte-hybride partielle.*

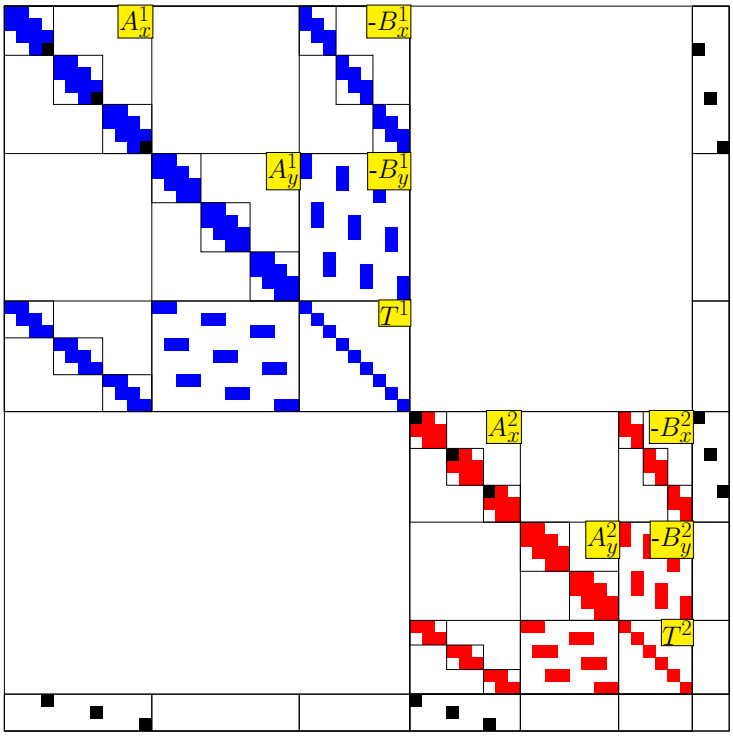
▷ Du fait de la formulation mixte, il est difficile de voir au niveau continu la méthode comme

une méthode de Schur primal ou Schur dual. Au niveau matriciel, du fait qu'il n'y ait sur les interfaces que des degrés de liberté de courant et qu'ils soient dupliqués, permet de qualifier la méthode de *Schur dual*.

▷ A la méthode de Schur dual, on associe naturellement la méthode FETI. La méthode présentée ne sera pas appelée méthode FETI car avec les équations de la diffusion et les éléments finis de Raviart-Thomas, on évite les problèmes liés à l'introduction de pseudo-inverses et de *cross-points* (degrés de liberté appartenant à plus de deux sous-domaines).

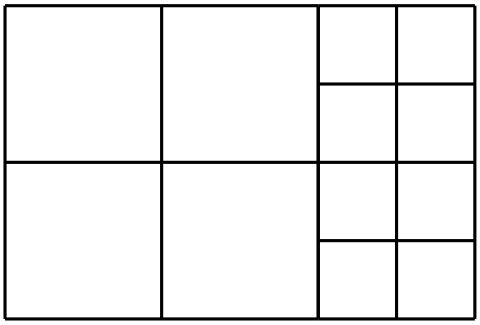


(a) Profil de la matrice associée au problème monodomaine

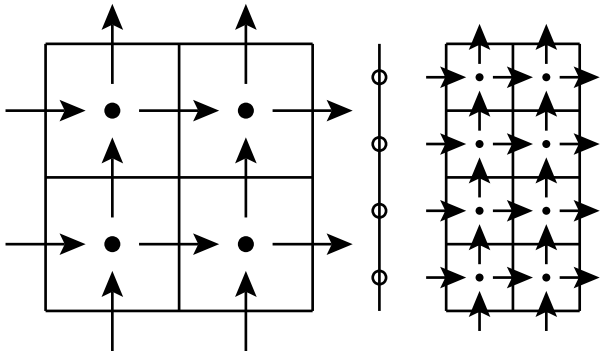


(b) Profil de la matrice associée au problème à deux sous-domaines

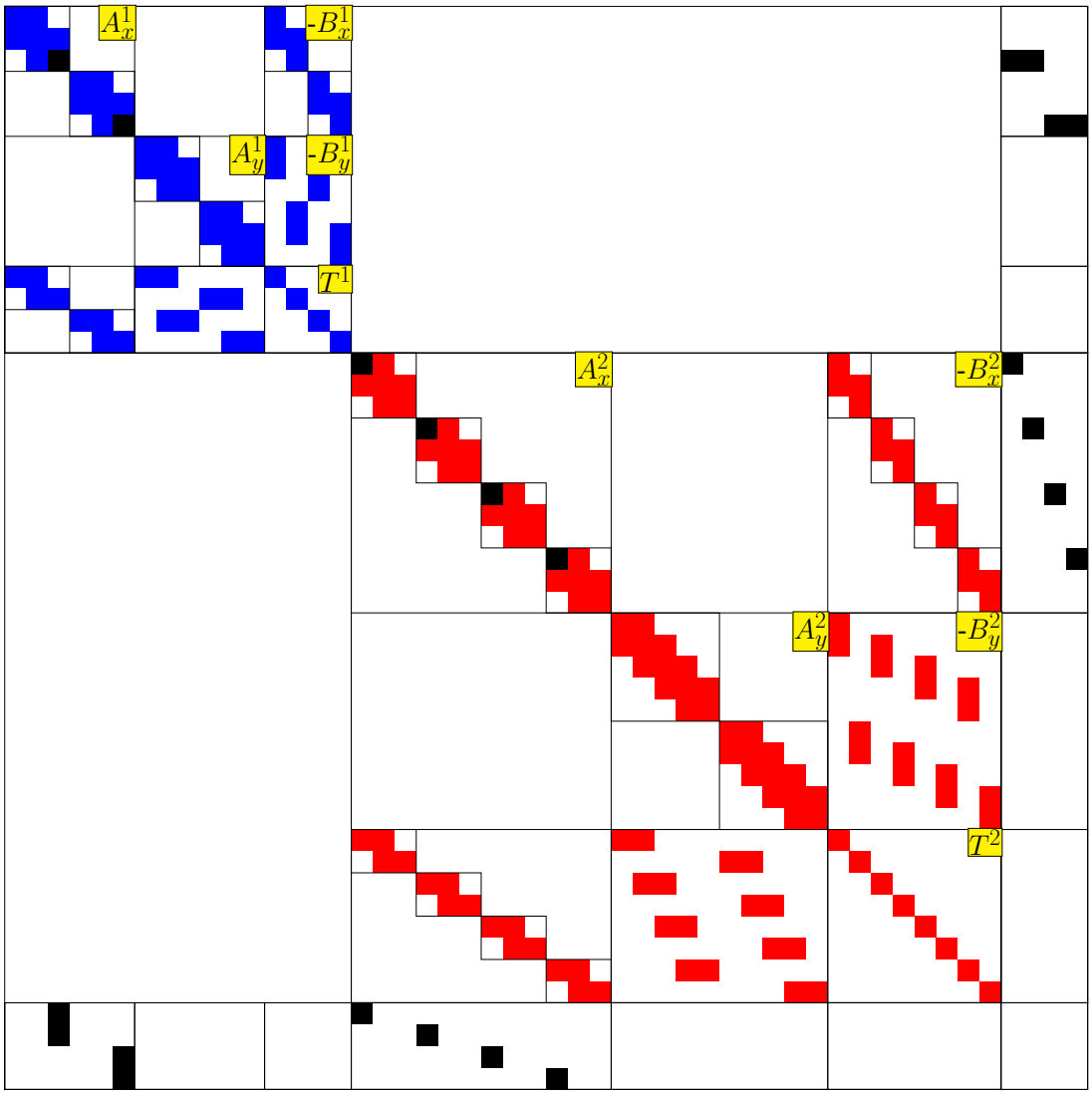
FIG. 3.3 – Comparaison entre les matrices monodomaine et multidomaine



(a) Maillage non-conforme.



(b) Décomposition en deux sous-domaines avec maillages non-coïncidents



(c) Profil de la matrice

FIG. 3.4 – Problème à deux sous-domaines avec maillages non-coïncidents

3.2 Cas général multidomaine

On considère que Ω est décomposé en K sous-domaines $\left(\Omega_k\right)_{1\leq k\leq K}$ ne se recouvrant pas et on note :

- $\mathfrak{N}_k$ l'ensemble des sous-domaines voisins de Ω_k ;
- $\tilde{\mathfrak{N}}_k$ l'ensemble des sous-domaines voisins de Ω_k d'indice supérieur à k ;
- $\mathfrak{N}$ le sous-ensemble de $[\![1,K]\!]\times[\![1,K]\!]$ défini par $\bigcup\limits_{1\leq k\leq K}\bigcup\limits_{l\in\mathfrak{N}_k}\{(k,l)\}$;
- $\tilde{\mathfrak{N}}$ le sous-ensemble de $[\![1,K]\!]\times[\![1,K]\!]$ défini par $\bigcup\limits_{1\leq k\leq K}\bigcup\limits_{l\in\tilde{\mathfrak{N}}_k}\{(k,l)\}$;
- $(\Gamma_{k,l})_{(k,l)\in\tilde{\mathfrak{N}}}$ les interfaces entre les sous-domaines Ω_k et Ω_l, $l>k$;
- $(\vec{n}_{k,l})_{(k,l)\in\tilde{\mathfrak{N}}}$ les normales de $\Gamma_{k,l}$ sortantes pour le domaine Ω_k ;
- $(\vec{\tilde{n}}_{k,l})_{(k,l)\in\tilde{\mathfrak{N}}}$ les vecteurs définis par : $\forall(k,l)\in\tilde{\mathfrak{N}},\quad \vec{\tilde{n}}_{k,l}=\vec{n}_{min(k,l),max(k,l)}$;
- $\epsilon_{k,l}$ le signe de $(k-l)$ avec $\epsilon_{k,k}=0$.

Pour résoudre le **Problème 8**, on exprime le problème dans chaque sous-domaine Ω_k en modifiant les conditions limites pour assurer la continuité des courants et des flux aux interfaces :

Trouver $(\phi_k,\vec{J}_k)\in L^2(\Omega_k)\times H(div,\Omega_k)$ tel que

$$\begin{cases} div\left(\vec{J}_k\right) + \Sigma_k\phi_k = f_k & \text{dans } \Omega_k \\ \dfrac{\vec{J}_k}{D_k} + \vec{\nabla}\phi_k = \vec{0} & \text{dans } \Omega_k \\ \phi_k = 0 & \text{sur } \partial\Omega_k\cap\partial\Omega \\ \forall l\in\tilde{\mathfrak{N}}_k \quad \vec{J}_k\cdot\vec{n}_{k,l} = -\vec{J}_l\cdot\vec{n}_{l,k} & \text{sur } \Gamma_{k,l} \\ \forall l\in\tilde{\mathfrak{N}}_k \quad \phi_k = \phi_l & \text{sur } \Gamma_{k,l} \end{cases} \tag{3.1}$$

où $\Sigma_k=\Sigma_{|\Omega_k}$, $\quad D_k=D_{|\Omega_k}$, $\quad f_k=f_{|\Omega_k}$ et $\vec{n}_{k,l}=-\vec{n}_{l,k}$.

On peut montrer simplement que $\overrightarrow{J_k}=\overrightarrow{J}_{|\Omega_k}$ et $\phi_k=\phi_{|\Omega_k}$ avec (J,ϕ) solution du **Problème 8**.

3.2.1 Formulation variationnelle

(3.1) conduit à la formulation variationnelle multidomaine suivante[22] :

Pour tout $1\leq k\leq K$, pour tout $(k,l)\in\tilde{\mathfrak{N}}$, trouver $(\phi_k,\vec{J}_k)\in L^2(\Omega_k)\times H(div,\Omega_k)$ et $\Lambda_{k,l}\in H^{1/2}(\Gamma_{k,l})$ tels que

$$\begin{cases} \displaystyle\int_{\Omega_k} div\left(\vec{J}_k\right)v_k\,d\vec{r}+\int_{\Omega_k}\Sigma_k\phi_k v_k\,d\vec{r} = \int_{\Omega_k} f_k v_k\,d\vec{r} & \forall\, v_k\in L^2(\Omega_k) \\ \displaystyle\int_{\Omega_k}\frac{\vec{J}_k}{D_k}\cdot\vec{w}_k\,d\vec{r}-\int_{\Omega_k}\phi_k\,div\left(\vec{w}_k\right)d\vec{r}+\sum_{l\in\mathfrak{N}_k}\epsilon_{k,l}\int_{\Gamma_{k,l}}\Lambda_{k,l}\vec{w}_k\cdot\vec{\tilde{n}}_{k,l}\,d\Gamma=0 & \forall\,\vec{w}_k\in H(div,\Omega_k) \end{cases} \tag{3.2}$$

[22] Voir la partie 3 de [48].

et pour tout $(k,l) \in \tilde{\mathfrak{N}}$,

$$\int_{\Gamma_{k,l}} \left(\vec{J}_k - \vec{J}_l\right) \cdot \vec{n}_{k,l}\, \mu_{k,l}\, d\Gamma_{k,l} = 0 \quad \forall\, \mu_{k,l} \in V_{k,l}. \tag{3.3}$$

$(\Lambda_{k,l} \in V_{k,l})_{1\leq k<l\leq K}$ désignent les multiplicateurs de Lagrange associés à la continuité faible de la composante normale du courant sur $\Gamma_{k,l}$. $V_{k,l}$ est constitué des fonctions de $H^{1/2}(\Gamma_{k,l})$ qui s'annulent sur $\Gamma_{k,l} \cap \partial\Omega$. $\Lambda_{k,l}$ désigne dans la suite le vecteur des degrés de liberté associé au multiplicateur introduit sur $\Gamma_{k,l}$. Dans (3.2-3.3), $\Lambda_{k,l}$ désignera $\Lambda_{min(k,l),max(k,l)}$ par abus de notation.

3.2.2 Formulation matricielle

On introduit les notations suivantes :

- $J_{k,l}$ $(k \neq l)$ représente les degrés de liberté de courant associés à Ω_k situés sur l'interface $\Gamma_{k,l}$;
- les matrices éléments finis de couplage $C_{k\to l}$ (respectivement $C_{l\to k}$) entre $J_{k,l}$ (respectivement $J_{l,k}$) et $\Lambda_{k,l}$. On rappelle que
  - $J_{k,l}$ et $J_{l,k}$ ne désignent pas le même courant ;
  - $\Lambda_{k,l}$ et $\Lambda_{l,k}$ sont deux notations équivalentes.
- $q_{k,l}$ la restriction des degrés de liberté de courant de Ω_k sur ceux de $\Gamma_{k,l}$ et $Q_{k,l}$ la matrice associée ;
- ${}^t q_{k,l}$ l'indicatrice des degrés de liberté de $\Gamma_{k,l}$ dans ceux de courant de Ω_k et ${}^t Q_{k,l}$ la matrice associée ;
- $p_{k,l}$ la restriction des degrés de liberté de Γ sur ceux de $\Gamma_{k,l}$ et $P_{k,l}$ la matrice associée ;
- ${}^t p_{k,l}$ l'indicatrice des degrés de liberté de $\Gamma_{k,l}$ dans ceux de Γ et ${}^t P_{k,l}$ la matrice associée.

On réutilise les notations des matrices définies page 17 en ajoutant un indice k pour le sous-domaine Ω_k. La définition des matrices de couplage $C_{k\to l}$ est détaillée dans l'annexe C.2. Après discrétisation, (3.2-3.3) deviennent donc :

$$\forall\, 1 \leq k \leq K, \quad \begin{pmatrix} A_D^k & -B_D^k \\ {}^t B_D^k & T_k \end{pmatrix} \begin{pmatrix} J_D^k \\ \phi_k \end{pmatrix} = \begin{pmatrix} 0 \\ S_k \end{pmatrix} - \begin{pmatrix} \sum_{l\in\mathfrak{N}_k} \epsilon_{k,l}\, {}^t Q_{k,l} C_{k\to l} \Lambda_{k,l} \\ 0 \end{pmatrix} \tag{3.4}$$

et pour tout $(k,l) \in \tilde{\mathfrak{N}}$

$${}^t C_{k\to l} J_{k,l} - {}^t C_{l\to k} J_{l,k} = 0. \tag{3.5}$$

Afin de simplifier (3.4-3.5), on introduit les notations suivantes :

- $H_k = \begin{pmatrix} A_D^k & -B_D^k \\ {}^t B_D^k & T_k \end{pmatrix}$;
- $X_k = \begin{pmatrix} J_D^k \\ \phi_k \end{pmatrix}$;
- $L_k = \begin{pmatrix} 0 \\ S_k \end{pmatrix}$;

– $C_D^k = \sum_{l \in \mathfrak{N}_k} {}^t Q_{k,l} \epsilon_{k,l} C_{k \to l} P_{k,l}$;

– $C_k = \begin{pmatrix} C_D^k \\ 0 \end{pmatrix}$;

– $\Lambda = \sum_{(k,l) \in \tilde{\mathfrak{N}}} {}^t P_{k,l} \Lambda_{k,l}$.

Afin de visualiser les espaces de départ et d'arrivée associés aux nombreuses notations concernant les matrices de couplage, le lecteur peut se référer à l'annexe C.1.

Les équations (3.4-3.5) s'écrivent ainsi pour tout $1 \leq k \leq K$

$$H_k X_k = L_k - C_k \Lambda \tag{3.6}$$

et

$$\sum_k {}^t C_k X_k = 0 \tag{3.7}$$

car

$$\begin{aligned}
(3.5) &\iff \forall (k,l) \in \mathfrak{N}, \quad \epsilon_{k,l} {}^t C_{k \to l} Q_{k,l} J_k + \epsilon_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 && \text{car } J_{k,l} = Q_{k,l} J_k \\
&\iff \forall (k,l) \in \mathfrak{N}, \quad \epsilon_{k,l} {}^t P_{k,l} {}^t C_{k \to l} Q_{k,l} J_k + \epsilon_{l,k} {}^t P_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 && \text{car } {}^t P_{k,l} = {}^t P_{l,k} \\
&\iff \forall k \in [\![1, K]\!], \\
&\qquad \sum_{l \in \mathfrak{N}_k} \epsilon_{k,l} {}^t P_{k,l} {}^t C_{k \to l} Q_{k,l} J_k + \sum_{l \in \mathfrak{N}_k} \epsilon_{l,k} {}^t P_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 && \text{en sommant sur } l \\
&\iff \forall k \in [\![1, K]\!], \quad {}^t C_D^k J_k + \sum_{l \in \mathfrak{N}_k} \epsilon_{l,k} {}^t P_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 \\
&\iff \sum_k {}^t C_D^k J_k + \sum_k \sum_{l \in \mathfrak{N}_k} \epsilon_{l,k} {}^t P_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 && \text{en sommant sur } k \\
&\iff \sum_k {}^t C_D^k J_k + \sum_l \sum_{k \in \mathfrak{N}_l} \epsilon_{l,k} {}^t P_{l,k} {}^t C_{l \to k} Q_{l,k} J_l = 0 \\
&\iff \sum_k {}^t C_D^k J_k + \sum_l {}^t C_l J_l = 0 \\
&\iff \sum_k {}^t C_D^k J_k = 0 \\
&\iff \sum_k {}^t C_k X_k = 0.
\end{aligned}$$

(3.6) et (3.7) conduisent au final au système matriciel suivant

$$\begin{pmatrix} H_1 & & & & & C_1 \\ & \ddots & & & & \vdots \\ & & H_k & & & C_k \\ & & & \ddots & & \vdots \\ & & & & H_K & C_K \\ {}^tC_1 & \dots & {}^tC_k & \dots & {}^tC_K & 0 \end{pmatrix} \begin{pmatrix} X_1 \\ \vdots \\ X_k \\ \vdots \\ X_K \\ \Lambda \end{pmatrix} = \begin{pmatrix} L_1 \\ \vdots \\ L_k \\ \vdots \\ L_K \\ 0 \end{pmatrix} \tag{3.8}$$

En notant

$$\hat{H} = \begin{pmatrix} H_1 & & & & \\ & \ddots & & & \\ & & H_k & & \\ & & & \ddots & \\ & & & & H_K \end{pmatrix}, \quad C = \begin{pmatrix} C_1 \\ \vdots \\ C_k \\ \vdots \\ C_K \end{pmatrix}, \quad \hat{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_k \\ \vdots \\ X_K \end{pmatrix}, \quad \hat{L} = \begin{pmatrix} L_1 \\ \vdots \\ L_k \\ \vdots \\ L_K \end{pmatrix},$$

on obtient alors le système de point-selle suivant

$$\begin{pmatrix} \hat{H} & C \\ {}^tC & 0 \end{pmatrix} \begin{pmatrix} \hat{X} \\ \Lambda \end{pmatrix} = \begin{pmatrix} \hat{L} \\ 0 \end{pmatrix}. \tag{3.9}$$

3.3 Les différents placements de la boucle de décomposition de domaine

Par souci de simplicité, la méthode de décomposition de domaine a été présentée au niveau spatial. Ceci correspond à un placement de la méthode de décomposition de domaine au niveau de la résolution de chaque bloc diagonal impliqué par l'algorithme de Gauss-Seidel sur les harmoniques. Toutefois il est possible de placer l'algorithme de décomposition de domaine à chaque niveau du solveur global caractérisé par un système d'itérations imbriquées. Ainsi on considère plusieurs algorithmes de décomposition de domaine basés sur

le solveur multigroupe : l'algorithme de décomposition de domaine est situé sous la boucle de la puissance. A chaque itération, le calcul de l'inverse de la matrice multigroupe H est fait par décomposition de domaine. Le solveur utilisé dans chaque sous-domaine est donc le solveur multigroupe. On parle par la suite d'approche multigroupe (cf. **Algorithme 8**).

le solveur monogroupe : l'algorithme de décomposition de domaine, est appliqué au problème monogroupe. Cela signifie que l'algorithme de décomposition de domaine se place entre l'algorithme de Gauss-Seidel sur les groupes et celui sur les harmoniques. On parle d'approche monogroupe.

le solveur de diffusion : l'algorithme de décomposition de domaine est appliqué au problème de diffusion (cf. **Algorithme 9**). On parle d'approche spatiale. Dans le cas des équations SP1, l'approche spatiale est équivalente à l'approche monogroupe.

le solveur unidimensionnel W : l'algorithme de décomposition de domaine est appliqué aux matrices W obtenues après élimination des flux (cf. **Algorithme 10**). On parle d'approche unidimensionnelle.

Afin de pouvoir choisir *a priori* la méthode la plus prometteuse, on liste les avantages et inconvénients de chacune de ces approches.

3.3.1 Propriétés du système d'interface

Dans ce paragraphe, on s'intéresse essentiellement aux propriétés de symétrie du système d'interface. Dans le cas de l'approche spatiale, la matrice d'interface ${}^tC\hat{H}^{-1}_{g=g'}C$ est symétrique définie positive.

Preuve : Comme la somme de matrices symétriques est symétrique, il suffit de prouver que $S_i = {}^tC_i(H_i^{g=g'})^{-1}C_i$ est symétrique. Cette matrice ne faisant intervenir qu'un groupe d'énergie et qu'un seul sous-domaine, on supprime les indices $g = g'$ et i dans la preuve qui suit. Comme il n'existe pas de degrés de liberté de flux sur l'interface, S peut s'écrire sous la forme :

$$S = ({}^tC_D \quad 0) \begin{pmatrix} A_D & -B_D \\ {}^tB_D & T \end{pmatrix}^{-1} \begin{pmatrix} C_D \\ 0 \end{pmatrix}$$

avec C_D de rang plein.

Soit ω un vecteur d'interface ; pour calculer $S\omega$ on résout le système suivant :

$$\begin{pmatrix} A_D & -B_D \\ {}^tB_D & T \end{pmatrix} \begin{pmatrix} J_D \\ \phi \end{pmatrix} = \begin{pmatrix} C_D\omega \\ 0 \end{pmatrix}.$$

En éliminant les flux (On pose $W = A_D + B_D T^{-1} {}^tB_D$), on obtient :

$$\begin{cases} J_D &= W^{-1}C_D\omega \\ \phi &= -T^{-1}{}^tB_D J_D \end{cases}$$

avec $W = A_D + B_D T^{-1} {}^tB_D$. On en déduit donc que pour chaque vecteur d'interface ω

$$S\omega = {}^tC_D W^{-1} C_D \omega$$

assurant ainsi l'égalité $S = {}^tC_D W^{-1} C_D$. La matrice W étant symétrique, on a prouvé la symétrie du système d'interface. Le caractère défini positif provient directement du caractère défini positif de la matrice $\hat{H}^{-1}_{g=g'}$ et du rang plein de C_D.

[CQFD]

En revanche il faut faire attention au fait que le solveur interne est basé sur un algorithme de Gauss-Seidel non convergé : les propriétés de symétrie sont donc perdues. Pour les conserver, il faudrait changer le solveur interne en utilisant par exemple un algorithme de Gauss-Seidel symétrisé.

Dans le cas de l'approche unidimensionnelle, les matrices des systèmes d'interface sont symétriques définies positives, de part le caractère symétrique et défini positif des matrices W_d. Dans le cas de l'approche monogroupe (SPn, $n \geq 3$), comme dans l'approche multigroupe, les systèmes d'interface ne sont pas symétriques suite à la non symétrie de la matrice globale.

3.3.2 Robustesse vis-à-vis de la dégradation des solveurs internes

A notre connaissance, il n'existe pas de résultat permettant de prédire le comportement d'un système d'itérations imbriquées. Pour le cas de l'approche unidimensionnelle, il est envisageable d'implémenter un solveur direct pour le système d'interface, car elle se base sur des solveurs eux même directs dans chaque sous-domaine. C'est donc la seule voie où l'on puisse garantir,

avant implémentation, que l'imbrication d'un nouveau solveur n'implique pas de problème de convergence.

Dans [65], l'auteur a développé une méthode basée sur un algorithme de Schwarz pour coupler différents types de discrétisation de la variable angulaire. Deux placements pour la méthode de décomposition de domaine étaient possibles ; la première au dessus des groupes, la seconde juste en dessous. Les résultats ont montré que la seconde méthode est plus efficace et que lorsque l'algorithme de Gauss-Seidel sur les groupes est bloqué à une itération, les résultats sont équivalents en termes de nombre d'itérations de puissance. A partir de ces résultats, il est difficile de tirer des conclusions quant au placement de la boucle de décomposition de domaine pour le problème que l'on cherche à résoudre.

3.3.3 Facilité d'implémentation et capacité d'évolution

Quelle que soit l'approche utilisée, la répartition des données est la suivante : toutes les matrices et tous les vecteurs d'un sous-domaine sont stockés sur le processus associé à ce sous-domaine. En effet, supposons que pour un sous-domaine deux matrices pour des groupes d'énergie différents soient stockées sur des processeurs différents. Cela impliquerait au minimum la communication d'un champ spatial complet pour prendre en compte les termes de down-scattering et d'up-scattering lors de la boucle sur les groupes d'énergie. En revanche, si l'on garantit que toutes les données d'un sous-domaine sont sur le même processeur, alors les communications ne se font que sur des champs d'interfaces, par nature de plus petites tailles.

On souhaite de plus qu'un processeur puisse gérer plusieurs sous-domaines pour
- équilibrer la charge avec des partitionnements plus complexes ;
- mettre en œuvre une parallélisation hybride (mémoires partagée et distribuée).

Dans ce contexte, la méthode la plus simple consiste à considérer pour chaque vecteur et pour chaque matrice impliqués dans l'algorithme de la puissance, un vecteur indicé par les sous-domaines locaux aux processeurs. Cette structure est également nécessaire si l'on souhaite une implémentation séquentielle de la méthode décomposition de domaine. Comme il est plus simple de placer les structures de données et les algorithmes au même niveau, l'approche multigroupe est plus simple à implémenter.

Une autre source de difficultés pour les autres approches provient du fait qu'il faille connaître le fonctionnement interne au solveur multigroupe et donc une meilleure connaissance du code existant.

Enfin seule l'approche multigroupe permettrait d'envisager des nombres de groupes différents par sous-domaines.

3.3.4 Granularité

Plus l'algorithme de décomposition de domaine est placé haut dans le système d'itérations imbriquées, plus la granularité des calculs sera élevée. A même volume de calculs et de communications, une granularité plus élevée permet de réduire le coût de latence des communications. En revanche, une granularité plus faible laisse généralement plus de possibilités pour ordonnancer les calculs et diminuer les problèmes d'équilibrage de charge. Dans [66, Chapitre 4], l'auteur analyse différentes stratégies en terme de granularité et d'équilibrage de charge. Dans notre contexte, les résultats de cette étude ne sont pas applicables car l'équilibrage de la charge est géré lors du partitionnement en sous-domaines ; les différentes approches ont donc un équilibrage équivalent. Si l'on considère la complexité du solveur comme parfaitement linéaire par rapport au nombre

Approche	Multigroupe	Monogroupe	Spatiale	Uni-dimensionnelle
Nombre d'itérations de puissance par rapport au cas séquentiel	*a priori* supérieur	*a priori* supérieur	*a priori* supérieur	égale*
Symétrie du système d'interface	non	non	oui**	oui
Granularité	élevée	moyenne +	moyenne −	faible
Coût par appel supplémentaire aux solveurs locaux	élevé +	élevé	élevé −	faible
Complexité d'implémentation et possibilités d'évolution	relativement simple et nombreuses possibilités d'évolution	complexe	complexe +	complexe et très spécifique

* en utilisant un solveur d'interface direct.
** en modifiant le solveur interne.

TAB. 3.1 – Récapitulatif des avantages - inconvénients des différentes approches

de degrés de liberté, on cherchera plutôt à maximiser la granularité. Hors, cette propriété de linéarité n'existe que si l'on néglige dans les algorithmes de Gauss-Seidel le produit par les termes extra-diagonaux. Si l'on considère que l'algorithme de décomposition de domaine fait plusieurs fois appel aux solveurs locaux, le fait de descendre le placement de la boucle de décomposition de domaine permet d'économiser des produits par les termes extra-diagonaux externes.

Comparons dans le cas SP1 l'approche multigroupe et l'approche spatiale. On considère deux appels aux solveurs locaux dans l'algorithme de résolution du système de point-selle : la première méthode implique un produit par la matrice de down-scattering et un par celle d'up-scattering en plus. Dans le cas séquentiel ce surcoût représente environ 6% d'une itération multigroupe[23]. Au sein du solveur de diffusion, les termes extra-diagonaux représentent une partie importante du solveur séquentiel : la partie diagonale (les descentes - remontées sur les matrices W factorisées) ne représente que 40% du temps d'exécution pour une itération du solveur de diffusion.

3.3.5 Choix

On récapitule dans le tableau TAB. 3.1 les propriétés des différentes approches. Le choix n'est pas facile, car on a très peu d'informations sur un critère très important concernant les propriétés de convergence. Dans un premier temps, on choisit la méthode la plus simple à implémenter et qui offre le plus de possibilités d'évolution. Le chapitre 4 est donc consacré à l'approche multigroupe. Suite aux difficultés rencontrées avec l'approche multigroupe, on étudie dans un deuxième temps l'approche unidimensionnelle dans le chapitre 5. Avec cette approche, on se focalisera plus sur la performance.

[23] Dans le cas SP3, ce surcoût est de l'ordre 20% (6% + 14%).

Algorithme 8 : Itérations emboîtées avec approche multigroupe

Algorithme des puissances inverses
- **Algorithme de décomposition de domaine**
  - **pour chaque** *sous-domaine* **faire**
    - **Gauss-Seidel par bloc**
      - **pour chaque** *groupe g* **faire**
        - **Gauss-Seidel par bloc**
          - **pour chaque** *hamonique l* **faire**
            - **Gauss-Seidel par bloc**
              - **pour chaque** *direction de l'espace* **faire**
                - Résolution par descente-remontée;
            - Calcul de ϕ_g ;

Algorithme 9 : Itérations emboîtées avec approche spatiale

Algorithme des puissances inverses
- **Gauss-Seidel par bloc**
  - **pour chaque** *groupe g* **faire**
    - **Gauss-Seidel par bloc**
      - **pour chaque** *harmonique l* **faire**
        - **Algorithme de décomposition de domaine**
          - **pour chaque** *sous-domaine* **faire**
            - **Gauss-Seidel par bloc**
              - **pour chaque** *direction de l'espace* **faire**
                - Résolution par descente-remontée;
            - Calcul de ϕ_g ;

Algorithme 10 : Itérations emboîtées avec approche unidimensionnelle

Algorithme des puissances inverses
- **Gauss-Seidel par bloc**
  - **pour chaque** *groupe g* **faire**
    - **Gauss-Seidel par bloc**
      - **pour chaque** *harmonique h* **faire**
        - **Gauss-Seidel par bloc**
          - **pour chaque** *direction de l'espace* **faire**
            - **Algorithme de décomposition de domaine**
              - **pour chaque** *sous-domaine* **faire**
                - Résolution par descente-remontée;
          - Calcul de ϕ_g ;

Chapitre 4

Approche multigroupe

Sommaire

Dans ce chapitre, on détaille l'approche multigroupe. Cette dernière a été choisie dans un premier temps car elle est la plus simple à développer et qu'elle permet *a priori* le plus d'évolutions possibles. On accepte *a priori* de ne pas obtenir une performance optimale afin d'obtenir une méthode simple et générique. On présente une généralisation de la méthode de décomposition de domaine pour les équations multigroupes, puis les algorithmes de type Uzawa permettant de résoudre le système de point-selle obtenu. Enfin on présente les bases de l'implémentation parallèle et les résultats numériques (publiés dans [67]) obtenus.

4.1 Obtention des équations multigroupes multidomaines

A chaque itération de l'algorithme de la puissance, on cherche à résoudre par décomposition de domaine le système linéaire multigroupe obtenu via le problème suivant

Problème 12
Trouver $(\phi^{g=1}, \vec{J}^{g=1}) \in L^2(\Omega) \times H(\mathrm{div}\,, \Omega)$ *et* $(\phi^{g=2}, \vec{J}^{g=2}) \in L^2(\Omega) \times H(\mathrm{div}\,, \Omega)$ *tels que*

$$
\begin{cases}
\mathrm{div}\left(\vec{J}^{g=1}\right) + \Sigma_a^{g=1}\phi^{g=1} - \Sigma_{s,0}^{2\to 1}\phi^{g=2} = s^{g=1} & \text{dans } \Omega \\
\dfrac{\vec{J}^{g=1}}{D^{g=1}} + \vec{\nabla}\phi^{g=1} - \Sigma_{s,1}^{2\to 1}\vec{J}^{g=2} = \vec{0} & \text{dans } \Omega \\
\phi^{g=1} = 0 & \text{sur } \partial\Omega \\
\mathrm{div}\left(\vec{J}^{g=2}\right) + \Sigma_a^{g=2}\phi^{g=2} - \Sigma_{s,0}^{1\to 2}\phi^{g=1} = s^{g=2} & \text{dans } \Omega \\
\dfrac{\vec{J}^{g=2}}{D^{g=2}} + \vec{\nabla}\phi^{g=2} - \Sigma_{s,1}^{1\to 2}\vec{J}^{g=1} = \vec{0} & \text{dans } \Omega \\
\phi^{g=2} = 0 & \text{sur } \partial\Omega
\end{cases}
$$

avec $D^{g=g'}$, $\Sigma_{s,0}^{g\to g'}$, $\Sigma_{s,1}^{g\to g'}$ *et* $\Sigma_a^{g=g'}$ *bornées et positives.*

L'introduction d'un multiplicateur par groupe d'énergie conduit à la formule variationnelle suivante

Problème 13
Pour tout $(k,l) \in \mathfrak{N}$ *trouver*

- $(\phi_k^{g=1}, \vec{J}_k^{\,g=1}) \in L^2(\Omega_k) \times H(div, \Omega_k)$,
- $(\phi_k^{g=2}, \vec{J}_k^{\,g=2}) \in L^2(\Omega_k) \times H(div, \Omega_k)$,
- $\Lambda_{k,l}^{g=1} \in H^{1/2}(\Gamma_{k,l})$,
- $\Lambda_{k,l}^{g=2} \in H^{1/2}(\Gamma_{k,l})$,

tels que $\forall\, v_k \in L^2(\Omega_k), \quad \forall\, \vec{w}_k \in H(div, \Omega_k), \quad \forall\, \mu_{k,l} \in V_{k,l}$

$$\left\{\begin{array}{l}
\displaystyle\int_{\Omega_k} div\big(\vec{J}_k^{g=1}\big)\, v_k\, d\vec{r} + \int_{\Omega_k} \Sigma_{a,k}^{g=1}\phi_k^{g=1} v_k\, d\vec{r} \; - \int_{\Omega_k} \Sigma_{s,k,0}^{2\to 1}\phi_k^{g=2} v_k\, d\vec{r} = \int_{\Omega_k} s_k^{g=1} v_k\, d\vec{r} \\
\displaystyle\int_{\Omega_k} \frac{\vec{J}_k^{g=1}}{D_k^{g=1}} \cdot \vec{w}_k\, d\vec{r} \quad - \int_{\Omega_k} \phi_k^{g=1}\, div\big(\vec{w}_k\big)\, d\vec{r} - \int_{\Omega_k} \Sigma_{s,k,1}^{2\to 1}\vec{J}_k^{g=2} \cdot \vec{w}_k\, d\vec{r} \\
\qquad\qquad + \displaystyle\sum_{l' \in \mathfrak{N}_k} \epsilon_{k,l'} \int_{\Gamma_{k,l'}} \Lambda_{k,l'}^{g=1} \vec{w}_k \cdot \vec{\hat{n}}_{k,l'}\, d\Gamma = 0 \\
\displaystyle\int_{\Omega_k} div\big(\vec{J}_k^{g=2}\big)\, v_k\, d\vec{r} + \int_{\Omega_k} \Sigma_{a,k}^{g=2}\phi_k^{g=2} v_k\, d\vec{r} \; - \int_{\Omega_k} \Sigma_{s,k,0}^{1\to 2}\phi_k^{g=1} v_k\, d\vec{r} = \int_{\Omega_k} s_k^{g=2} v_k\, d\vec{r} \\
\displaystyle\int_{\Omega_k} \frac{\vec{J}_k^{g=2}}{D_k^{g=2}} \cdot \vec{w}_k\, d\vec{r} \quad - \int_{\Omega_k} \phi_k^{g=2}\, div\big(\vec{w}_k\big)\, d\vec{r} - \int_{\Omega_k} \Sigma_{s,k,1}^{1\to 2}\vec{J}_k^{g=1} \cdot \vec{w}_k\, d\vec{r} \\
\qquad\qquad + \displaystyle\sum_{l' \in \mathfrak{N}_k} \epsilon_{k,l'} \int_{\Gamma_{k,l'}} \Lambda_{k,l'}^{g=2} \vec{w}_k \cdot \vec{\hat{n}}_{k,l'}\, d\Gamma = 0
\end{array}\right. \tag{4.1}$$

et

$$\left\{\begin{array}{l}
\displaystyle\int_{\Gamma_{k,l}} \Big(\vec{J}_k^{g=1} - \vec{J}_l^{g=1}\Big) \cdot \vec{\hat{n}}_{k,l}\, \mu_{k,l}\, d\Gamma = 0 \\
\displaystyle\int_{\Gamma_{k,l}} \Big(\vec{J}_k^{g=2} - \vec{J}_l^{g=2}\Big) \cdot \vec{\hat{n}}_{k,l}\, \mu_{k,l}\, d\Gamma = 0.
\end{array}\right. \tag{4.2}$$

$V_{k,l}$ désigne l'espace des multiplicateurs de Lagrange ($\Lambda_{k,l}^{g=1}$ et $\Lambda_{k,l}^{g=2}$) associés à la continuité faible de la composante normale du courant pour chaque groupe d'énergie. Comme on choisit pour chaque groupe d'énergie la même discrétisation en espace, on fait le même choix pour les multiplicateurs. Les matrices de couplage ne dépendent donc pas du groupe d'énergie considéré.

Dans chaque sous-domaine Ω_k on définit

- la matrice B_k correspondant à la matrice B_D^k définie dans le paragraphe 1.5.3. Dans ce chapitre, où l'on n'aborde pas les problématiques liées aux directions alternées, on omet l'indice D ;
- pour chaque groupe g, les matrices $A_k^{g=g}$ et $T_k^{g=g}$ correspondant à la matrice A_D^k et T_k du paragraphe 1.5.3 ;
- les matrices $U_k^{g\to g'}$ associées aux formes bilinéaires $u_k(\vec{J}_k, \overrightarrow{w}_k) = -\displaystyle\int_{\Omega_k} \Sigma_{s,k,1}^{g\to g'} \vec{J}_k \cdot \vec{w}_k\, d\vec{r}$;
- les matrices $V_k^{g\to g'}$ associées aux formes bilinéaires $v_k(\phi_k, v_k) = -\displaystyle\int_{\Omega_k} \Sigma_{s,k,0}^{g\to g'} \phi_k v_k d\vec{r}$;
- les vecteurs $S_k^{g=g}$ associés aux formes linéaires $s_k(u_k) = \displaystyle\int_{\Omega_k} s_k^{g=g} u_k\, d\vec{r}$.

On remarque que les matrices $diag(U_k^{g\to g'}, V_k^{g\to g'})$ correspondent aux matrices $H^{g\to g'}$ définies dans la partie monodomaine en considérant $\Omega = \Omega_k$.

La discrétisation de (4.1-4.2) conduit au système matriciel suivant :

$\forall k \in [\![1, K]\!]$,

$$\left(\begin{array}{cc|cc} A_k^{g=1} & -B_k & U_k^{2\to1} & \\ {}^tB_k & T_k^{g=1} & & V_k^{2\to1} \\ \hline U_k^{1\to2} & & A_k^{g=2} & -B_k \\ & V_k^{1\to2} & {}^tB_k & T_k^{g=2} \end{array}\right) \left(\begin{array}{c} J_k^{g=1} \\ \phi_k^{g=1} \\ J_k^{g=2} \\ \phi_k^{g=2} \end{array}\right) = \left(\begin{array}{c} 0 \\ S_k^{g=1} \\ 0 \\ S_k^{g=2} \end{array}\right) - \left(\begin{array}{c} \displaystyle\sum_{l\in\mathfrak{N}_k} \epsilon_{k,l}\, {}^tQ_{k,l} C_{k\to l} \Lambda_{k,l}^{g=1} \\ 0 \\ \displaystyle\sum_{l\in\mathfrak{N}_k} \epsilon_{k,l}\, {}^tQ_{k,l} C_{k\to l} \Lambda_{k,l}^{g=2} \\ 0 \end{array}\right) \tag{4.3}$$

et

$$\forall g' \in \{1, 2\}, \qquad \sum_{k=1}^{K} {}^tC_k \left(\begin{array}{c} J_k^{g=g'} \\ \phi_k^{g=g'} \end{array}\right) = 0. \tag{4.4}$$

En posant dans chaque sous-domaine Ω_k

- la matrice multigroupe $H_k = \left(\begin{array}{cc|cc} A_k^{g=1} & -B_k & U_k^{2\to1} & \\ {}^tB_k & T_k^{g=1} & & V_k^{2\to1} \\ \hline U_k^{1\to2} & & A_k^{g=2} & -B_k \\ & V_k^{1\to2} & {}^tB_k & T_k^{g=2} \end{array}\right)$;
- la matrice de couplage multigroupe $C_k^{mg} = \left(\begin{array}{cc} C_k & 0 \\ 0 & C_k \end{array}\right)$;
- le vecteur source multigroupe $L_k^{mg} = \left(\begin{array}{c} 0 \\ S_k^{g=1} \\ 0 \\ S_k^{g=2} \end{array}\right)$;
- le vecteur solution multigroupe $X_k^{mg} = \left(\begin{array}{c} J_k^{g=1} \\ \phi_k^{g=1} \\ J_k^{g=2} \\ \phi_k^{g=2} \end{array}\right)$;

et pour le problème global

- le vecteur des multiplicateurs multigroupes $\Lambda^{mg} = \left(\begin{array}{c} \Lambda_{g=1} \\ \Lambda_{g=2} \end{array}\right)$;

(4.3) et (4.4) s'écrivent alors sous la forme du système de point-selle suivant :

$$\left(\begin{array}{ccccc} H_1^{mg} & & & & C_1^{mg} \\ & \ddots & & & \vdots \\ & & H_k^{mg} & & C_k^{mg} \\ & & & \ddots & \vdots \\ & & & H_K^{mg} & C_K^{mg} \\ {}^tC_1^{mg} & \dots & {}^tC_k^{mg} & \dots & {}^tC_K^{mg} \quad 0 \end{array}\right) \left(\begin{array}{c} X_1^{mg} \\ \vdots \\ X_k^{mg} \\ \vdots \\ X_K^{mg} \\ \Lambda^{mg} \end{array}\right) = \left(\begin{array}{c} L_1^{mg} \\ \vdots \\ L_k^{mg} \\ \vdots \\ L_K^{mg} \\ 0 \end{array}\right). \tag{4.5}$$

Pour obtenir un système de point-selle similaire à (3.8) on pose

$$\hat{H} = \begin{pmatrix} H_1^{mg} & & & & \\ & \ddots & & & \\ & & H_k^{mg} & & \\ & & & \ddots & \\ & & & & H_K^{mg} \end{pmatrix}, \; \hat{X} = \begin{pmatrix} X_1^{mg} \\ \vdots \\ X_k^{mg} \\ \vdots \\ X_K^{mg} \end{pmatrix}, \; C = \begin{pmatrix} C_1^{mg} \\ \vdots \\ C_k^{mg} \\ \vdots \\ C_K^{mg} \end{pmatrix}, \; \hat{L} = \begin{pmatrix} L_1^{mg} \\ \vdots \\ L_k^{mg} \\ \vdots \\ L_K^{mg} \\ 0 \end{pmatrix}.$$

On obtient ainsi

$$\begin{pmatrix} \hat{H} & C \\ {}^tC & \end{pmatrix} \begin{pmatrix} \hat{X} \\ \Lambda \end{pmatrix} = \begin{pmatrix} \hat{L} \\ 0 \end{pmatrix}. \tag{4.6}$$

A la différence du système (3.8), la matrice $\hat{H}$ ainsi que le complément de Schur ${}^tC\hat{H}C$ ne sont pas symétriques ; on parle de système de point-selle généralisé. Dans la section suivante, on s'intéresse aux algorithmes de résolution de ce système.

4.2 Algorithme de résolution

Il existe une grande variété de méthodes permettant la résolution de problèmes de type point-selle [68]. On se focalise sur les méthodes de type Uzawa permettant de réutiliser les solveurs itératifs locaux. Dans un premier temps, on rappelle comment sont obtenus les algorithmes de type Uzawa, puis on détaillera comment ces algorithmes et les solveurs internes sont initialisés.

4.2.1 Algorithmes de type Uzawa

On considère le système (4.6). On fait l'hypothèse que les valeurs propres de la matrice $\hat{H}$ sont strictement positives[24]. En éliminant $\hat{X}$, on obtient le système condensé à l'interface faisant intervenir le complément de Schur

$${}^tC\hat{H}^{-1}C\Lambda = {}^tC\hat{H}^{-1}\hat{L}. \tag{4.7}$$

Seule *l'application*[25] *de* $\hat{H}^{-1}$ à un vecteur est à notre disposition. Elle est réalisée par l'appel en parallèle de tous les solveurs locaux. C'est pourquoi, on utilise un algorithme itératif pour la résolution de (4.7). Deux types d'algorithme sont à notre disposition :

– les algorithmes de descente qui s'écrivent

$$\Lambda_{k+1} = \Lambda_k + \rho_k \, {}^tC\hat{H}^{-1}\left(\hat{L} - C\Lambda_k\right)$$

où ρ_k dépend de la méthode choisie.
Le choix s'est porté sur l'algorithme MR[26] [47, p. 134] car il ne requiert pas la symétrie du système d'interface. L'utilisation de MR est valide suite à l'hypothèse de stricte positivité des valeurs propres de $\hat{H}$. La direction de descente est le résidu associé au système (4.7) et ρ_k provient de la minimisation du nouveau résidu. L'algorithme d'Uzawa (**Algorithme 11**)

[24] Cette hypothèse revient à faire l'hypothèse que les valeurs propres de la matrice multigroupe sont strictement positives, ce qui est en pratique le cas.
[25] L'opérateur n'est pas construit.
[26] MR pour Minimal Residual.

correspond à prendre ρ_k constant. Une des difficultés de l'algorithme d'Uzawa est le choix de ce paramètre ρ_k qui doit être choisi entre 0 et la plus grande valeur propre du système d'interface (valeur *a priori* inconnue) ;
- les algorithmes de Krylov tels que BiCGStab [47, p. 217], GMRES [47, p. 158], GC[27] (Gradient Conjugué [69]),[47, p. 176] , ...

Ces algorithmes nécessitent tous le calcul du résidu à la première itération. En considérant une initialisation de l'algorithme par Λ_0, le résidu r_0 s'écrit

$$\begin{aligned} r_0 &= {}^tC\hat{H}^{-1}\hat{L} - {}^tC\hat{H}^{-1}C\Lambda_0 \\ &= {}^tC\hat{H}^{-1}(\hat{L} - C\Lambda_0). \end{aligned}$$

Cette écriture permet d'économiser une résolution de système, car on n'a jamais besoin de construire explicitement le second membre du système d'interface. Ensuite dans la version standard de ces algorithmes itératifs résolvant le problème de point-selle, on recalcule le vecteur $\hat{X}$ après le calcul du vecteur d'interface Λ par

$$\hat{X} = \hat{H}^{-1}\big(\hat{L} - C\Lambda\big).$$

Avec les algorithmes de type Uzawa, on s'affranchit de cette dernière étape en calculant le vecteur $\hat{X}$ à chaque itération de l'algorithme résolvant le problème d'interface. Plus précisément, on a après une itération

$$\Lambda_1 = \Lambda_0 + \rho_1 d_1$$

où $d_1 = {}^tC\hat{H}^{-1}\big(\hat{L} - C\Lambda_0\big) = r_0$ est la première direction de descente (w ou r dans les algorithmes donnés dans la suite). Il vient donc

$$\begin{aligned} \hat{X}_1 &= \hat{H}^{-1}\big(\hat{L} - C\Lambda_1\big) \\ &= \hat{H}^{-1}\big(\hat{L} - C\Lambda_0\big) - \rho_1\hat{H}^{-1}Cd_1 \\ &= \hat{X}_0 - \rho_1 X_{d_1} \end{aligned}$$

avec $X_{d_i} = \hat{H}^{-1}Cd_i$. Comme le calcul de X_{d_1} est nécessaire pour le calcul de ρ_1 et de la direction de descente suivante[28] d_2, on recalcule $\hat{X}_1$ au moment du calcul de Λ_1, et ceci à chaque itération plutôt que de recalculer $\hat{X}$ en fin d'algorithme :

$$\begin{aligned} \hat{X} &= \hat{H}^{-1}(\hat{L} - C\Lambda) \\ &= \hat{H}^{-1}(\hat{L} - C\Lambda_0) - \sum_i \rho_i\hat{H}^{-1}Cd_i \\ &= \hat{X}_0 - \sum_i \rho_i X_{d_i}. \end{aligned}$$

Ainsi, on économise une inversion de $\hat{H}$. Par ailleurs, suite à l'inversion approchée[29] de $\hat{H}$, ce recalcul de $\hat{X}$ n'est pas équivalent au recalcul de $\hat{X}$ en fin d'algorithme. On a remarqué un meilleur comportement des algorithmes en recalculant $\hat{X}$ à chaque itération.

D'un point de vue algorithmique, à chaque modification de la variable duale Λ, on associe la modification correspondante de la variable primale X. On obtient au final les algorithmes de type Uzawa suivants : **Algorithme 12**, **Algorithme 13** et **Algorithme 14**. L'algorithme Uzawa-GC est cité dans ce chapitre, bien que l'on ne soit pas autorisé à l'utiliser du fait de la non symétrie du système d'interface, car

[27] Pour des systèmes symétriques.
[28] Ou pour un vecteur de la base de Krylov dans le cas des algorithmes de Krylov.
[29] On bloque à 1 le nombre d'itérations des boucles de Gauss-Seidel de tous les solveurs locaux à l'instar du solveur monodomaine.

– il s'agit du premier algorithme de type Uzawa basé sur un solveur de Krylov [70] ;
– dans [71], il a été testé et les résultats se sont avérés étonnement meilleurs qu'avec l'algorithme Uzawa-BiCGStab.

Algorithme 11 : Uzawa

$\Lambda = \Lambda_0$;
tant que *!Convergence* **faire**
  $\hat{X} = \hat{H}^{-1}(\hat{L} - C\Lambda)$;
  $\Lambda = \Lambda + \rho\, {}^tC\hat{X}$;
fin

Algorithme 12 : Uzawa-BiCGStab

$\Lambda = \Lambda_0$;
$r_0^* = 1$;
$\hat{X} = \hat{H}^{-1}(\hat{L} - C\Lambda)$;
$r = {}^tC\hat{X}$;
$p = r$;
tant que *!Convergence* **faire**
  $\hat{X}_p = \hat{H}^{-1}(Cp)$;
  $S_p = {}^tC\hat{X}_p$;
  $\alpha = \dfrac{< r, r_0^* >}{< S_p, r_0^* >}$;
  $s = r - \alpha S_p$;
  $\hat{X}_s = \hat{H}^{-1}(Cs)$;
  $S_s = {}^tC\hat{X}_s$;
  $\omega = \dfrac{< S_s, s >}{< S_s, S_s >}$;
  $\Lambda = \Lambda + \alpha p + \omega s$;
  $\hat{X} = \hat{X} - \alpha\hat{X}_p - \omega\hat{X}_s$;
  $r_{old} = r$;
  $r = s - \omega S_s$;
  $\beta = \dfrac{< r, r_0^* >}{< r_{old}, r_0^* >} \times \dfrac{\alpha}{\omega}$;
  $p = r + \beta(p - \omega S_p)$;
fin

Algorithme 13 : Uzawa-MR

$\Lambda = \Lambda_0$;
$\hat{X} = \hat{H}^{-1}(\hat{L} - C\Lambda)$;
$r = {}^tC\hat{X}$;
tant que *!Convergence* **faire**
  $\hat{X}_r = \hat{H}^{-1}(Cr)$;
  $M_r = {}^tC\hat{X}_r$;
  $\rho = -\dfrac{< r, M_r >}{< M_r, M_r >}$;
  $\Lambda = \Lambda + \rho r$;
  $r = r + \rho M_r$;
  $\hat{X} = \hat{X} - \rho\hat{X}_r$;
fin

Algorithme 14 : Uzawa-GC

$\Lambda = \Lambda_0$;
$\hat{X} = \hat{H}^{-1}(\hat{L} - C\Lambda)$;
$g = {}^tC\hat{X}$;
$w = g$; $dotgg =< g, g >$;
tant que *!Convergence* **faire**
  $\hat{X}_w = \hat{H}^{-1}(Cw)$;
  $M_w = {}^tC\hat{X}_w$;
  $\rho = -\dfrac{< g, w >}{< w, M_w >}$;
  $\Lambda = \Lambda + \rho w$;
  $g = g + \rho M_w$;
  $\hat{X} = \hat{X} - \rho\hat{X}_w$;
  $w = g + \dfrac{< g, g >}{dotgg} w$;
  $dotgg = < g, g >$;
fin

4.2.2 Initialisation des solveurs itératifs

4.2.2.1 Initialisation du solveur d'interface

Si l'on considère l'algorithme de la puissance sans accélération (cf. **Algorithme 3**), l'initialisation de l'algorithme itératif sur Λ s'effectue en choisissant pour Λ_0 la valeur de Λ à l'itération

précédente de l'algorithme de la puissance.

Dans la pratique, l'accélération de Tchebychev permet de réduire le nombre d'itérations de puissance [11, 12]. Cette accélération est appliquée sur les sources de fission (correspondant au vecteur S dans l'**Algorithme 1**). A l'itération n de l'algorithme de la puissance, la source de fission S_n est modifiée en fonction des itérations précédentes de la façon suivante

$$S_n = \alpha S_n + \beta S_{n-1} + \gamma S_{n-2}. \tag{4.8}$$

Les coefficients α, β et γ sont recalculés à chaque itération [12]. Avec les notations multigroupes, l'opération (4.8) devient au niveau du système de point-selle multidomaine

$$\hat{L}_n = \alpha \hat{L}_n + \beta \hat{L}_{n-1} + \gamma \hat{L}_{n-2}. \tag{4.9}$$

La même accélération est appliquée au multiplicateur Λ. Cela signifie qu'à l'itération n de la puissance, l'algorithme d'interface est initialisé par une combinaison linéaire des multiplicateurs obtenus aux itérations précédentes

$$\alpha\Lambda_{n-1} + \beta\Lambda_{n-2} + \gamma\Lambda_{n-3}. \tag{4.10}$$

Cette opération nécessite de stocker deux vecteurs d'interface supplémentaires (ce coût est faible). Cette opération peut également être faite sur le vecteur X de l'**Algorithme 3** pour mieux initialiser les solveurs locaux. Toutefois, le stockage de X étant plus important que celui de S, cela ne s'avère pas rentable en termes de temps de calcul.

4.2.2.2 Initialisation des solveurs locaux

Les algorithmes de type Uzawa font appel pour le calcul des directions de descente à la résolution itérative dégradée de systèmes linéaires impliquant la matrice $\hat{H}$. Selon le second membre du système, on adapte l'initialisation :

- le calcul de $\hat{H}^{-1}\big(\hat{L}_n - C\Lambda_{n-1}\big)$ est initialisé par le vecteur $\hat{X}^{n-1}$ qui vérifie $\hat{X}^{n-1} = \hat{H}^{-1}\big(\hat{L}_{n-1} - C\Lambda_{n-1}\big)$ où n désigne l'itération de puissance ;
- le calcul des intermédiaires ($\hat{X}_w$, $\hat{X}_r$, $\hat{X}_p$ ou $\hat{X}_s$) sont initialisés par le vecteur nul.

Le choix parmi ces algorithmes provient du contexte d'utilisation. La non-symétrie de la matrice d'interface ${}^tC\hat{H}^{-1}C$ exclut Uzawa-GC. On doit donc considérer, soit l'algorithme de point fixe Uzawa-MR, soit l'algorithme de Krylov Uzawa-BiCGStab[30]. Notre choix s'est porté sur l'algorithme Uzawa-MR car étant un algorithme de type point fixe, il supporte mieux la dégradation des résolutions itératives impliquant $\hat{H}$ que l'algorithme Uzawa-BiCGStab de type Krylov. Ce choix est confirmé par les résultats numériques obtenus.

4.3 Implémentation en mémoire distribuée

Dans cette partie, on décrit les choix faits concernant l'implémentation parallèle de la méthode proposée. La partie la plus critique concerne la distribution des données. Pour justifier nos choix, on fait l'hypothèse que la taille d'un sous-domaine est très supérieure à celle de l'interface qu'il porte. Cette hypothèse est réaliste pour la taille des problèmes visés. Considérons par exemple un calcul *crayon par crayon* cœur complet en RT0 à un groupe d'énergie[31] :

[30] Les résultats avec Uzawa-GMRES étant très peu satisfaisants, on se contente de présenter les résultats obtenus avec Uzawa-BiCGStab.

[31] Le nombre de groupes d'énergie n'intervient pas dans le rapport entre le nombre de degrés de liberté d'interface et de celui des degrés de liberté internes.

- la taille du maillage est donc $289 \times 289 \times 38$;
- le nombre de degrés de liberté de flux est 3 173 798 ;
- le nombre de degrés de liberté de courant est 9 626 879 ;
- le nombre total de degrés de liberté est 12 800 677.

En considérant le partitionnement $10 \times 10 \times 2$ à 200 sous-domaines[32], on obtient les encadrements suivants pour un sous-domaine :

- la taille du maillage est compris entre $28 \times 28 \times 19$ et $29 \times 29 \times 19$;
- le nombre de degrés de liberté de flux est compris entre 14 896 et 15 979 ;
- le nombre de degrés de liberté de courant est compris entre 78 392 et 82 940 ;
- le nombre de degrés de liberté total est compris entre 93 288 et 98 919 ;
- le nombre de degrés de liberté des multiplicateurs portés par un sous-domaine (au nombre maximal de 6) est majoré par $4.(19.29) + 2.(29.29) = 3\,886$.

Le rapport entre le nombre de degrés de liberté d'un sous-domaine et celui de la partie du vecteur d'interface concernant ce sous-domaine est donc plus grand que 24. Ce rapport permet de négliger le coût des opérations sur les vecteurs d'interface devant celles faites sur un vecteur contenant les degrés de liberté d'un sous-domaine. En revanche, ce facteur n'est pas suffisant pour pouvoir négliger les temps de communication.

4.3.1 Distribution des données

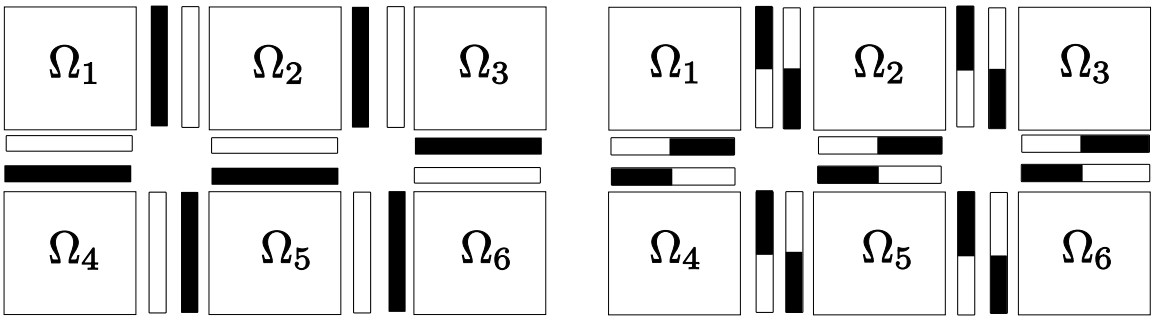
L'ensemble des données concernant un sous-domaine est stocké sur le même processeur. Les matrices $H_k^{g=g'}$, $H_k^{g\to g'}$, $F_k^{g=g'}$ et le vecteur $X_k^{g=g'}$ sont donc stockés sur le processeur k. Ainsi, on n'introduit pas de communications liées aux matrices de remontée et de descente en énergie, communications dont le volume est au minimum le nombre de degrés de liberté de flux d'un sous-domaine.

La taille des vecteurs d'interface étant négligeable vis-à-vis de la taille du vecteur d'inconnues dans chaque sous-domaine, l'équilibrage de la charge dépend de la taille des sous-domaines lors du partitionnement de Ω. La répartition des vecteurs d'interface (le multiplicateur Λ et les vecteurs issus de l'algorithme choisi pour résoudre le problème d'interface) est plus complexe. Suite à l'utilisation de l'élément fini de Raviart-Thomas, il n'existe en 3D aucun degré de liberté sur les angles et sur les arêtes d'un sous-domaine. Les degrés de liberté d'interface sont situés sur des surfaces communes à seulement deux sous-domaines. Un vecteur d'interface ω peut donc se décomposer suivant les interfaces entre chaque sous-domaine. La partie de ω correspondant à l'interface $\Gamma_{k,l}$ est notée $\omega_{k,l}$. Tous les vecteurs d'interface sont stockés de la même manière pour pouvoir faire les opérations de base avec le minimum de communications. Lors d'une opération AXPY ($Y = aX + Y$), on ne communique que le scalaire a. Pour le stockage de $\omega_{k,l}$, trois stratégies sont possibles :

1. $\omega_{k,l}$ est stocké soit par le processus k soit par le processus l ;
2. une partie de $\omega_{k,l}$ est stockée par le processus k et l'autre par le processus l ;
3. $\omega_{k,l}$ est stocké par le processus k et par le processus l .

▷ Les deux premières solutions sont illustrées sur les figures Fig. 4.1(a) et Fig. 4.1(b). Les parties en noir correspondent aux parties du vecteur dont un sous-domaine est responsable (responsable signifie qu'à la fin de chaque opération de base, cette partie contient la valeur à jour du vecteur d'interface). La partie blanche est un vecteur qui est utilisé comme buffer de

[32] Ce partitionnement permet d'obtenir des sous-domaines de taille équilibrée dans chacune des directions.



(a) sans partage (b) avec partage

FIG. 4.1 – Distribution des vecteurs d'interface

communication. Bien que la première solution puisse être considérée comme un cas particulier de la seconde, on a choisi de l'implémenter. En effet, lorsqu'on se place dans le cas de maillages non-coïncidents, le fait d'appliquer le produit par la matrice de couplage $C_{k\to l}$ permet de réduire la quantité de données à communiquer. Si le sous-domaine Ω_k est maillé plus finement que le sous-domaine Ω_l, la discrétisation du multiplicateur est guidée par Ω_k. Ainsi le nombre de degrés de liberté de courant de Ω_l présents sur $\Gamma_{k,l}$ est plus petit que le nombre de degrés de liberté du vecteur d'interface. Pour pouvoir tirer parti de cette propriété, l'ensemble du vecteur d'interface ainsi que les matrices de couplage $C_{k\to l}$ et $C_{l\to k}$ sont stockés[33] sur le processeur k, comme l'illustre la figure FIG. 4.2. Avec ce choix, on se prive pour des maillages coïncidents

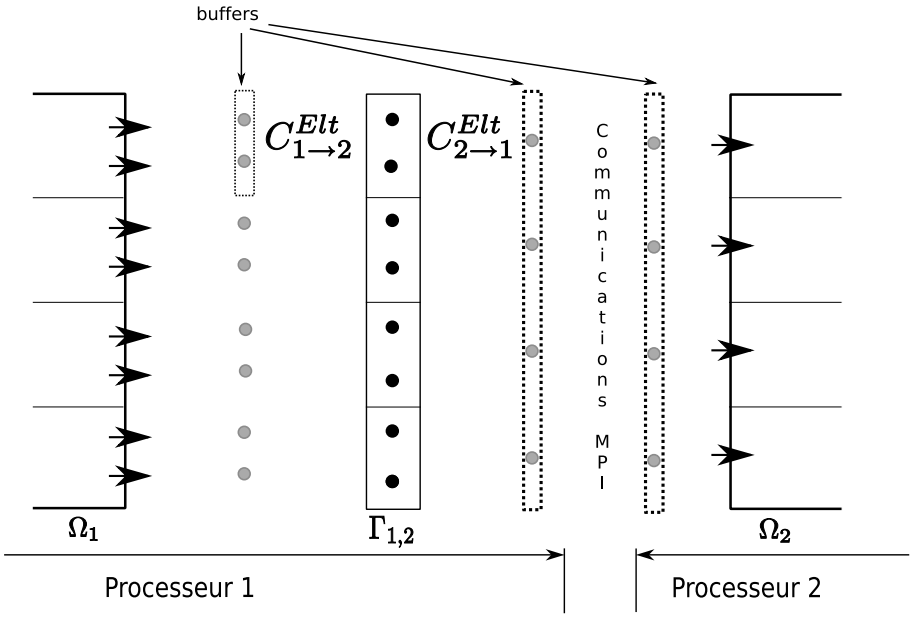


FIG. 4.2 – Schéma de communication

des possibilités offertes par la deuxième solution qui permet un meilleur équilibrage dans la répartition des vecteurs d'interface. Ce manque n'est pas crucial car

– la taille des vecteurs d'interface est négligeable devant la taille des sous-domaines ;

[33] Dans la pratique, on ne stocke qu'un des blocs diagonaux pour chaque matrice $C^{Elt}_{k\to l}$ et $C^{Elt}_{l\to k}$.

– quand le nombre de processeurs est élevé[34], il est possible de répartir les vecteurs d'interface de manière presque optimale avec la solution choisie ;
– une granularité plus faible permet de mieux équilibrer la charge, mais elle a également un coût : bien que le volume global des communications soit le même, leur nombre augmente.

Par la suite, le sous-domaine (Ω_k ou Ω_l) qui est responsable de l'interface $\Gamma_{k,l}$ est dit maître et l'autre est dit esclave.

▷ La troisième solution a été écartée pour l'approche multigroupe, car on se place dans le cadre de maillages non-coïncidents. En effet, cette solution ne permet pas de profiter de la réduction du volume des communications dues aux matrices de couplage. Pour l'approche unidimensionnelle, c'est la stratégie qui sera utilisée. Dans le paragraphe 5.4.1, on décrit donc plus en détails cette stratégie.

4.3.2 Schéma de communication

Tous les processeurs exécutent l'algorithme de la puissance et l'algorithme de résolution du système d'interface avec des synchronisations pour les données scalaires qui doivent être globales. Les produits scalaires issus de ces deux algorithmes sont donc calculés localement, puis les contributions de chaque processeur sont regroupées[35] à l'aide de la fonction MPI_Allreduce. Ces communications sont les seules communications globales. Les communications faisant intervenir des vecteurs sont des communications locales : il s'agit de communications point à point d'un processeur avec les processeurs gérant les sous-domaines voisins. Ces communications interviennent lors de deux opérations de base sur les vecteurs d'interface :

$X += C\Lambda$ **:** cette opération faite par **Algorithme 15** permet de prendre en compte les conditions limites induites par le multiplicateur ;

$\omega = {}^tCX$ **:** cette opération faite par **Algorithme 16**, permet de calculer le vecteur d'interface correspondant au saut de courant entre les sous-domaines.

4.4 Applications numériques

▷ Pour obtenir les résultats numériques, on a implémenté la méthode proposée avec les sources du solveur SPn de la plate-forme industrielle Cocagne écrit en C++. Dans un premier temps, on ne considère que le module SPn, sans prendre en compte l'interface python de plus haut-niveau permettant le chaînage avec d'autres modules (cf. page 96). Ce choix, à comparer au développement d'une maquette dans un langage de plus haut-niveau (Python, SciLab, ...), permet d'utiliser un solveur monodomaine optimisé et éprouvé :
– les ordres de grandeur entre temps de calcul et temps de communication sont représentatifs ;
– les temps de référence pour le calcul séquentiel peuvent constituer une référence valable pour le calcul de l'efficacité parallèle ;
– des problèmes industriels (tant au niveau de la taille et que des sections) peuvent être traités ;
– l'impact de la méthode dans le code existant est simple à évaluer.

[34] On peut remarquer, que les limites de validité de l'hypothèse conduisant au premier point sont atteintes avec un nombre de processeurs élevé.
[35] Le résultat est redistribué sur chaque processeur.

Algorithme 15 : Calculer $X+ = C\Lambda$

▷ *Exécuté sur le processus k*
pour chaque *l interface esclave* **faire**
  Lancer les communications en réception venant de l;
pour chaque *l interface maître* **faire**
  Calculer $C_{l\to k}\Lambda_{k,l}$;
  Lancer les communications pour envoi vers l;
pour chaque *l interface maître* **faire**
  Calculer $C_{k\to l}\Lambda_{k,l}$ pour mettre à jour X_k;
  Attendre que les données soient envoyées à l ;
pour chaque *l interface esclave* **faire**
  Attendre que les données provenant de l soient reçues ;
  Utiliser $C_{k\to l}\Lambda_{k,l}$ pour mettre à jour X_k;

Algorithme 16 : Calculer $\omega = {}^tCX$

▷ *Exécuté sur le processus k*
pour chaque *l interface maître* **faire**
  Lancer les communications en réception venant de l;
  Calculer $\omega_{k,l} = {}^tC_{k\to l}X_k$;
pour chaque *l interface esclave* **faire**
  Calculer $tmp_{k,l} = {}^tC_{k\to l}X_k$;
  Lancer l'envoi de $tmp_{k,l}$ vers l ;
pour chaque *l interface maître* **faire**
  Attendre la réception des données ($tmp_{k,l}$) de l;
  $\omega_{k,l}+ = tmp_{k,l}$;
pour chaque *l interface esclave* **faire**
  Attendre que les données soient envoyées vers l;

Le contre-coût de ce choix réside dans une difficulté accrue des développements.

▷ L'algorithme de décomposition de domaine conduit à ajouter un niveau d'itérations dans l'algorithme global. Pour rester cohérent avec la stratégie du solveur séquentiel, il convient de bloquer à un faible nombre d'itérations l'algorithme de résolution du système d'interface. On note donc :

- `MRi` l'algorithme Uzawa-MR bloqué à i itérations ;
- `BiCGStabi` l'algorithme Uzawa-BiCGStab bloqué à i itérations.

Pour ces algorithmes, on reporte dans le tableau Tab. 4.1 le coût d'une itération de puissance en prenant comme unité le coût d'une itération de puissance du solveur séquentiel. L'algorithme BiCGStab est plus coûteux que l'algorithme MR car il fait deux appels aux solveurs locaux par itération du solveur d'interface.

Comme pour chaque algorithme, le nombre d'itérations des solveurs internes est fixé ; on caractérise la convergence par le nombre d'itérations externes. La solution (notée k_{eff} pour la valeur propre et X pour le vecteur propre) obtenue pour le critère de convergence $\epsilon = 10^{-6}$ (défini page 26) est comparée à

- la solution de référence, notée (k_{eff}^{ref}, X^{ref}), obtenue avec le solveur monodomaine en poussant la convergence de la boucle externe à 2000 itérations ;
- la solution du solveur monodomaine, notée $(k_{eff}^{mono}, X^{mono})$, obtenue avec le solveur séquentiel en utilisant le même critère de convergence ($\epsilon = 10^{-6}$).

▷ Le partitionnement du cœur dépend du nombre de sous-domaines K_d dans chaque direction d. Le nombre de sous-domaines est égal au produit $K_x \times K_y \times K_z$. Entre deux et huit sous-domaines, le domaine n'est partitionné que suivant la direction x. Avec un plus grand nombre de sous-domaines, on équilibre autant que possible le partitionnement dans la direction x et la direction y. Le domaine n'est pas coupé axialement car

- le nombre de mailles dans la direction z est faible (38 ou 40) ;
- le nombre de sous-domaines dans la direction z doit diviser le nombre de mailles pour conserver un bon équilibrage de la charge ;
- en prévision d'un couplage avec un module thermo-hydraulique monodimensionnel considérant un flux axial (dans la direction z), ce type de partitionnement permet de limiter l'impact du parallélisme dans ce solveur.

Les résultats présentés correspondent aux partitionnements suivants :

$$(K_x, K_y, K_z) \in \{(i, i, 1) \quad | \quad 3 \leq i \leq 17\} \cup \{(2, 1, 1), (4, 1, 1), (8, 1, 1)\}.$$

Le découpage est réalisé au niveau du maillage de calcul ; on est donc amené à placer les interfaces au sein des assemblages, à l'exception de la partition $(17, 17, 1)$ où un sous-domaine correspond à un assemblage.

Le cluster utilisé (annexe D) limite l'analyse de performance à 196 sous-domaines. Lors du processus de soumission de jobs parallèles, on associe un sous-domaine à un nœud car les

Algorithme	`BiCGStab2`	`MR2`	`MR3`	`MR4`	`BiCGStabi`	`MRi`
Coût	5	3	4	5	$2i + 1$	$i + 1$

Tab. 4.1 – Estimation du coût des algorithmes d'interface : nombre d'exécutions du solveur local dans chaque sous-domaine à chaque itération de puissance.

problèmes de contention mémoire (cf. figure FIG. D.1(b)) ne permettent pas de tirer parti des deux processeurs par nœud. Pour la mesure du temps d'exécution, on exécute plusieurs fois le code et on garde le meilleur temps. Ce dernier prend en compte la construction des matrices et l'exécution de la boucle de la puissance. Pour l'analyse des critères numériques (nombre d'itérations, erreur sur la valeur propre, erreur sur le vecteur propre), on utilise plusieurs processus par nœuds pour pouvoir atteindre 289 sous-domaines.

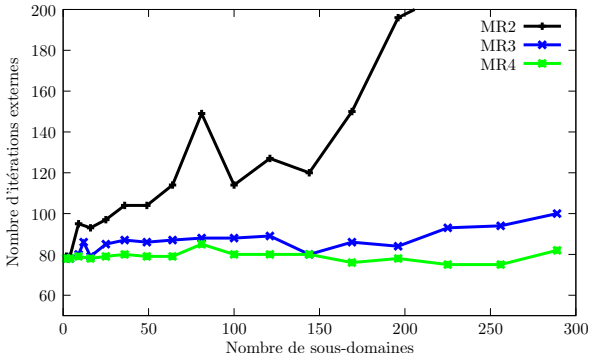
4.4.1 BenchMarkAIEA

Pour un calcul SP1 avec l'élément fini RT0, on compare l'utilisation de l'algorithme de point fixe Uzawa-MR et l'algorithme de Krylov Uzawa-BiCGStab. La comparaison entre ces deux algorithmes (cf. figure FIG. 4.3) met en évidence un meilleur comportement de l'algorithme Uzawa-MR. En effet lorsque que l'algorithme Uzawa-MR ne permet pas la convergence de l'algorithme de la puissance en un nombre raisonnable d'itérations (ici fixé à 200), l'augmentation du nombre d'itérations de l'algorithme MR améliore ce comportement (cf. figure FIG. 4.3(a)). Avec l'algorithme Uzawa-BiCGStab, on n'observe pas ce comportement. En effet `BiCGStab3` nécessite souvent plus d'itérations externes que `BiCGStab2` : sur la figure FIG. 4.3(b), on observe qu'au delà de 100 sous-domaines, la boucle de la puissance ne converge pas en 200 itérations. Le meilleur comportement de l'algorithme Uzawa-MR par rapport à Uzawa-BiCGStab est expliqué page 66. Sur les figures FIG. 4.3(c), FIG. 4.3(d), FIG. 4.3(e) et FIG. 4.3(f), on observe que la solution obtenue est aussi précise que le solveur monodomaine pour l'ensemble des algorithmes présentés.

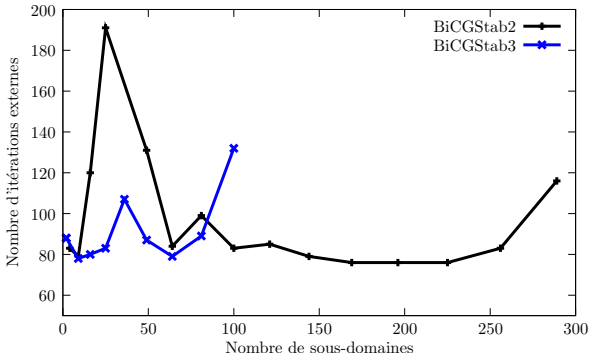
Sur la figure FIG. 4.4(a), on représente le facteur d'accélération (speed-up) pour les algorithmes `MRi`. Pour l'algorithme `MR3` (respectivement `MR4`) le facteur d'accélération est linéaire, correspondant à une efficacité parallèle de 20% (respectivement 18%) pour une efficacité théorique de 25% (respectivement 20%) liée aux 4 appels (respectivement 5) aux solveurs locaux (cf tableau TAB. 4.1). Pour `MR2`, l'augmentation du nombre d'itérations externes pour certaines partitions crée des décrochages sur le plan de la performance. Au final le meilleur compromis entre le nombre d'itérations externes et le coût des itérations du système d'interface est atteint avec le solveur `MR3`. Ce compromis est également le meilleur pour les calculs SP3-RT0 et SP1-RT1. Avec la figure FIG. 4.4(b), on constate que l'efficacité est plus faible pour les calculs SP3 (l'efficacité est comprise en 5% et 10%) suite à une augmentation du nombre d'itérations externes.

On est déçu par le fait que le meilleur compromis ne soit pas atteint avec `MR1` à l'instar du solveur monodomaine où l'on bloque toutes les itérations internes (algorithmes de Gauss-Seidel) à 1. Néanmoins, atteindre un facteur d'accélération de 40 constitue déjà un bon résultat au vue des précédentes tentatives (chapitre 2.2 et chapitre 2.3). Le facteur d'accélération de 10 pour les calculs SP3 reste faible : ce constat est à relativiser par le fait que la méthode permet de diminuer l'empreinte mémoire et que l'ensemble des résultats décrits dans la littérature ne concernent que les équations SP1.

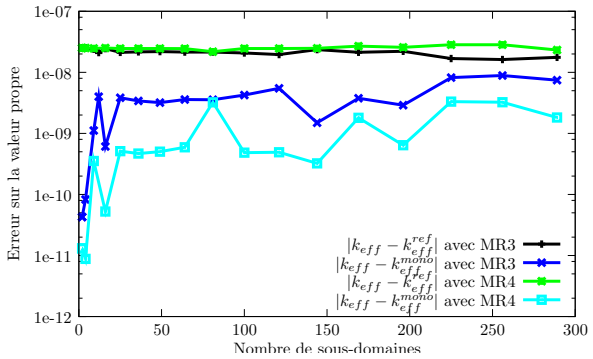
Malgré ces résultats mitigés, on teste la méthode avec des sections provenant d'un cas industriel.



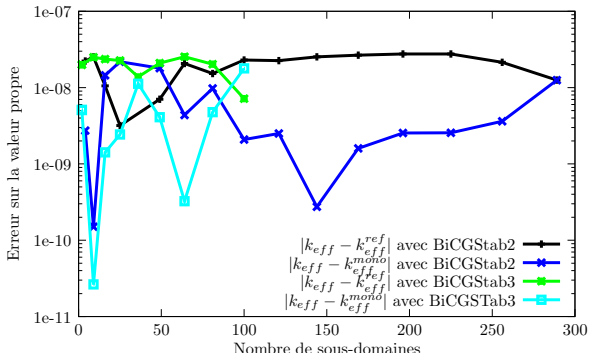
(a) Nombre d'itérations externes avec Uzawa-MR



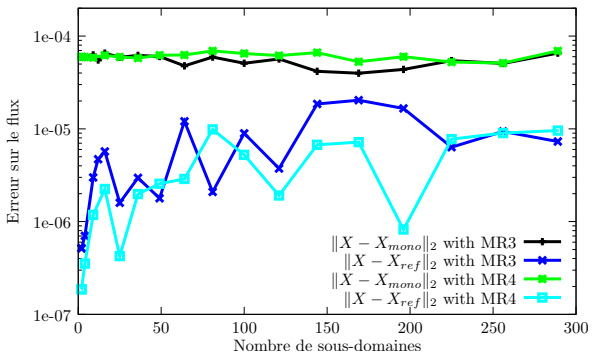
(b) Nombre d'itérations externes avec Uzawa-BiCGStab



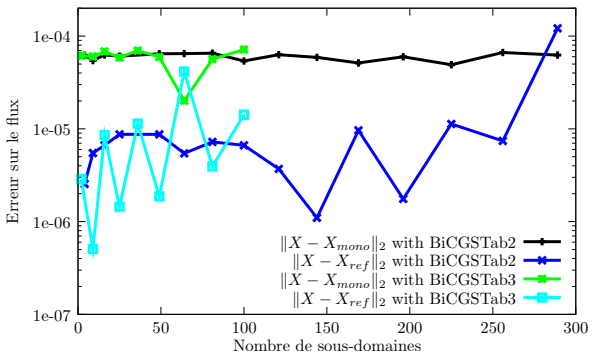
(c) Erreur sur k_{eff} avec Uzawa-MR



(d) Erreur sur k_{eff} avec Uzawa-BiCGStab

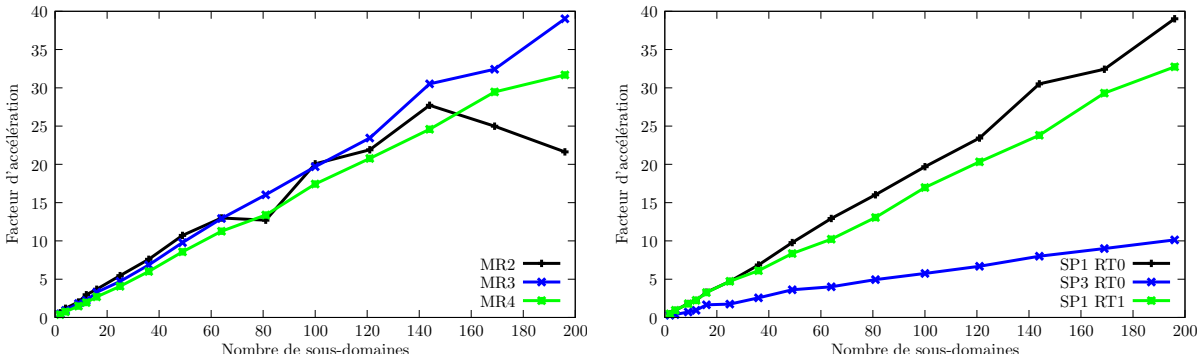


(e) Erreur sur le flux avec Uzawa-MR



(f) Erreur sur le flux avec Uzawa-BiCGStab

FIG. 4.3 – BenchMarkAIEA : comparaison des algorithmes Uzawa-MR et Uzawa-BiCGStab



(a) Comparaison entre algorithmes de type Uzawa-MR (b) L'algorithme `MR3` pour différentes discrétisations

FIG. 4.4 – BenchMarkAIEA : facteur d'accélération

4.4.2 REP 900MW

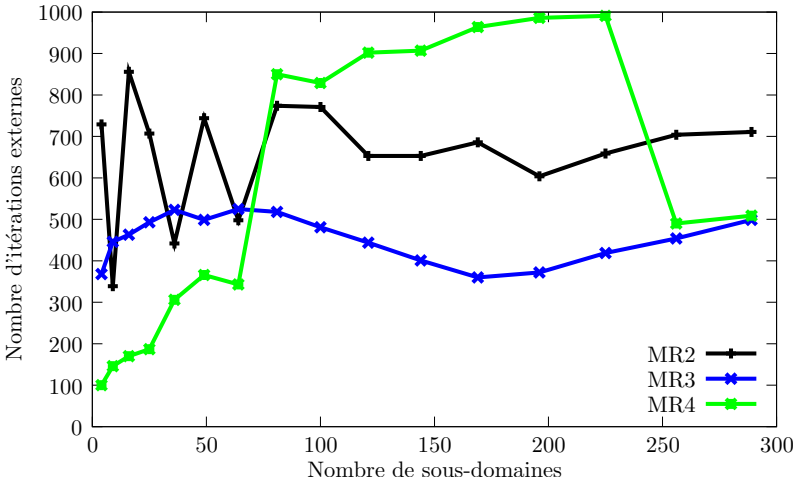
Avec les données provenant d'une campagne REP 900MW, l'augmentation du nombre d'itérations de puissance (cf. figure FIG. 4.5(a)) rend la méthode peu *scalable* (cf. figure FIG. 4.5(a)). Si la précision de la solution obtenue est acceptable avec l'algorithme `MR4` avec moins de 225 sous-domaines, on ne peut pas se satisfaire de la précision obtenue avec l'algorithme `MR3` (cf. figures FIG. 4.5(b) et FIG. 4.5(c)). On n'explique pas cette augmentation du nombre d'itérations externes qui ne se produit pas sur le cas test académique BenchMarkAIEA. Toutefois quelques pistes peuvent être mises en avant :

– les sections d'entrée sont plus hétérogènes ;
– les pas du maillage axial ne sont pas réguliers.

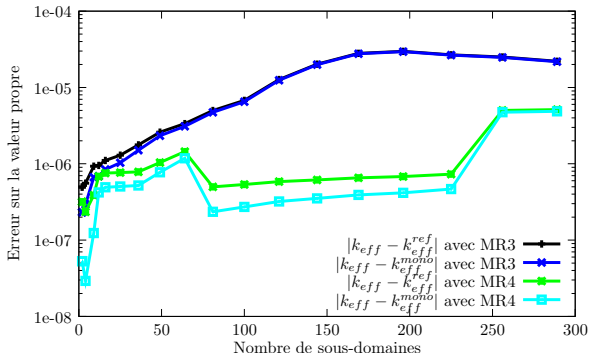
La présence d'up-scattering seule n'explique pas ce phénomène, qui a été aussi observé sur le cas test du REP 900MW où les termes d'up-scattering ont été supprimés.

Sur le plan de la performance, 9 nœuds sont nécessaires à `MR4` pour obtenir un code parallèle aussi rapide que le code séquentiel. Avec 25 nœuds, le code parallèle est deux fois plus rapide. L'implémentation n'a pas été optimisée de manière poussée, car le principal problème de la méthode réside dans l'augmentation du nombre d'itérations et non dans une implémentation non-optimale de la méthode.

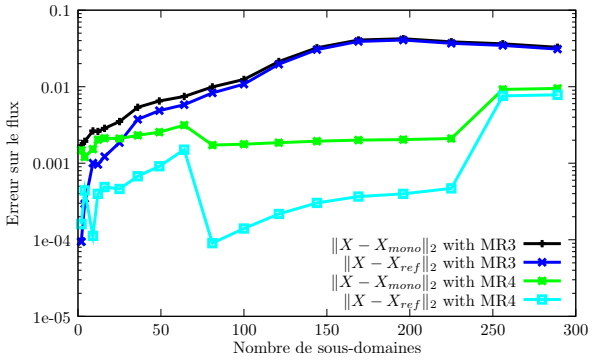
Cette méthode n'est pas à envisager pour obtenir un code plus rapide, mais uniquement pour résoudre des problèmes de consommation mémoire. Suite à ces résultats décevants, on s'est focalisé sur l'approche unidimensionnelle pour éviter les difficultés provenant de l'augmentation du nombre d'itérations externes. Il aurait également été intéressant d'étudier l'ajout d'un préconditionneur pour résoudre les problèmes de convergence de la méthode proposée. Par souci de simplicité de développement, on n'a donc pas choisi cette voie.



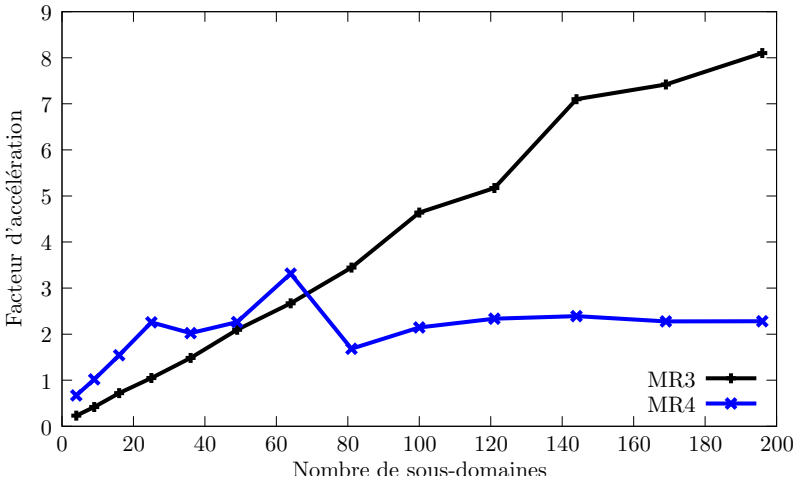
(a) Nombre d'itérations externes



(b) Erreur sur k_{eff}



(c) Erreur sur le flux



(d) Facteur d'accélération

FIG. 4.5 – REP 900Mw

Chapitre 5

Approche unidimensionnelle

Sommaire

Suite aux difficultés rencontrées avec l'approche multigroupe, on a implémenté l'approche unidimensionnelle. Celle-ci consiste à placer la boucle de décomposition de domaine sous l'algorithme de Gauss-Seidel des directions alternées (**Algorithme 10**). Les principaux avantages de cette approche sont :

- l'algorithme de décomposition de domaine se base sur des solveurs locaux directs, permettant ainsi de réduire les difficultés provenant des itérations imbriquées. En cas de difficultés liées aux itérations imbriquées, il est possible d'envisager l'utilisation d'un solveur d'interface direct ayant un coût calculatoire raisonnable. Les résultats montrent qu'un solveur itératif est suffisant ;
- étant en dessous des algorithmes de Gauss-Seidel, plusieurs appels aux solveurs locaux ne dupliquent pas les produits par les termes extra-diagonaux des matrices de niveau supérieur ;
- le système d'interface est symétrique défini positif.

Le principal inconvénient de la méthode réside dans la difficulté d'implémentation. Cette méthode nécessite de connaître l'ensemble du solveur multigroupe. Le fait que le stockage des données se fasse au niveau multigroupe, alors que l'algorithme de résolution se situe à un niveau plus bas, rend complexe l'implémentation d'une solution avec plusieurs sous-domaines par processus MPI. Celle-ci a ainsi été repoussée dans les perspectives.

Contrairement à l'approche multigroupe où l'on cherche à rester le plus générique possible, la performance est plus privilégiée dans le développement de cette approche. Ainsi, on n'a implémenté la méthode que dans le cas des maillages coïncidents, tout en prenant soin de montrer comment il est possible d'adapter la méthode pour les maillages non-coïncidents.

Dans un premier temps, on montre comment on se ramène à une méthode de décomposition de domaine unidimensionnelle. Ensuite on décrit la méthode de résolution et les optimisations possibles pour les maillages coïncidents. Enfin, les résultats numériques mettent en évidence la pertinence de la méthode.

Ce chapitre correspond aux travaux développés dans [72].

5.1 Placement de la boucle de décomposition de domaine

A l'aide de quelques manipulations de nature algébrique, on transforme le problème de décomposition de domaine sur les équations de la diffusion en plusieurs problèmes de décomposition de domaine unidimensionnels. Cette transformation revient à placer la boucle de décomposition de domaine sous la boucle des directions de l'espace provenant des directions alternées.

Afin de garder des notations lisibles, on ne considère que deux sous-domaines d'un espace à deux dimensions. En revanche, pour être convaincant il est nécessaire d'utiliser un partitionnement suivant deux directions. On considère donc deux sous-domaines ayant à la fois une interface suivant x et une suivant y (cf. figure Fig. 5.1). L'implémentation informatique ne gère pas de domaine non parallélépipédique (comme Ω_1), mais il n'y a aucune difficulté à considérer ce type de sous-domaine au niveau matriciel.

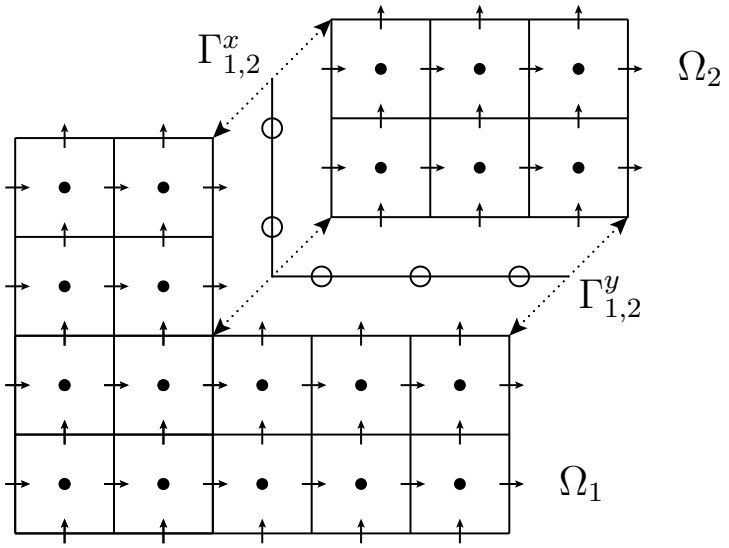


FIG. 5.1 – Partitionnement en deux sous-domaines séparés par deux interfaces

Dans ce cadre, on est amené à résoudre le système suivant

$$\left(\begin{array}{ccc|ccc|cc} A^1_x & & -B^1_x & & & & C^x_{\Lambda\to 1} & \\ & A^1_y & -B^1_y & & & & & C^y_{\Lambda\to 1} \\ {}^tB^1_x & {}^tB^1_y & T^1 & & & & & \\ \hline & & & A^2_x & & -B^2_x & C^x_{\Lambda\to 2} & \\ & & & & A^2_y & -B^2_y & & C^y_{\Lambda\to 2} \\ & & & {}^tB^2_x & {}^tB^2_y & T^2 & & \\ \hline {}^tC^x_{\Lambda\to 1} & & & {}^tC^x_{\Lambda\to 2} & & & & \\ & {}^tC^y_{\Lambda\to 1} & & & {}^tC^y_{\Lambda\to 2} & & & \end{array}\right)\left(\begin{array}{c} J^1_x \\ J^1_y \\ \phi^1 \\ \hline J^2_x \\ J^2_y \\ \phi^2 \\ \hline \Lambda_x \\ \Lambda_y \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ S^1 \\ \hline 0 \\ 0 \\ S^2 \\ \hline 0 \\ 0 \end{array}\right). \quad (5.1)$$

Lors de la première étape de cette transformation, les flux sont éliminés dans chaque sous-domaine Ω_i de la même manière que pour le solveur séquentiel

$$\phi_i = T_i^{-1}(S_i - {}^tB^i_x J^i_x - {}^tB^i_y J^i_y). \quad (5.2)$$

Les flux sont réintroduits dans les équations

$$A^i_d J^i_d - B^i_d \phi_i + C^d_{\Lambda\to i}\Lambda_d = 0.$$

Pour chacune des directions de l'espace d, J^i_d satisfait l'équation suivante

$$\left(A^i_d + B^i_d T_i^{-1\,t}B^i_d\right) J^i_d + \left(B^i_d T_i^{-1\,t}B^i_{d'}\right) J^i_{d'} + C^d_{\Lambda\to i}\Lambda_d = B^i_d T_i^{-1} S.$$

En posant $W^i_d = A^i_d + B^i_d T_i^{-1\,t}B^i_d$ et $W^i_{d,d'} = B^i_d T_i^{-1\,t}B^i_{d'}$, on obtient le système suivant

$$\left(\begin{array}{cc|cc|cc} W^1_x & W^1_{x,y} & & & C^x_{\Lambda\to 1} & \\ W^1_{y,x} & W^1_y & & & & C^y_{\Lambda\to 1} \\ \hline & & W^2_x & W^2_{x,y} & C^x_{\Lambda\to 2} & \\ & & W^2_{y,x} & W^2_y & & C^y_{\Lambda\to 2} \\ \hline {}^tC^x_{\Lambda\to 1} & & {}^tC^x_{\Lambda\to 2} & & & \\ & {}^tC^y_{\Lambda\to 1} & & {}^tC^y_{\Lambda\to 2} & & \end{array}\right)\left(\begin{array}{c} J^1_x \\ J^1_y \\ \hline J^2_x \\ J^2_y \\ \hline \Lambda_x \\ \Lambda_y \end{array}\right) = \left(\begin{array}{c} B^1_x T_1^{-1} S^1 \\ B^1_y T_1^{-1} S^1 \\ \hline B^2_x T_2^{-1} S^2 \\ B^2_y T_2^{-1} S^2 \\ \hline 0 \\ 0 \end{array}\right). \quad (5.3)$$

Grâce à un réordonnancement matriciel par direction, (5.3) devient

$$\left(\begin{array}{ccc|ccc} W_x^1 & & C_{\Lambda\to1}^x & W_{x,y}^1 & & \\ & W_x^2 & C_{\Lambda\to2}^x & & W_{x,y}^2 & \\ {}^tC_{\Lambda\to1}^x & {}^tC_{\Lambda\to2}^x & & & & \\ \hline W_{y,x}^1 & & & W_y^1 & & C_{\Lambda\to1}^y \\ & W_{y,x}^2 & & & W_y^2 & C_{\Lambda\to2}^y \\ & & & {}^tC_{\Lambda\to1}^y & {}^tC_{\Lambda\to2}^y & \end{array}\right)\left(\begin{array}{c} J_x^1 \\ J_x^2 \\ \Lambda_x \\ \hline J_y^1 \\ J_y^2 \\ \Lambda_y \end{array}\right) = \left(\begin{array}{c} B_x^1T_1^{-1}S^1 \\ B_x^2T_2^{-1}S^2 \\ 0 \\ \hline B_y^1T_1^{-1}S^1 \\ B_y^2T_2^{-1}S^2 \\ 0 \end{array}\right).$$

Cette matrice bloc de taille[36] 2×2 est inversée grâce à un algorithme de Gauss-Seidel par bloc. Les blocs extra-diagonaux étant diagonaux par bloc, leur produit se parallélise aisément car il s'agit de produits indépendants dans chaque sous-domaine. La difficulté restant à lever est la résolution du système linéaire suivant (le second membre provient de l'algorithme de Gauss-Seidel)

$$\left(\begin{array}{ccc} W_d^1 & & C_{\Lambda\to1}^d \\ & W_d^2 & C_{\Lambda\to2}^d \\ {}^tC_{\Lambda\to1}^d & {}^tC_{\Lambda\to2}^d & \end{array}\right)\left(\begin{array}{c} J_d^1 \\ J_d^2 \\ \Lambda_d \end{array}\right) = \left(\begin{array}{c} F_d^1 \\ F_d^2 \\ 0 \end{array}\right). \tag{5.4}$$

On introduit les notations suivantes[37]

$$\hat{W}_d = \left(\begin{array}{cc} W_d^1 & \\ & W_d^2 \end{array}\right) \quad C^d = \left(\begin{array}{c} C_{\Lambda\to1}^d \\ C_{\Lambda\to2}^d \end{array}\right) \quad \hat{J}^d = \left(\begin{array}{c} J_1^d \\ J_2^d \end{array}\right) \quad \hat{F}^d = \left(\begin{array}{c} F_1^d \\ F_2^d \end{array}\right).$$

Pour chaque direction d, le système de point-selle suivant doit être résolu

$$\left(\begin{array}{cc} \hat{W}_d & C^d \\ {}^tC^d & \end{array}\right)\left(\begin{array}{c} \hat{J}_d \\ \Lambda_d \end{array}\right) = \left(\begin{array}{c} \hat{F}_d \\ 0 \end{array}\right). \tag{5.5}$$

On a donc transformé une décomposition de domaine bidimensionnelle (respectivement tridimensionnelle) en plusieurs résolutions de problèmes unidimensionnels par décomposition de domaine. Par la suite l'indice d sera omis, lorsque que cela ne prête pas à confusion.

5.2 Algorithme de résolution

Λ est solution du système d'interface suivant

$$\left({}^tC\hat{W}^{-1}C\right)\Lambda = {}^tC\hat{W}^{-1}\hat{F}\,. \tag{5.6}$$

Comme $\hat{W}$ est symétrique définie positive et C de rang plein, la matrice d'interface provenant du système (5.6) est donc également symétrique définie positive. Ce système est alors résolu *via* un algorithme de Gradient Conjugué préconditionné. Ensuite $\hat{J}$ est calculé par la relation $\hat{J} = \hat{W}^{-1}\left(\hat{F} - C\Lambda\right)$. L'**Algorithme 17** décrit ces deux opérations.

[36] De taille 3×3 en 3D.

[37] L'exposant $\hat{}$ permet de distinguer la matrice assemblée W_d sur l'ensemble du domaine Ω de la matrice bloc diagonale $\hat{W}_d$ constituée des matrices W_d^i locales à chaque sous-domaine Ω_i.

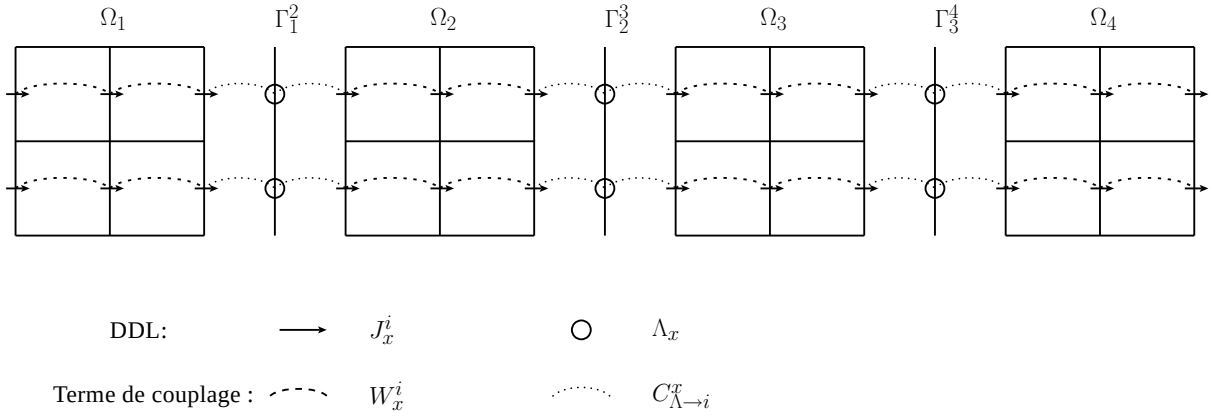


FIG. 5.2 – Ω est partitionné en 4 sous-domaines dans la direction x. Comme on s'intéresse à la résolution par décomposition de domaine dans la direction x après élimination des flux, le flux ϕ_i, le courant J_i^y, les termes de couplages B_i^x, B_i^y et A_i^y sont ignorés.

Pour décrire le préconditionneur utilisé, il est nécessaire d'introduire la version multidomaine du système matriciel. Avec un partitionnement unidirectionnel en K sous-domaines (illustré par la figure FIG. 5.2 pour $K = 4$), les notations suivantes sont introduites :

$$\hat{W} = \begin{pmatrix} W^1 & & & \\ & W^2 & & \\ & & \ddots & \\ & & & W^K \end{pmatrix} \quad \hat{J} = \begin{pmatrix} J_1 \\ J_2 \\ \vdots \\ J_K \end{pmatrix} \quad \hat{F} = \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_K \end{pmatrix}$$

$$C = \begin{pmatrix} C_{\Lambda_1^2\to 1} & & & \\ C_{\Lambda_1^2\to 2} & \ddots & & \\ & \ddots & C_{\Lambda_{K-1}^K\to K-1} & \\ & & C_{\Lambda_{K-1}^K\to K} & \end{pmatrix} \quad \Lambda = \begin{pmatrix} \Lambda_{1,2} \\ \Lambda_{2,3} \\ \vdots \\ \Lambda_{K,K-1} \end{pmatrix}.$$

Avec $S_{i\to j}^k = {}^tC_{\Lambda_k^j\to j}(W^j)^{-1}C_{\Lambda_i^j\to j}$ la matrice de (5.6) appelée S s'écrit

$$S = {}^tC\hat{W}^{-1}C = \begin{pmatrix} \left(S_{1\to 2}^1 + S_{2\to 1}^2\right) & S_{3\to 2}^1 & & \\ S_{1\to 2}^3 & \ddots & \ddots & \\ & \ddots & \ddots & S_{K\to K-1}^{K-2} \\ & & S_{K-2\to K-1}^K & \left(S_{K-1\to K}^{K-1} + S_{K\to K-1}^K\right) \end{pmatrix}.$$

On choisit le préconditionneur bloc diagonal

$$P^{-1} = \begin{pmatrix} \left(S_{1\to 2}^1 + S_{2\to 1}^2\right) & & \\ & \ddots & \\ & & \left(S_{K-1\to K}^{K-1} + S_{K\to K-1}^K\right) \end{pmatrix}.$$

Généralement ce préconditionneur est coûteux en termes de temps de calcul et de consommation mémoire, car chacun des blocs est dense. Grâce aux propriétés des éléments finis de Raviart-Thomas, on obtient une implémentation efficace de ce préconditionneur.

Algorithme 17 : Gradient Conjugué Préconditionné

entrée : $\hat{F}$
entrée/sortie : Λ initialisé à Λ_0
sortie : $\hat{J}$

$\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda\right)$;
$g = \left({}^t\hat{C}\hat{J}\right)$;
$z = Pg$;
$w = z$; $dotGZ = < g, z >$;
while *! Convergence* **do**
  $S_w = ({}^t\hat{C}\hat{W}^{-1}\hat{C})w$;
  $\rho = -\dfrac{dotGZ}{< w, S_w >}$;
  $\Lambda = \Lambda + \rho w$;
  $g = g + \rho S_w$;
  $z = Pg$;
  $dotGZold = dotGZ$; $dotGZ = < g, z >$;
  $w = z + \dfrac{dotGZ}{dotGZold} w$;
end
$\hat{J} = \hat{W}^{-1}\left(\hat{F} - \hat{C}\Lambda\right)$

5.3 Optimisations dans le cadre des maillages coïncidents

Les optimisations présentées sont basées sur la structure très creuse des matrices $S^k_{i\to j}$ qui peuvent ainsi être stockées. En effet, la méthode naturelle pour calculer le produit par une matrice $S^k_{i\to j}$ consiste à faire un appel au solveur local $(W^j)^{-1}$ qui a une complexité linéaire avec le nombre de degrés de liberté de courant dans la direction d. Pour des maillages coïncidents, les lignes de courant sont indépendantes (cf. figure Fig. 5.2). Les matrices $S^k_{i\to j}$ sont donc diagonales ; on peut donc facilement les stocker et les opérations sur ces matrices telle que le produit par un vecteur sont ainsi linéaires avec le nombre de degrés de liberté du multiplicateur entre les deux sous-domaines Ω_i et Ω_j. Pour construire les matrices $S^k_{i\to j}$, le produit par un vecteur d'interface ne contenant que des 1 est calculé en utilisant le solveur local basé sur une descente/remontée. Ce calcul est fait lors de la première itération de la puissance. Le coût de ce calcul (au maximum deux appels[38] au solveur local dans chaque sous-domaine) est amorti avec le nombre d'itérations de puissance.

Cette méthode peut se généraliser à des maillages non-coïncidents s'ils conservent le caractère creux des matrices $S^k_{i\to j}$. Si le maillage fin est inclus dans le maillage grossier avec un ratio r, cette matrice est bande de largeur égale à $2r^3 - 1$. L'adaptation aux maillages non-coïncidents ne requière plus de stocker $S^k_{i\to j}$ mais de décomposer cette matrice à l'aide des matrices de $Q_{i,j}$ (cf. page 52)

$$S^k_{i\to j} = {}^tC_{\Lambda^j_k\to j}(W^j)^{-1}C_{\Lambda^j_i\to j} = {}^tC_{j\to k}\epsilon_{j,k}Q_{j,k}(W^j)^{-1}{}^tQ_{i,j}\epsilon_{i,j}C_{i\to j},$$

et de stocker les matrices $\epsilon_{j,k}Q_{j,k}(W^j)^{-1}{}^tQ_{i,j}\epsilon_{i,j}$ qui sont diagonales.

[38] Un appel par interface.

5.3.1 Une implémentation efficace du préconditionneur

Comme les matrices $S^k_{i\to j}$ sont diagonales, les matrices P^{-1} et donc P sont également diagonales. Le préconditionneur bloc diagonal est donc un préconditionneur diagonal. Seul un vecteur d'interface est nécessaire pour stocker P. Pour chaque interface $\Gamma_{i,j}$, les contributions provenant de $S^i_{i\to j}$ et $S^j_{j\to i}$ sont additionnées pour obtenir P^{-1}. P est obtenue en inversant terme à terme[39] les éléments diagonaux de P. L'application de P (lignes 3 et 10 de l'**Algorithme 17**) est peu coûteuse car il s'agit de produit terme à terme entre deux vecteurs.

5.3.2 Une implémentation efficace du produit par la matrice d'interface

Pour calculer S_ω (ligne 6 de l'**Algorithme 17**), le produit de S par le vecteur ω qui est décomposé sur chaque partie $\Gamma_{i,j}$ de l'interface $\omega = {}^t(\omega^2_1, \omega^3_2, \ldots, \omega^K_{K-1})$, quatre contributions par interface sont à prendre en compte. Pour une interface Γ^r_l, entre un sous-domaine de gauche Ω_l et un sous-domaine de droite Ω_r, les quatres contributions de la composante $(S_\omega)^r_l$ du vecteur S_ω sont regroupées deux par deux. On explicite ces termes, en notant l' l'indice $l-1$ et r' l'indice $r+1$:

$$\begin{aligned}(S_\omega)^r_l &= S^l_{l\to l'}\omega^{l'}_l + \left(S^l_{l\to r} + S^r_{r\to l}\right)\omega^r_l + S^r_{r'\to r}\omega^{r'}_r \\ &= \left(S^l_{l\to l'}\omega^{l'}_l + S^l_{l\to r}\omega^r_l\right) + \left(S^r_{r\to l}\omega^r_l + S^r_{r\to r'}\omega^{r'}_r\right).\end{aligned}$$

L'expression de $(S_\omega)^r_l$ contient quatre termes correspondant aux quatre termes calculés. Le regroupement en deux termes correspond à l'influence des sous-domaines Ω_l et Ω_r. Les matrices étant stockées, le coût est linéaire par rapport à la taille des interfaces. Le coût de cette opération est donc essentiellement lié au coût des communications des vecteurs. La séparation suivant les influences de chacun des sous-domaines, correspond de fait aux vecteurs qui sont communiqués. Dans la partie concernant les schémas de communication, l'**Algorithme 18** décrit cette opération avec une vision centrée sur un sous-domaine et non sur une interface.

5.3.3 Re-calcul efficace de la solution locale

Avec le stockage des matrices $S^k_{i\to j}$, on évite un appel coûteux aux solveurs locaux $(W^i)^{-1}$ par itération de Gradient Conjugué. L'implémentation de l'**Algorithme 17** contient encore deux appels aux solveurs locaux (lignes 1 et 14). Afin de diminuer le coût du second appel (ligne 14), on le décompose en deux parties, dont une a déjà été calculée lors du calcul du résidu (ligne 1), et une qui prend en compte les modifications des multiplicateurs de Lagrange Λ (ligne 8) :

$$\hat{J} = \hat{W}^{-1}\left(\hat{F} - C\Lambda_0\right) + \hat{W}^{-1}C\Delta \quad \text{with } \Delta = \Lambda_0 - \Lambda.$$

Le vecteur $\tilde{J}_i$, défini comme la restriction du second terme ($\tilde{J} = \hat{W}^{-1}C\Delta$) au sous-domaine Ω_i, peut être calculé efficacement. En effet le vecteur $\tilde{J}_i$ se décompose en un terme prenant en compte les modifications du multiplicateur de gauche (Δ^l_i) et en un autre pour celles du multiplicateur de droite (Δ^r_i) :

$$\tilde{J}_i = \left((W^i)^{-1}C_{\Lambda^l_i\to i}\right)\Delta^l_i + \left((W^i)^{-1}C_{\Lambda^r_i\to i}\right)\Delta^r_i. \tag{5.7}$$

[39] Dans le cas de maillages non-coïncidents, cette opération peut être plus complexe. Si P est trop dense, il faut envisager un préconditionneur moins coûteux.

Pour chaque ligne de courant indicée par c, on note :
- $V_{|c}$ la restriction d'un vecteur de courant V sur la ligne c ;
- $\omega[c]$ la restriction d'un vecteur d'interface ω sur la ligne c.

On utilise des notations différentes car $V_{|c}$ est un vecteur et $\omega[c]$ un scalaire. On note $\mathbb{1}$, un vecteur d'interface ne contenant que des 1. Grâce à l'indépendance des lignes de courant, l'équation (5.7) est équivalente à

$$\forall c, \quad \tilde{J}_{i|c} = \Delta_i^l[c].\left((W^i)^{-1}C_{\Lambda_i^l \to i}\mathbb{1}\right)_{|c} + \Delta_i^r[c].\left((W^i)^{-1}C_{\Lambda_i^r \to i}\mathbb{1}\right)_{|c}. \tag{5.8}$$

Le calcul de $\hat{J}$ (ligne 14 de l'**Algorithme 17**) s'écrit donc dans chaque sous-domaine comme une combinaison linéaire de trois vecteurs dont les coefficients sont mis à jour pour chaque ligne de courant. Ce calcul est moins coûteux que l'appel aux solveurs locaux qui implique chacun une descente-remontée utilisant la factorisation de Cholesky d'une matrice W_d. A l'initialisation, il n'y a pas de surcoût en termes de calcul car les vecteurs $(W^i)^{-1}C_{\Lambda_i^j \to i}\mathbb{1}$ ont déjà été calculés pour la construction des matrices $S^i_{i\to j}$. Le surcoût se situe au niveau mémoire car pour chaque sous-domaine Ω_i, ces deux vecteurs ($((W^i)_i^{-1}C_{\Lambda_i^l \to i}\mathbb{1}$ et $(W^i)^{-1}C_{\Lambda_i^r \to i}\mathbb{1}$) ne sont *a priori* que des vecteurs temporaires ; désormais, ils sont stockés durant le déroulement de l'algorithme de la puissance.

5.4 Implémentation en mémoire distribuée

5.4.1 Distribution des données

Le partitionnement en sous-domaines et le stockage des données associées à chaque sous-domaine sont identiques à ceux de l'approche multigroupe (cf. page 67). Le domaine spatial Ω est partitionné en $K_x \times K_y \times K_z$ sous-domaines. Chaque processus stocke les données correspondant à son sous-domaine, pour chaque groupe d'énergie, pour chaque harmonique, et ce afin d'éviter de communiquer un champs spatial complet.

Dans le cas où l'on envisage de traiter des domaines avec géométrie non complétée (cf. page 26), ou avec des maillages non-coïncidents, l'équilibrage de la charge doit se faire dans chaque direction de l'espace. Dans la pratique, il suffit d'équilibrer la charge en considérant le nombre de degrés de liberté de flux. On note $n_x^i * n_y^i * n_z^i$ le nombre de degrés de liberté de flux pour le sous-domaine Ω_i. On suppose que Ω_i et Ω_j sont équilibrés en termes de nombre de degrés de liberté de flux : $n_x^i * n_y^i * n_z^i = n_x^j * n_y^j * n_z^j$. On cherche à évaluer l'équilibre dans la direction x des degrés de liberté de courant en évaluant le rapport suivant :

$$r = \frac{(n_x^i + 1).n_y^i.n_z^i}{(n_x^j + 1).n_y^j.n_z^j} = \frac{(1 + \frac{1}{n_x^i})}{(1 + \frac{1}{n_x^j})}.$$

Ainsi si l'on évite les sous-domaines de petite taille, le rapport r est proche de 1 : les sous-domaines sont donc équilibrés pour les courants dans chaque direction.

Pour les vecteurs d'interface, on opte pour la stratégie 3 (cf. page 67) généralement plus complexe à mettre en œuvre mais ici plus efficace. Celle-ci consiste à stocker chaque vecteur d'interface sur les deux sous-domaines voisins. Chaque vecteur d'interface[40] ainsi que les opérations sur ces vecteurs sont ainsi dupliqués. Toutefois, ce coût est compensé par la diminution des

[40] Le préconditionneur est également dupliqué.

communications. La figure FIG. 5.3 permet de comparer les communications engendrées par la stratégie 1 sans duplication des vecteurs d'interface et la stratégie 3 avec duplication des vecteurs d'interface. On remarque qu'avec la stratégie 3, on économise une communication[41] et que ces communications peuvent se recouvrir[42] (cf. annexe D). Un autre avantage de cette solution réside dans l'unicité du schéma de communication point à point car seules les matrices ${}^tC$ impliquent des communications point à point. On décrit plus en détails le schéma de communication dans le paragraphe suivant.

Cet algorithme est généralement plus délicat à mettre en œuvre car il faut vérifier que les vecteurs soient numériquement parfaitement équivalents. Ainsi, lorsque des opérations sont dupliquées, il faut faire attention au fait que tous les opérateurs ne sont plus associatifs. En effet de légères différences liées à des erreurs d'arrondi peuvent entraîner des effets de seuil différents dans deux sous-domaines. Dans notre cas, on n'est pas confronté à ces difficultés car les opérations dupliquées sont soit effectuées dans le même ordre (opération de type AXPY sur les vecteurs d'interface), soit elles ne concernent que deux vecteurs (somme de deux contributions provenant du produit par la matrice ${}^tC$).

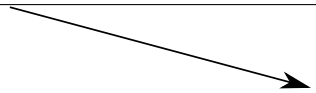
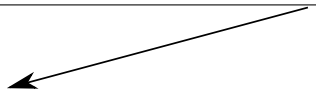
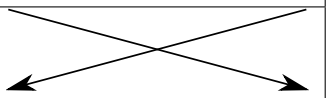
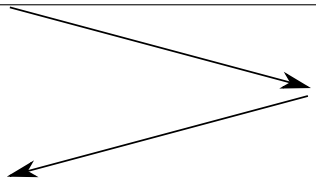
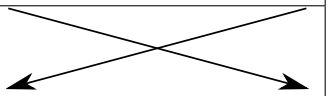
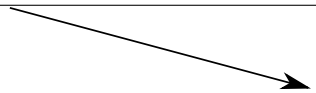
Ligne de l'**Algorithme 17**	Stratégie 1 (sans duplication) Ω_1 : maître esclave : Ω_2	Stratégie 3 (avec duplication) Ω_1 Ω_2
$1 : \hat{J} = \hat{W}^{-1}\left(\hat{F} - C\Lambda\right)$		
$2 : g = \left({}^tC\hat{J}\right)$		
$6 : S_w = \left({}^tC\hat{W}^{-1}C\right)w$		
$14 : \hat{J} = \hat{W}^{-1}\left(\hat{F} - C\Lambda\right)$		

FIG. 5.3 – Nombre de communications avec et sans duplication des vecteurs d'interface

5.4.2 Schéma de communication

Le schéma de communication est basé sur deux types de communication :
- les communications *point à point* ;
- les communications *globales*.

[41] Dans le cas de l'approche multigroupe, les algorithmes d'Uzawa ne nécessitent pas cette dernière communication.

[42] Dans le cas de l'approche multigroupe, les communications peuvent être en partie recouvertes par le calcul du produit par les matrices de couplage.

Algorithme 18 : Produit par S

exécuté par : processus i
entrée : ω_i^l et ω_i^r
sortie : $S\omega_i^l$ et $S\omega_i^r$
pour $n \in \{l, r\}$ **faire**
  Demande de réception asynchrone du processus n dans $tmpRecv_i^n$;
fin
pour $n \in \{l, r\}$ **faire**
  $S\omega_i^n = S_{i\to l}^n \omega_i^l + S_{i\to r}^n \omega_i^r$;
  $tmpSend_i^n = S\omega_i^n$;
  Envoi asynchrone de $tmpSend_i^n$ au processus n;
fin
pour $n \in \{l, r\}$ *dans l'ordre de réception* **faire**
  Attente de réception de $tmpRecv_i^n$;
  $S\omega_i^n$+=$tmpRecv_i^n$;
fin
Attendre que les données soient envoyées;

Algorithme 19 : Produit par ${}^tC$

exécuté par : processus i
entrée : J_i
sortie : ω_i^l et ω_i^r tel que $\omega = {}^tCJ$
pour $n \in \{l, r\}$ **faire**
  Demande de réception asynchrone du processus n dans $tmpRecv_i^n$;
fin
pour $n \in \{l, r\}$ **faire**
  $\omega_i^n = {}^tC_{\Lambda_n^i \to i} J_i$;
  $tmpSend_i^n = \omega_i^n$;
  Envoi asynchrone de $tmpSend_i^n$ au processus n;
fin
pour $n \in \{l, r\}$ *dans l'ordre de réception* **faire**
  Attente de réception de $tmpRecv_i^n$;
  ω_i^n+=$tmpRecv_i^n$;
fin
Attendre que les données soient envoyées;

Rappel : l désigne l'interface de gauche et r celle de droite du sous-domaine Ω_i.

▷ Les communications *point à point* sont nécessaires pour échanger des vecteurs d'interface. Avec le choix de dupliquer les vecteurs d'interface, les seules communications requises proviennent du produit par ${}^tC$ de **Algorithme 17** (lignes 2 et 6).

- Pour la ligne 2, l'**Algorithme 19** décrit cette opération au moyen de communications asynchrones ;
- Pour la ligne 6, l'**Algorithme 18** suit exactement le même schéma de communication ; seules les notations et les données calculées sont modifiées pour prendre en compte l'ensemble du produit par la matrice d'interface.

▷ Les communications *globales* sont nécessaires pour calculer des produits scalaires. Selon le type de vecteur pour lequel on calcule un produit scalaire, on est amené à considérer différentes familles de communicateurs :

- Le communicateur `CommWorld`[43] contient les $K_x \times K_y \times K_z$ processeurs. Ce communicateur est utilisé pour calculer le produit scalaire des itérations externes grâce à l'opération de réduction `MPI_Allreduce`. Les algorithmes de Gauss-Seidel (groupes d'énergie et harmoniques) peuvent impliquer des produits scalaires pour évaluer le critère de convergence[44]. L'algorithme de la puissance implique également de calculer des produits scalaires pour le calcul de la valeur propre, pour l'accélération de Tchebychev et enfin pour l'évaluation du critère de convergence ;
- Pour chaque sous-domaine, un communicateur par direction est créé. Pour un cas 2D, ces communicateurs sont appelés `CommX` et `CommY` (cf. figure Fig. 5.4). Comme les lignes

[43] Généralement `MPI_COMM_WORLD` est utilisé pour ce communicateur.
[44] Si le critère de convergence fixe le nombre d'itération à 1, il n'y a pas d'évaluation de produits scalaires.

de courant sont indépendantes, l'algorithme du Gradient Conjugué peut être appelé sur une ligne ou un groupe de lignes de courant. Ces différentes possibilités ne sont *a priori* pas équivalentes. On a décidé de regrouper ces lignes afin que les groupes respectent le partitionnement cartésien en sous-domaines (sur la figure Fig. 5.4, on observe 2 groupes de 2 lignes dans la direction x, et 3 groupes de 2 lignes dans la direction y). Cela permet de

- minimiser la taille des communicateurs. Le communicateur `CommX` contient K_y processus. Si l'on regroupait toutes les lignes, `CommX` serait égale à `CommWorld` et le coût des opérations de réduction en serait plus élevé ;
- minimiser la taille des données lors des communications globales : on ne communique que des données scalaires. Si l'on avait choisi des groupes ne contenant qu'une ligne de courant, les opérations de réduction ne s'effectueraient plus sur un scalaire mais sur un vecteur dont la taille est celle d'un vecteur d'interface. On n'observe pas de problèmes de convergence avec la solution actuelle. S'il l'on en observait, on pourrait envisager une solution intermédiaire consistant à choisir plusieurs groupes de lignes par ligne de sous-domaines.

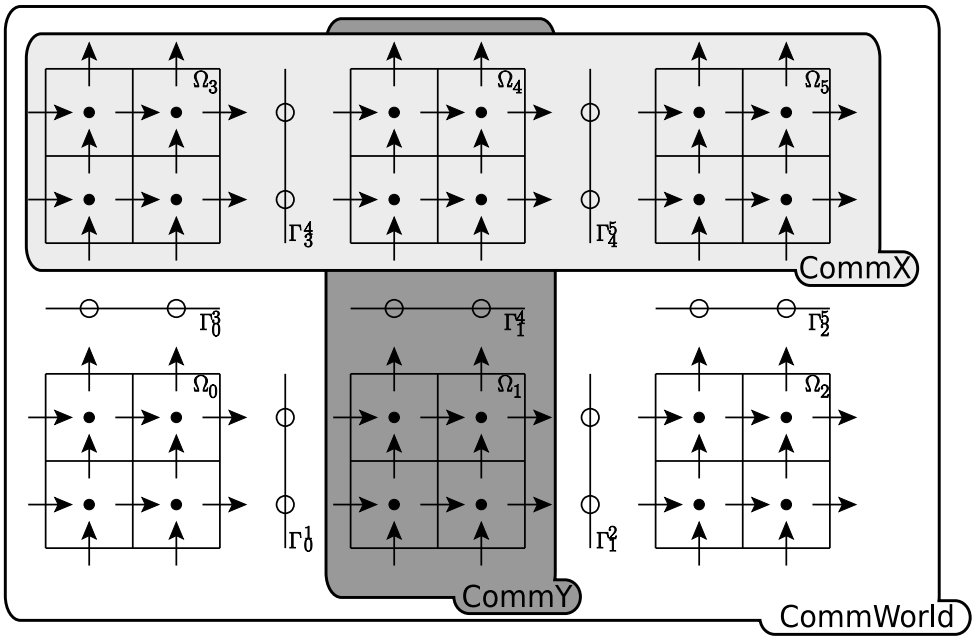


Fig. 5.4 – Communicateurs du sous-domaine Ω_4

5.4.3 Algorithmes itératifs imbriqués

Comparé à l'algorithme monodomaine, on ajoute une nouvelle boucle (cf. **Algorithme 10**) pour obtenir la convergence de l'**Algorithme 17**. En termes d'intégration dans le processus itératif imbriqué, il faut répondre à trois points :

1. comment initialise-t-on l'algorithme de Gradient Conjugué Préconditionné ?
2. comment implémente-t-on les produits matriciels des algorithmes externes ?
3. quel critère de convergence utilise-t-on pour le système d'interface ?

5.4.3.1 Initialisation du solveur d'interface

Avec l'approche unidimensionnelle, les solveurs locaux appelés par l'algorithme d'interface sont des solveurs directs ne nécessitant pas d'initialisation. Le seul problème à résoudre est donc l'initialisation du problème d'interface. Pour ce faire on utilise la même initialisation que pour l'approche multigroupe. On utilise la valeur du multiplicateur à l'itération précédente de la puissance. Comme dans le paragraphe 4.2.2.1, on applique également l'accélération de Tchebychev à cette valeur d'initialisation.

5.4.3.2 Implémentation des produits matriciels externes

Dans la première partie de ce chapitre, on a transformé algébriquement une méthode de décomposition de domaine sur l'ensemble du domaine spatial en plusieurs décompositions de domaine unidimensionnelles. Lors de l'application au problème multigroupe, on a aussi à paralléliser les produits matriciels externes. Il s'agit des produits par les termes extra-diagonaux des matrices multi-harmoniques et multigroupes. On rappelle donc dans un premier temps comment on parallélise un produit matriciel. On considère le produit suivant

$$\begin{pmatrix} S_0 \\ S_\Gamma \\ S_1 \end{pmatrix} = \begin{pmatrix} A_{0,0} & A_{\Gamma,0} & \\ A_{0,\Gamma} & A^0_{\Gamma\Gamma} + A^1_{\Gamma\Gamma} & A_{\Gamma,1} \\ & A_{1,\Gamma} & A_{11} \end{pmatrix} \begin{pmatrix} X_0 \\ X^0_\Gamma + X^1_\Gamma \\ X_1 \end{pmatrix}. \tag{5.9}$$

Pour réaliser ce produit, on effectue les produits locaux

$$\begin{pmatrix} s_0 \\ s^0_\Gamma \end{pmatrix} \begin{pmatrix} A_{0,0} & A_{\Gamma,0} \\ A_{0,\Gamma} & A^0_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} X_0 \\ X^0_\Gamma + X^1_\Gamma \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} s_1 \\ s^1_\Gamma \end{pmatrix} = \begin{pmatrix} A_{11} & A_{1,\Gamma} \\ A_{\Gamma,1} & A^1_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} X_1 \\ X^0_\Gamma + X^1_\Gamma \end{pmatrix}. \tag{5.10}$$

Puis on remarque que

$$\begin{pmatrix} S_0 \\ S_\Gamma \\ S_1 \end{pmatrix} = \begin{pmatrix} s_0 \\ s^0_\Gamma + s^1_\Gamma \\ s_1 \end{pmatrix}. \tag{5.11}$$

Plus généralement le produit matriciel est décomposé en quatre étapes :

1. communication des vecteurs X^0_Γ et X^1_Γ ;
2. calcul des produits locaux (5.10) ;
3. communication des vecteurs s^0_Γ et s^1_Γ ;
4. somme des vecteurs s^0_Γ et s^1_Γ .

Dans notre cas, le vecteur X est issue de la méthode de décomposition de domaine. La valeur à l'interface $(X^0_\Gamma + X^1_\Gamma)$ est donc déjà dans chaque sous-domaine ; la première étape a donc déjà été effectuée implicitement par l'algorithme de décomposition de domaine. Comme le résultat des produits matriciels sert de second membre à la méthode de décomposition de domaine, le terme sur l'interface est réparti pour moitié sur chaque sous-domaine. Les étapes 3 et 4 ne sont donc pas nécessaires. Au final, les produits par les matrices extra-diagonales ne nécessitent aucune communication supplémentaire.

5.4.3.3 Critère de convergence

On utilise la même stratégie que pour les autres boucles en fixant un nombre d'itération maximal pour l'algorithme du gradient conjugué et la convergence du processus global est contrôlée

par les itérations externes (la boucle de la puissance). Dans la suite, on note `GCPi` l'algorithme de Gradient Conjugué Préconditionné avec i itérations. Les études numériques montrent que pour le nombre de sous-domaines nous intéressant (inférieur à 200), l'algorithme `GCP1` est suffisant. S'il s'avérait nécessaire d'utiliser un nombre d'itérations plus élevé, cela ne serait pas très pénalisant car une itération supplémentaire ne requiert pas de calcul sur l'ensemble du sous-domaine. Si l'on souhaite néanmoins optimiser le nombre de communications, on peut envisager d'utiliser des critères de convergence différents par direction ou par groupe d'énergie ou par harmonique.

5.5 Applications numériques

Pour analyser les résultats numériques, on a utilisé la même méthodologie que pour l'approche multigroupe (cf. chapitre 4.4 page 69). Concernant l'analyse des résultats (hors performance), on a utilisé plus de sous-domaines (jusqu'à 961 sous-domaines), afin d'atteindre les limites de la méthode qui n'étaient pas atteintes avec 289 sous-domaines.

5.5.1 BenchMarkAIEA

On reporte sur la figure Fig. 5.5(a) le nombre d'itérations externes pour les algorithmes `GCP1` et `GCP2`. On observe un très bon comportement des deux algorithmes. En effet, entre 2 et 121 sous-domaines le nombre d'itérations de puissance est le même que celui de l'algorithme séquentiel monodomaine (78). Entre 144 et 575 sous-domaines, l'algorithme `GCP1` provoque une légère augmentation du nombre d'itérations externes, et au delà de 676 sous-domaines, l'algorithme de la puissance ne converge plus en moins de 300 itérations externes. L'utilisation de l'algorithme `GCP2` améliore le comportement de la puissance. De 2 à 575 sous-domaines, le comportement est identique à celui du solveur séquentiel. A partir de 625 sous-domaines, le nombre d'itérations externes augmente légèrement (inférieur à 12%).

Avec les algorithmes `GCP1` et `GCP2`, la précision obtenue est très satisfaisante (cf. figures Fig. 5.5(b) et Fig. 5.5(c)). En effet, la solution obtenue est toujours très proche de la solution de référence à l'exception de trois points concernant la valeur propre k_{eff}. Ces trois points ne sont pas problématiques car

- le solveur monodomaine est plus précis que nécessaire sur la valeur propre. Le critère d'arrêt limitant est celui sur le vecteur propre et non sur la valeur propre ;
- on n'observe pas cet écart pour le vecteur propre ;
- ces points correspondent à un grand nombre de sous-domaines (400, 529 et 625) ;
- avec `GCP2`, ces points disparaissent.

L'algorithme `GCP2` aboutit à une solution plus proche de la solution monodomaine que l'algorithme `GCP1`, sans pour autant obtenir une solution plus proche de la solution de référence. Ce résultat justifie *a posteriori* l'utilisation d'un algorithme itératif pour le système d'interface.

Comme jusqu'à 169 sous-domaines[45] les algorithmes `GCP1` et `GCP2` n'induisent pas une augmentation du nombre d'itérations externes, la comparaison de performance met en évidence le surcoût de calcul et de communication lié à une itération supplémentaire de l'algorithme de Gradient Conjugué. On reporte donc sur la figure Fig. 5.5(d) l'efficacité parallèle de ces deux algorithmes. Les résultats sont très bons :

- sur certains points l'efficacité est supérieure à 100% ;

[45] Comme il n'a pas été possible d'avoir l'ensemble du cluster à disposition, on n'est pas en mesure de présenter des résultats à 196 nœuds.

– le surcoût de `GCP2` par rapport à `GCP1` est faible et croissant avec le nombre de sous-domaines. En effet le rapport taille d'interface sur taille de sous-domaine est également croissant.

Les efficacités supérieures à 100% sont expliquées par la petite taille des sous-domaines permettant de tirer parti des effets de cache (ceux-ci ont été mis en évidence avec la figure Fig. 1.13). Les efficacités inférieures à l'idéal s'explique de la façon suivante :

– les opérations sur les vecteurs d'interface et pour la reconstruction des courants ont un coût faible mais non nul ;
– le fait d'utiliser deux nouveaux vecteurs pour reconstruire les courants, peut nuire aux effets de cache ;
– la duplication des degrés de liberté sur les interfaces génère un peu de calcul supplémentaire ;
– le partitionnement cartésien peut créer un léger déséquilibre de charge ;
– pour un nombre élevé de sous-domaines, les communications ne peuvent être négligées.

Il est difficile de faire la part entre ces différentes explications. Pour comprendre en détails les performances, il faudrait un modèle de coût prenant en compte de manière précise le cache. La figure Fig. 5.5(e), détaillant trois parties du solver pour le premier nœud :

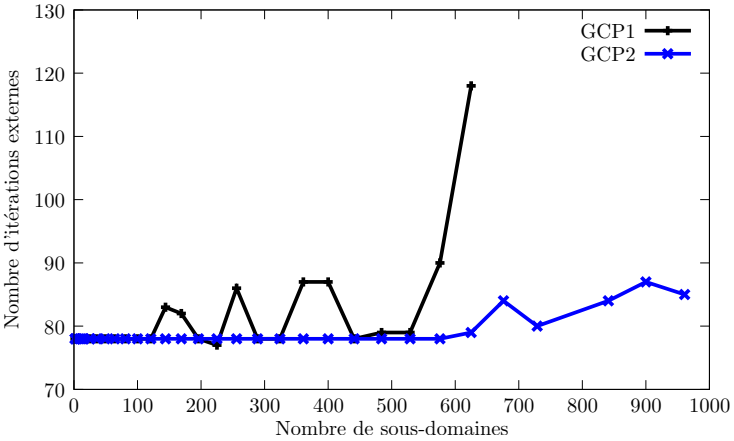
– les descentes-remontées sur les matrices W_d^0 factorisées ;
– le calcul du second membre du système (5.4) et le re-calcul des flux (5.2) ;
– les communications.

permet donc de mettre en évidence deux grandes tendances :

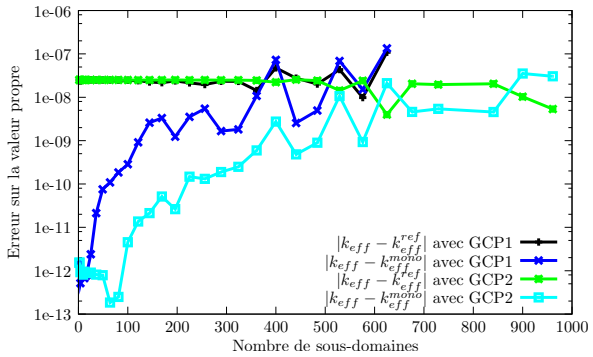
– à partir de 25 sous-domaines, le calcul du second membre impliquant des produits par les matrices B_d bénéficie d'effet de cache ;
– avec un grand nombre de sous-domaines, le temps des communications n'est plus négligeable.

Sur la figure Fig. 5.6, on reporte l'efficacité de l'algorithme `GCP1` pour les discrétisations SP1/RT0, SP3/RT0 et SP1/RT1. On ne présente pas de résultats pour le cas SP3/RT1 car on ne dispose pas de temps de référence monodomaine séquentiel. La moins bonne efficacité avec le cas SP1/RT1 s'explique par le fait que le problème est beaucoup plus gros ($\times 8$) et les sous-domaines trop gros pour bénéficier des effets de cache. On obtient une meilleure efficacité avec le cas SP3/RT0 car le couplage entre les systèmes de diffusion met en œuvre les matrices B_d qui bénéficient des effets de cache.

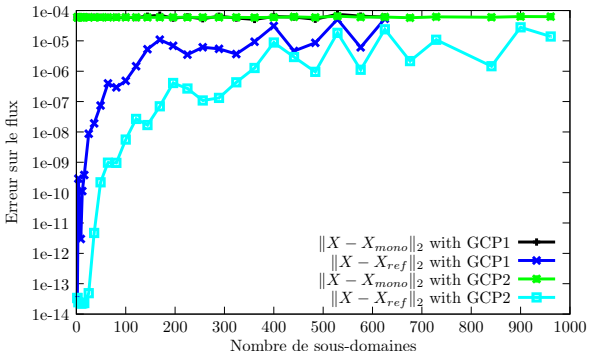
Ces résultats très encourageants demandent d'être confirmés sur des données provenant d'un cas industriel.



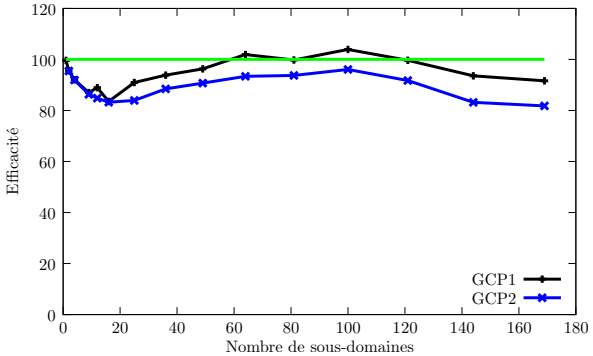
(a) Nombre d'itérations externes



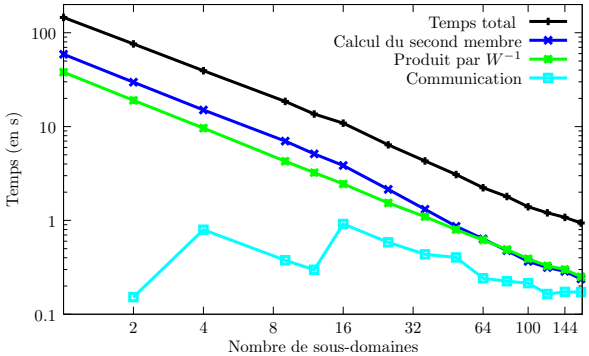
(b) Erreur sur k_{eff}



(c) Erreur sur le flux



(d) Efficacité parallèle



(e) Temps d'exécution détaillé pour l'algorithme `GCP1`

FIG. 5.5 – BenchMarkAIEA en SP1 RT0

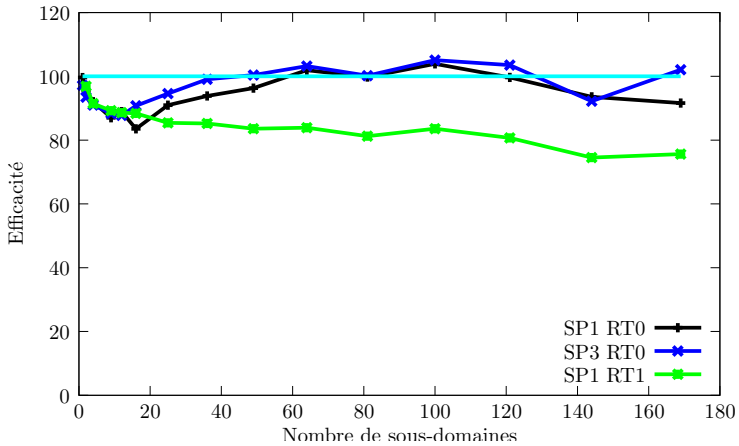


FIG. 5.6 – BenchMarkAIEA : efficacité parallèle pour différentes discrétisations

5.5.2 REP 900MW

Contrairement à l'approche multigroupe où le comportement diffère entre les cas tests, on observe également avec le REP 900MW un très bon comportement pour le nombre d'itérations externes (cf. figure FIG. 5.7(a)), pour la précision de la solution obtenue (cf. figures FIG. 5.7(b) et FIG. 5.7(c)) et pour les performances (cf. figures FIG. 5.7(d), FIG. 5.7(e) et FIG. 5.8). Sur ce cas ayant 40 cellules axiales, on peut utiliser des partitionnements en 2 et 4 parties suivant la direction z sans déséquilibre de charge. Les résultats avec 36 sous-domaines (respectivement 144 sous-domaines) sont reportés sur le tableau TAB. 5.1 (respectivement tableau TAB. 5.2) pour l'algorithme `GCP1` :

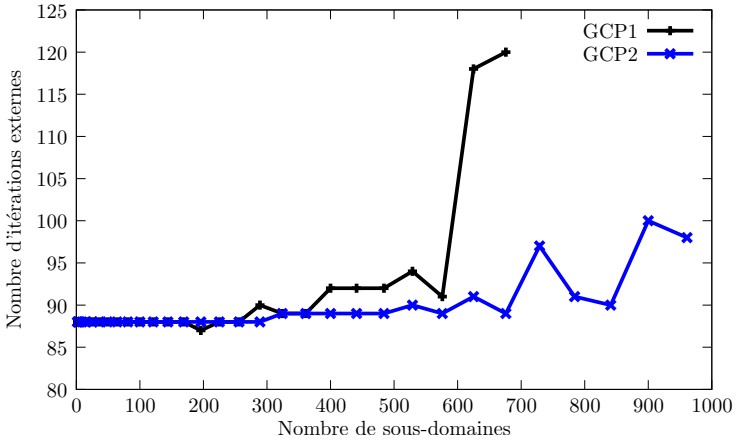
- le choix générique de partitionnement (décrit page 71) conduit sur ces deux cas au partitionnement le plus efficace ;
- les partitionnements unidimensionnels tel que $(36, 1, 1)$ font augmenter le nombre d'itérations externes ; ce type de partition est donc à proscrire. On observe également une légère perte de précision ;
- partitionner axialement ne diminue que faiblement les performances ;
- les différences de performance pour des partitions symétriques tel que $(9, 1, 4)$ et $(1, 9, 4)$ ne s'expliquent pas uniquement par le bruit des mesures. En effet l'algorithme de Gauss-Seidel crée des dissymétries par le choix de l'ordre de parcours des trois directions de l'espace.

partition	itérations externes	temps (en s)	$\lvert k_{eff} - k_{eff}^{ref} \rvert$	$\lVert X - X^{ref} \rVert_2$
(6, 6, 1)	88	$6,680$	$2,64.10^{-7}$	$1,63.10^{-3}$
(4, 9, 1)	88	$6,749$	$2,66.10^{-7}$	$1,63.10^{-3}$
(9, 4, 1)	88	$6,731$	$2,64.10^{-7}$	$1,62.10^{-3}$
(36, 1, 1)	830	$70,17$	$4,42.10^{-7}$	$2,38.10^{-3}$
(1, 36, 1)	826	$68,89$	$4,59.10^{-7}$	$2,42.10^{-3}$
(6, 3, 2)	88	$6,912$	$2,63.10^{-7}$	$1,63.10^{-3}$
(3, 6, 2)	88	$6,884$	$2,63.10^{-7}$	$1,63.10^{-3}$
(9, 1, 4)	88	$7,144$	$2,60.10^{-7}$	$1,62.10^{-3}$
(1, 9, 4)	88	$7,271$	$2,61.10^{-7}$	$1,62.10^{-3}$
(3, 3, 4)	88	$7,090$	$2,63.10^{-7}$	$1,63.10^{-3}$

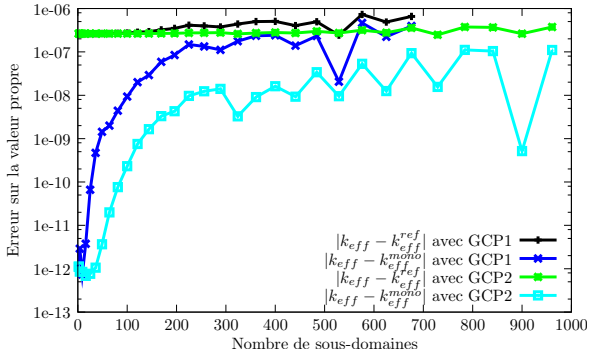
TAB. 5.1 – Influence du partitionnement : 36 sous-domaines

partition	itérations externes	temps (en s)	$\lvert k_{eff} - k_{eff}^{ref} \rvert$	$\lVert X - X^{ref} \rVert_2$
(12, 12, 1)	88	$1,548$	$2,92.10^{-7}$	$1,54.10^{-3}$
(8, 9, 2)	88	$1,588$	$2,60.10^{-7}$	$1,62.10^{-3}$
(9, 8, 2)	88	$1,606$	$2,59.10^{-7}$	$1,61.10^{-3}$
(6, 6, 4)	88	$1,695$	$2,63.10^{-7}$	$1,63.10^{-3}$

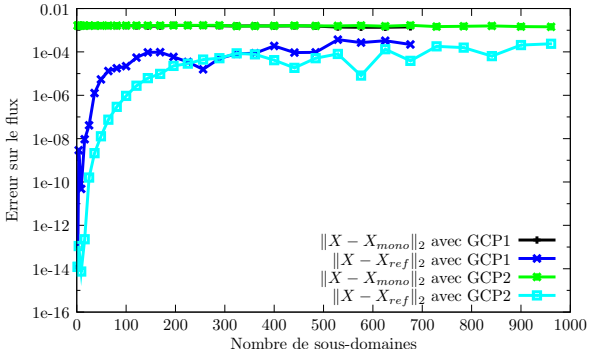
TAB. 5.2 – Influence du partitionnement : 144 sous-domaines



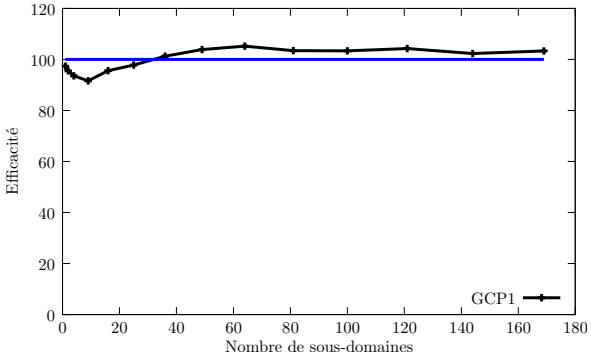
(a) Nombre d'itérations externes



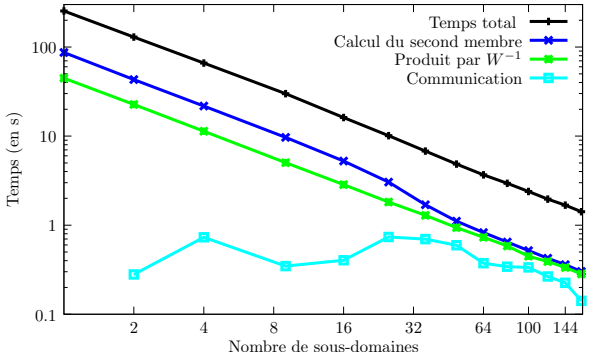
(b) Erreur sur k_{eff}



(c) Erreur sur le flux



(d) Efficacité parallèle



(e) Temps d'exécution détaillé pour l'algorithme `GCP1`

FIG. 5.7 – REP 900MW en SP1 RT0

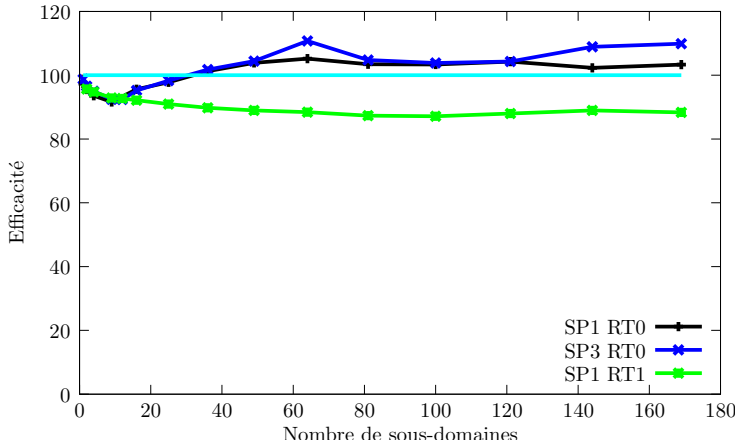


FIG. 5.8 – REP 900MW : efficacité parallèle pour différentes discrétisations

5.5.3 Bilan

Les résultats sont très satisfaisants. La stabilité du nombre d'itérations montre qu'il n'est pas nécessaire pour nos applications d'envisager un solveur direct ou un préconditionneur plus complexe. En effet, ceux-ci impliquent des communications supplémentaires, mais ne permettent pas de diminuer le nombre d'itérations externes.

L'approche multigroupe ne permet pas de se comparer favorablement à la méthode de Schwarz avec conditions de Robin [9, 73]. En revanche, l'approche unidimensionnelle ne souffre pas de la comparaison :

- on ne peut pas comparer exactement les performances car on ne peut pas faire tourner les mêmes cas tests et utiliser les mêmes machines. *A priori*, le volume de calculs et de communications sont du même ordre. On s'attend donc à des performances du même ordre ;
- il n'y a aucun paramètre à déterminer pour faire fonctionner la méthode correctement ;
- la méthode semble plus stable et plus précise. La méthode a été testée avec un plus grand nombre de sous-domaines sans observer une perte de précision ;
- il n'y a aucun recouvrement : la gestion des données est facilitée.

On note toutefois quelques inconvénients :

- la méthode est légèrement plus complexe à implémenter dans le cas d'un sous-domaine par processeur car elle nécessite l'ajout d'un solveur de Krylov. La méthode reste peu intrusive, car on réutilise le solveur interne W^{-1}. Cette réutilisation peut paraître anecdotique car il s'agit d'une *simple* descente-remontée sur la matrice factorisée. Or une implémentation performante de ce solveur dépend de l'architecture matérielle [74] ;
- la méthode est plus complexe à implémenter dans le cas de plusieurs sous-domaines par processeur. Celle-ci nécessitera de plus importantes modifications dans le code existant. Ces modifications sont beaucoup moins importantes lorsque le stockage et l'algorithme sont au niveau multigroupe ;
- lorsqu'on augmentera le nombre de groupes d'énergie, la granularité des calculs restera identique. L'efficacité parallèle sera *a priori* semblable à celle mesurée actuellement. Avec la méthode de Schwarz se plaçant au dessus des groupes, on peut espérer utiliser efficacement plus de processeurs sans être pénaliser par le coût des communications. Cet

inconvénient reste théorique car on ne sait pas comment va se comporter numériquement la méthode de Schwarz lorsque l'on augmentera le nombre de groupes d'énergie. La stabilité des résultats observés avec l'approche unidimensionnelle, laisse espérer un comportement sera proche de celui du solveur monodomaine séquentiel.

5.6 Vers une intégration au sein de la plate-forme *Cocagne*

Les résultats très encourageants obtenus avec l'approche unidimensionnelle nécessitent d'être confirmés au sein de la plate-forme *Cocagne*. En effet au sein de celle-ci, on ajoute des niveaux d'itérations supplémentaires, pour le couplage multi-physique et pour les applications métiers. Avant de mettre en œuvre une intégration performante au sein de la plate-forme, il s'agit de vérifier que l'impact de la décomposition de domaine sur les critères de convergence est faible. On présente donc dans cette section, un calcul *Métier* que l'on est actuellement incapable de passer sans décomposition de domaine, pour des ordres élevés d'éléments finis. Par la loi d'Amdahl, on sait avant implémentation que les performances du processus complet où seul le solveur SPn est parallélisé, sont modérées. L'objectif est donc de vérifier la faisabilité et les contraintes imposées par la parallélisation de la plate-forme en mémoire distribuée. Le reste de la plate-forme est d'un point de vue numérique simple à paralléliser :

- les contre-réactions et la génération des sections sont locales ;
- pour l'instant, le module de thermo-hydraulique est uniquement axial, il ne requiert donc pas de communications si on utilise des partitionnements radiaux.

5.6.1 Présentation du problème

On a choisi comme *procédure métier* une recherche de Bore critique [75] qui fait intervenir des itérations externes à la boucle de puissance et un couplage avec le module de thermo-hydraulique. L'objectif d'une recherche de Bore critique est d'ajuster la concentration en Bore du cœur afin d'obtenir un cœur critique ($k_{eff} = 1$). Cette concentration est obtenue par un algorithme itératif. Si le réacteur est sur-critique ($k_{eff} > 1$) la concentration en Bore est augmentée et elle est diminuée si le cœur est sous-critique ($k_{eff} < 1$). Ensuite la nouvelle concentration, et des températures du combustible et du modérateur (eau), on met à jour les sections servant d'entrées au solveur SPn. Le flux neutronique obtenu en sortie du solveur SPn sert au module de thermo-hydraulique pour calculer les températures combustible et modérateur. Le module de contre-réactions permet de calculer les sections en fonction des températures, et des concentrations des produits de fission (Xeon, Samarium, ...). Au sein de ce système itératif décrit sur la figure Fig. 5.9, l'algorithme de la puissance est bloqué à 15 itérations, et les boucles de couplage à 1 itération.

5.6.2 Description informatique

La plate-forme utilise essentiellement deux langages :

- Python un langage de haut niveau permettant à l'utilisateur de combiner des *procédures métier* ;
- C++ un langage de plus bas niveau, utilisé pour implémenter de manière performante les solveurs.

L'interface entre ces deux langages est réalisée par swig[46].

[46] http://www.swig.org/

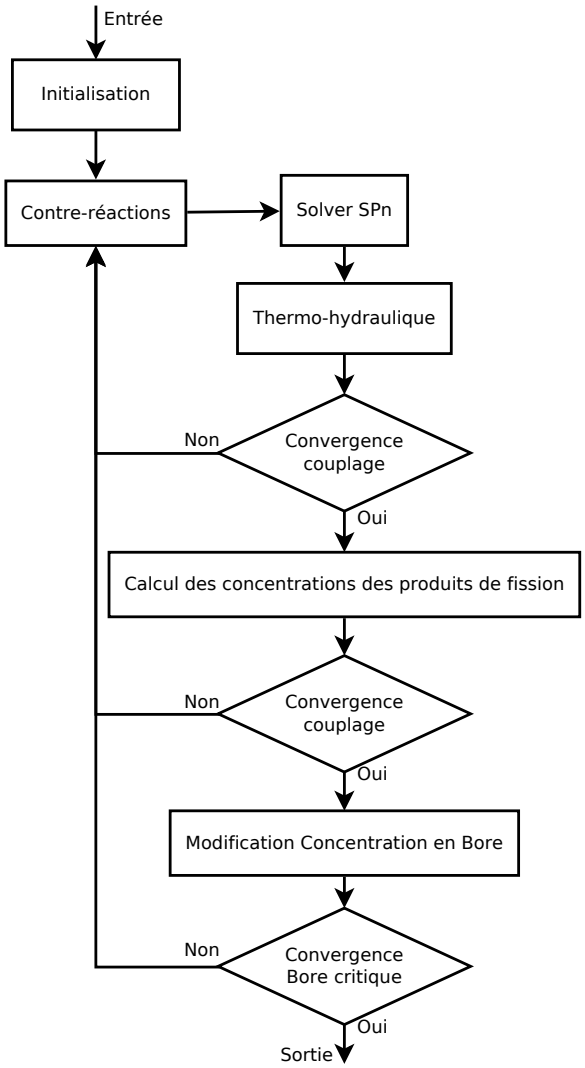


FIG. 5.9 – Recherche de Bore critique

Dans un premier temps, pour éviter d'impacter l'ensemble de la plate-forme, on a décidé d'intégrer le solveur SPn parallèle développé en conservant le reste de la plate-forme au sein d'un processus séquentiel. Lors de la première exécution du solveur SPn, on lance les processus MPI grâce à un appel système contenant la commande `mpirun`. L'ensemble de ces processus est appelé serveur MPI. Les sections d'entrée du solveur SPn sont transférées entre l'exécutable de la plate-forme et le premier processus du serveur MPI par fichiers. Ensuite, le premier processus MPI découpe les données et les distribue sur chacun des nœuds. Le calcul SPn parallèle peut alors être exécuté. Une fois le calcul fini le coefficient effectif de multiplication k_{eff} et le flux obtenus sont rapatriés sur le processus de la plate-forme par communication MPI puis par fichier. Lors du second appel au solveur SPn, on réutilise le serveur MPI lancé lors de la première exécution, afin d'économiser un appel à `mpirun` et de garder en mémoire les vecteurs permettant d'initialiser correctement les solveurs itératifs. On représente l'ensemble du flux de données sur la figure FIG. 5.10.

Les résultats présentés dans les sections 4.4 et 5.5 ont été obtenus grâce à l'implémentation faite sur les bases du module C++ du solveur SPn de la plate-forme Cocagne. Le travail fait

pour obtenir les résultats présentés dans cette section concerne essentiellement les problèmes de transferts de données du processus python au nœud 0 du serveur MPI, du découpage et envoi des données à chaque nœud MPI et le rapatriement des données au processus python. Une partie importante de ce travail a été effectuée par Hervé Ozdaba lors de son stage de fin d'étude [76].

5.6.3 Résultats

On a appliqué une recherche la Bore critique sur la campagne Gravelines 514 avec des données initialement crayon par crayon. Ce cas est donc plus complexe que celui décrit page 30 car il y a plus d'hétérogénéités dans les sections d'entrée du solveur SPn. On reporte dans le tableau Tab. 5.3 pour 1, 4, 36 et 100 processeurs et les éléments finis RT0, RT1 et RT2 :

- la concentration en Bore ;
- le temps total ;
- le temps passé dans le solveur SPn (incluant les transferts de données) ;
- le temps de la partie séquentielle de la procédure.

Dans un premier temps, on observe que le résultat (la concentration en Bore) ne dépend pas du nombre de processeurs, confirmant ainsi la validité de la méthode. En RT0, les facteurs d'accélération sont faibles, car on ne parallélise que 16% du code ; la loi d'Amdahl permet donc de majorer le facteur d'accélération par 1.19. Si l'on considère juste la partie du solveur SPn le facteur d'accélération reste limité (1.5 pour 4 nœuds). En effet le solveur SPn parallèle prend en compte les différents coûts du passage des données du processus séquentiel aux processus parallèles :

- le coût du transfert par fichier du processus de la plate-forme au nœud 0 du serveur MPI ;
- le coût du découpage des sections ;
- le coût de transfert par MPI des sections sur chaque nœud MPI ;
- le coût de rapatriement des flux par MPI ;
- le coût de rapatriement des flux dans la plate-forme par fichier.

En augmentant l'ordre des éléments finis, ces coûts de transfert et la durée de la partie séquentielle restent constants. On obtient donc une meilleure efficacité parallèle en RT1 et en RT2. Cette efficacité est d'autant meilleure en RT1, car le code séquentiel utilise le *swap*. En RT2 on n'est pas capable de passer ce calcul sur cette machine. Dans [77], sur une machine de 32 Gigas et avec un autre calcul *métier* (une longueur de campagne) le solveur SPn représente 14% du temps total en RT0, 57% en RT1 et 84% en RT2.

Avec cette intégration *naïve*, on a montré la pertinence au niveau numérique de la méthode. En l'état, elle permet de passer des cas trop gourmands en mémoire. En revanche les facteurs d'accélération du processus global restent limités. Ces résultats constituent un premier pas important en vue de la parallélisation de l'ensemble de la plate-forme permettant de tirer parti de la performance de l'approche unidimensionnelle :

- en réduisant la partie séquentielle, *via* la parallélisation de chaque module (contre-réactions, thermo-hydraulique, ...) ;
- en supprimant les temps de transfert à chaque appel aux solveurs.

élément fini	nombre de processeurs	concentration en Bore	temps total	temps SPn	temps partie séquentielle
RT0	1	1457.3	1488	241	1247
RT0	4	1457.3	1394	162	1231
RT0	36	1457.3	1344	102	1241
RT0	100	1457.3	1334	99	1235
RT1	1	1458.7	16974	14123	2850
RT1	4	1458.7	1755	532	1222
RT1	36	1458.7	1405	163	1241
RT1	100	1458.7	1369	119	1250
RT2	1	X	X	X	X
RT2	4	1459.1	2837	1597	1240
RT2	36	1459.1	1804	571	1232
RT2	100	1459.1	1554	315	1238

TAB. 5.3 – Recherche de Bore critique : résultats

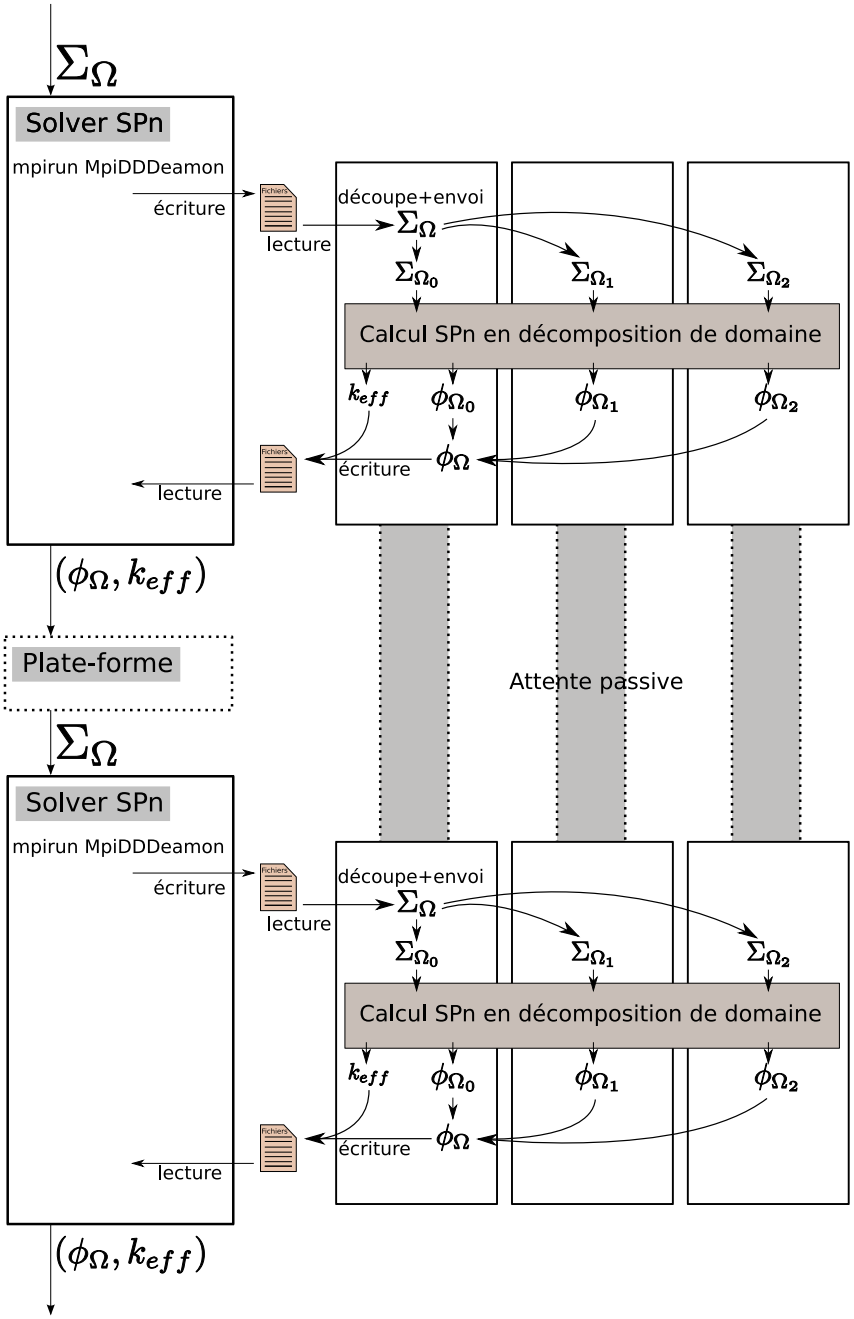


FIG. 5.10 – Intégration du solveur SPn parallèle dans la plate-forme *Cocagne* séquentielle

Conclusion et perspectives

Conclusion

La résolution d'un problème de criticité conduit à la résolution d'un problème aux valeurs propres généralisées. Ce problème est résolu par l'algorithme des puissances inverses basé sur la résolution approchée d'un problème matriciel issu des différentes approximations (en énergie, en angle et en espace). L'algorithme complet comprend ainsi quatre algorithmes itératifs imbriqués.

Afin de résoudre des problèmes mémoire et de temps de calcul, on a étudié au cours de cette thèse une méthode de décomposition de domaine de type Schur Dual au sein de ce système d'itérations imbriquées. Cette méthode nécessite la résolution d'un système de point-selle. Plusieurs approches pour le placement de la résolution du système de point-selle étaient envisageables. Deux d'entre elles ont été implémentées et analysées.

Avec l'approche multigroupe, on a privilégié la simplicité d'implémentation et la flexibilité. Le système de point-selle est résolu par l'algorithme d'Uzawa-MR qui tolère l'utilisation de solveurs approchés dans chaque sous-domaine. Les résultats, satisfaisants sur un cas académique, se sont avérés décevants pour des cas industriels. En effet l'augmentation du nombre d'itérations externes (algorithme de la puissance) pénalise fortement la méthode : l'efficacité parallèle obtenue est de l'ordre de 5%. Pour contourner ces difficultés, une voie naturelle aurait été la mise en place d'un préconditionneur. Celui-ci mettant à mal l'argument de simplicité en faveur de cette approche, on s'est tourné vers l'approche unidimensionnelle.

Avec l'approche unidimensionnelle, l'accent a été plus porté sur la performance. Cette approche utilise les spécificités des éléments de Raviart-Thomas et de l'algorithme de directions alternées pour obtenir une implémentation performante. En se plaçant sous les directions alternées, le système d'interface symétrique a un profil très creux permettant son stockage évitant ainsi des appels aux solveurs locaux. Suite des optimisations algorithmiques, on obtient un algorithme ne faisant appel qu'une seul fois aux solveurs locaux par itération de puissance, par groupe, par harmonique et par direction. Les résultats sont très encourageants, car le nombre d'itérations externes n'augmente pas par rapport au solveur monodomaine. Un bon comportement numérique associé à un algorithme économe en appel aux solveurs locaux permet d'obtenir une efficacité parallèle proche de 100% et parfois même supérieure grâce à des effets de cache.

L'algorithme obtenu avec l'approche unidimensionnelle possède beaucoup de qualités en vue d'une intégration dans la plate-forme industrielle. La méthode ne demande aucun nouveau paramètre, ou de modifier des paramètres existants. Comme il n'y a pas de recouvrement entre sous-domaines, on peut envisager une intégration sans redistribution de données à chaque appel au solveur SPn. Afin que l'utilisateur final puisse profiter des facteurs d'accélération obtenus sur le solveur SPn, il est nécessaire de développer un modèle de données parallèle permettant de paralléliser le reste de la plate-forme sans redistribution de données.

Perspectives

La méthode ayant de bonnes propriétés, il n'est pas *a priori* complexe d'un point de vue numérique de mettre en place un flux de données parallèle sur l'ensemble de la plate-forme. Dans la pratique cette tâche est rendue complexe par le nombre d'intervenants travaillant sur la plate-forme et ne souhaitant pas être impactés par un modèle de données parallèle. Un important travail de génie logiciel reste donc à faire pour conserver un code pérenne. Une contrainte supplémentaire pour le modèle de donnée parallèle, réside dans le fait qu'il doit être compatible avec les nouveaux solveurs neutroniques qui vont être intégrés dans la plate-forme :

- à court terme, le solveur Sn appelé Domino [78] ;
- à plus long terme un solveur Sn hétérogène [79], prenant en compte la géométrie des crayons (combustible, gaine ...).

L'algorithme a été pensé avec pour cible les machines à mémoire distribuée. Il sera intéressant de comparer cette approche implémentée en mémoire partagée avec une implémentation[47] utilisant l'indépendance des lignes de courants. Sur des machines à fort effet NUMA, notre nouvelle approche pourrait s'avérer plus efficace car l'algorithme respecte mieux la localité des données. Pour ce faire, il faut implémenter la possibilité de définir plusieurs sous-domaines par processus. Cette fonctionnalité est plus intrusive que l'implémentation actuelle, mais offrira plus de possibilités en termes d'équilibrage de charge et de flexibilité dans le découpage en sous-domaines. Ainsi, il sera possible de gérer une géométrie non complétée[48] en utilisant un nombre de processeurs inférieur au nombre d'assemblages combustible et réflecteur (241).

Selon la performance de l'algorithme en mémoire partagée, il se posera la question d'une implémentation hybride mémoire distribuée (MPI) - mémoire partagée (TBB). En effet, il s'agit de savoir si il est plus performant d'associer un sous-domaine par thread, ou d'utiliser un parallélisme de grain fin avec plusieurs threads par sous-domaine utilisant le parallélisme sur les lignes de courants. Par ailleurs, une implémentation sur GPU [80] a permis d'obtenir un facteur d'accélération de l'ordre de 30 en utilisant un parallélisme de grain fin. Comme la quantité de mémoire sur un GPU reste actuellement limitée, une implémentation hybride (MPI-GPU) permettrait de tirer parti de ce facteur d'accélération en contournant les problèmes mémoire.

La méthode obtenue fonctionnant très bien du point de vue numérique, l'essentiel des perspectives se situe donc au niveau informatique. Néanmoins deux points restent en suspens. Dans un premier temps, il faudra vérifier que l'algorithme fonctionne aussi bien dans le cadre des équations cinétiques. Dans un second temps, l'extension aux maillages non-coïncidents, qui a été mise de coté au cours de cette thèse, reste envisageable. La poursuite de ce point ne peut se penser indépendamment des questions d'intégration au sein de la plate-forme, car son impact sur le modèle de données peut être considérable. On ordonne ces deux points ainsi car on pense que l'utilisation de maillages non-coïncidents sera plus pertinente dans le contexte cinétique.

[47] Implémentation développée parallèlement à cette thèse en interne EDF. Celle-ci utilise les TBB.
[48] Cf. page 26.

Annexe A

Outils mathématiques

A.1 Polynômes de Legendre

Définition

Les polynômes de Legendre sont définis par l'expression suivante :

$$\forall l \in \mathbb{N} \quad P(\mu) = \frac{1}{2^l l!} \frac{d^l}{d\mu^l} \left(\mu^2 - 1\right)^2 .$$

Les premiers polynômes sont donc :

$$\begin{array}{l} P_0(\mu) = 1 \,, \\ P_1(\mu) = \mu \,, \\ P_2(\mu) = \frac{1}{2}(3\mu^2 - 1) \,. \end{array} \tag{A.1}$$

Propriétés

Relation de récurrence

Les polynômes de Legendre vérifient la relation de récurrence suivante :

$$\mu P_n(\mu) = \frac{n+1}{2n+1} P_{n+1}(\mu) + \frac{n}{2n+1} P_{n-1}(\mu) \text{ pour } n \geq 1 \,. \tag{A.2}$$

On remarque qu'avec la convention $P_{-1} = 0$, la relation reste valable pour $n = 0$.

Base orthogonale

Comme P_l est un polynôme de degré l, l'ensemble $\{P_l \quad | \quad 0 \leq l \leq n\}$ forme une base de $\mathbb{R}n[X]$. De plus, les éléments de cette base vérifient la propriété d'orthogonalité suivante :

$$\int_{-1}^{1} P_l(\mu) P_{l'}(\mu) d\mu = \frac{2}{2l+1} \delta_{l,l'} \,, \tag{A.3}$$

où $\delta_{l,l'}$ est le symbole de Kronecker qui vaut 1 si $l = l'$, 0 sinon. A partir de (A.3), on peut déduire le cas particulier suivant :

$$\int_{-1}^{1} P_l(\mu) d\mu = 2\delta_{l,0}. \tag{A.4}$$

Développement sur une base de polynômes de Legendre

Un polynôme ψ de degré N peut se décomposer sur la base des polynômes de Legendre ainsi :

$$\psi(\mu) = \sum_{n=0}^{N} \frac{2n+1}{2} \psi_n P_n(\mu)$$

avec

$$\psi_n = \int_{-1}^{+1} \psi(\mu) P_n(\mu) d\mu. \tag{A.5}$$

A.2 Intégration de Gauss-Legendre

Points et poids de Gauss

Les k points de Gauss $\{x_i \,|\, i \in [\![0, k-1]\!]\}$ sont définis comme les racines du polynôme de Legendre d'ordre k. Les poids de Gauss $\{w_i \mid i \in [\![0, k-1]\!]\}$ sont définis par la relation :

$$w_i = \frac{-2}{(n+1) P_n'(x_i) P_{n+1}(x_i)}.$$

On donne les points et poids de Gauss pour un faible nombre de points :

Nombre de points : k	Points de Gauss : $\{x_i \mid i \in [\![0, k-1]\!]\}$	Poids de Gauss : $\{w_i \mid i \in [\![0, k-1]\!]\}$
1	$\{0\}$	$\{2\}$
2	$\left\{-\sqrt{1/3}, \sqrt{1/3}\right\}$	$\{1, 1\}$
3	$\left\{-\sqrt{3/5}, 0, \sqrt{3/5}\right\}$	$\{5/9, 8/9, 5/9\}$

Intégration de Gauss-Legendre

On peut calculer numériquement l'intégrale d'une fonction par une somme pondérée par les poids de Gauss des valeurs de la fonction aux points de Gauss :

$$\int_{-1}^{1} f(x) dx \approx \sum_{i \in [\![0, k-1]\!]} w_i f(x_i).$$

Si f est un polynôme de degré inférieur ou égale à $2k-1$, ce calcul est exact.

Polynômes de Lagrange

Pour un ensemble de k points $(x_i)_{i \in [\![0, k-1]\!]}$, on définit k polynômes de Lagrange $(l_i)_{i \in [\![0, k-1]\!]}$ de degré k :

$$l_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}.$$

Par définition on a donc :

$$l_i(x_j) = \delta_{i,j}.$$

On considère des polynômes de Lagrange définis aux points de Gauss. En appliquant l'intégration de Gauss-Legendre avec $k+1$ points aux polynômes de Lagrange de degré k, on obtient :

$$\int_{-1}^{1} l_i(x) l_j(x) dx = \sum_{m \in [\![0, k]\!]} w_m l_i(x_m) l_j(x_m) = \sum_{m \in [\![0, k]\!]} w_m \delta_{i,m} \delta_{j,m} = \delta_{i,j} w_i \tag{A.6}$$

Annexe B

Les équations de transport simplifié SPn

Les équations SPn (n impair est noté par la suite N) sont dérivées de l'équation du transport neutronique en faisant l'hypothèse d'une solution localement plane et en décomposant le flux sur une base de polynômes de Legendre. Dans cette annexe, on présente rapidement les équations SPn pour montrer comment on se ramène à $(N+1)/2$ systèmes de diffusion couplés. Le lecteur souhaitant plus de détails peut se référer à [2, 7, 81, 82].

Considérons l'équation multigroupe dans le cas d'un calcul critique :

$$\begin{aligned}\operatorname{div}\left(\overrightarrow{\Omega}\psi^g(\overrightarrow{r},\overrightarrow{\Omega})\right)+\Sigma_t^g(\overrightarrow{r})\psi^g(\overrightarrow{r},\overrightarrow{\Omega}) &= \sum_{g'}\left(\int_0^{4\pi}\Sigma_s^{g'\to g}\left(\overrightarrow{r},\overrightarrow{\Omega}'.\overrightarrow{\Omega}\right)\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\right)\\ &\quad+\frac{1}{\lambda}\left(\frac{\mathcal{X}^g}{4\pi}\right).\left(\sum_{g'}\int_0^{4\pi}\nu\Sigma_f^{g'}(\overrightarrow{r})\psi^{g'}(\overrightarrow{r},\overrightarrow{\Omega}')\,d\overrightarrow{\Omega}'\right).\end{aligned}$$

Supposons que la solution soit localement plane suivant la direction x et posons $\mu=\overrightarrow{\Omega}\cdot\overrightarrow{x}$. On obtient l'équation de tranport 1D suivante :

$$\begin{aligned}\mu\frac{\partial}{\partial x}\,\psi^g(x,\mu)+\Sigma_t^g(x)\,\psi^g(x,\mu) &= 2\pi\sum_{g'}\int_{-1}^{+1}\Sigma_s^{g'\to g}(x,\mu.\mu')\,\psi^{g'}(x,\mu')d\mu'\\ &+ \frac{1}{\lambda}.\left(\frac{\chi^g}{2}\right)\left(\sum_{g'}\int_{-1}^{+1}\nu\Sigma_f^{g'}(x)\psi^{g'}(x,\mu')d\mu'\right). \qquad \text{(B.1)}\end{aligned}$$

Développons à l'ordre N (respectivement M) le flux (respectivement les sections efficaces de scattering[49]) sur la base des polynômes de Legendre :

$$\psi^g(x,\mu) = \sum_{n=0}^{N}\frac{2n+1}{2}\psi_n^g(x)P_n(\mu); \qquad \text{(B.2)}$$

$$\Sigma_s^{g\to g'}(x,\mu.\mu') = \sum_{n=0}^{M}\frac{2n+1}{4\pi}\Sigma_{s,n}^{g\to g'}(x)P_n(\mu)\cdot P_n(\mu'). \qquad \text{(B.3)}$$

[49] On développe très souvent les sections efficaces de scattering à un ordre moins élevé que le flux. Le détail de ce développement se trouve dans [82], en effet celui-ci dépend de l'approximation *localement plane*.

On obtient alors :

$$\begin{aligned}
&\sum_{n=0}^{N} \frac{2n+1}{2} \left(\mu P_n(\mu) \frac{\partial}{\partial x} \psi_n^g(x) + \Sigma_t^g(x) \, \psi_n^g(x) P_n(\mu) \right) \\
&\quad = \sum_{n=0}^{M} \frac{2n+1}{2} P_n(\mu) \left(\sum_{g'} \Sigma_{s,n}^{g \to g'}(x) \int_{-1}^{+1} P_n(\mu') \, \psi^{g'}(x, \mu') d\mu' \right) \\
&\quad + \frac{1}{\lambda} . \left(\frac{\chi^g}{2} \right) \left(\sum_{g'} \nu \Sigma_f^{g'}(x) \sum_{n=0}^{N} \frac{2n+1}{2} \psi_n^{g'}(x) \int_{-1}^{+1} P_n(\mu') d\mu' \right) .
\end{aligned} \tag{B.4}$$

Dans (B.4), après une multiplication par 2, on applique :

- à la ligne 1, la formule de récurrence (A.2) pour supprimer le terme $\mu P(\mu)$;
- à la ligne 2, on applique (B.2) ;
- à la ligne 3, on applique (A.5).

$$\begin{aligned}
&\sum_{n=0}^{N} \left(((n+1) P_{n+1}(\mu) + P_{n-1}(\mu)) \, \frac{\partial}{\partial x} \psi_n^g(x) + (2n+1) \Sigma_t^g(x) \, \psi_n^g(x) P_n(\mu) \right) \\
&\quad = \sum_{n=0}^{M} (2n+1) P_n(\mu) \left(\sum_{g'} \Sigma_{s,n}^{g \to g'}(x) \psi_n^{g'}(x) \right) + \frac{\chi^g}{\lambda} . \left(\sum_{g'} \nu \Sigma_f^{g'}(x) \psi_0^{g'}(x) \right) .
\end{aligned} \tag{B.5}$$

Dans (B.5) on effectue des renumérotations sur les indices de sommation :

$$\begin{aligned}
&\sum_{n=1}^{N} n P_n(\mu) \frac{\partial}{\partial x} \psi_{n-1}^g(x) + \sum_{n=0}^{N-1} (n+1) P_n(\mu) \frac{\partial}{\partial x} \psi_{n+1}^g(x) + \sum_{n=0}^{N} (2n+1) \Sigma_t^g(x) \, \psi_n^g(x) P_n(\mu) \\
&\quad = \sum_{n=0}^{M} (2n+1) P_n(\mu) \left(\sum_{g'} \Sigma_{s,n}^{g \to g'}(x) \psi_n^{g'}(x) \right) + \frac{\chi^g}{\lambda} . \left(\sum_{g'} \nu \Sigma_f^{g'}(x) \psi_0^g(x) \right) .
\end{aligned} \tag{B.6}$$

Par identification sur la base des polynômes de Legendre, on obtient :

$$\left\{ \begin{array}{l}
\Sigma_t^g(x) \, \psi_0^g(x) + \dfrac{\partial}{\partial x} \psi_1^g(x) = \left(\displaystyle\sum_{g'} \Sigma_{s,0}^{g \to g'}(x) \psi_0^{g'}(x) \right) + \dfrac{\chi^g}{\lambda} . \left(\displaystyle\sum_{g'} \nu \Sigma_f^{g'}(x) \psi_0^{g'}(x) \right) \\
\\
\forall n \in [\![1, N-1]\!], \\
\quad \dfrac{n}{2n+1} \dfrac{\partial}{\partial x} \psi_{n-1}^g(x) + \Sigma_t^g(x) \, \psi_n^g(x) + \dfrac{n+1}{2n+1} \dfrac{\partial}{\partial x} \psi_{n+1}^g(x) = \delta_{n \leq M} \left(\displaystyle\sum_{g'} \Sigma_{s,n}^{g \to g'}(x) \psi_n^{g'}(x) \right) , \\
\\
\dfrac{N}{2N+1} \dfrac{\partial}{\partial x} \psi_{N-1}^g(x) + \Sigma_t^g(x) \, \psi_N^g(x) = \delta_{M,N} \left(\displaystyle\sum_{g'} \Sigma_{s,M}^{g \to g'}(x) \psi_M^{g'}(x) \right)
\end{array} \right. \tag{B.7}$$

avec $\delta_{i \leq j} = 1$ si et seulement si $i \leq j$, 0 sinon.

On passe les termes correspondant au groupe d'énergie g dans le membre de gauche en utilisant la notation $\Sigma_{a,n}^g(x) = \Sigma_t^g(x) - \Sigma_{s,n}^{g \to g}(x)$. Pour obtenir un problème 3D à partir de (B.7), on multiplie les équations impaires par le vecteur unité $\overrightarrow{x}$ et on remplace les opérateurs

de dérivés partielles par les opérateurs 3D correspondant, à savoir div et $\overrightarrow{grad}$. Grâce aux notations suivantes :

$$\left|\begin{array}{ll}
S_0^g(x) = \left(\sum_{g'\neq g} \Sigma_{s,0}^{g\to g'}(x)\psi_0^{g'}(x)\right) + \frac{\chi^g}{\lambda}.\left(\sum_{g'}\nu\Sigma_f^{g'}(x)\psi_0^{g'}(x)\right) & \\
S_l^g(x) = \delta_{2l\leq M}\left(\sum_{g'\neq g} \Sigma_{s,2l}^{g\to g'}(x)\psi_{2l}^{g'}(x)\right) & \forall l \in [\![1, \frac{N-1}{2}]\!] \\
\overrightarrow{S_l^g}(x) = \delta_{2l+1\leq M}\left(\sum_{g'\neq g} \Sigma_{s,2l+1}^{g\to g'}(x)\psi_{2l+1}^{g'}(x)\right) & \forall l \in [\![0, \frac{N-1}{2}]\!] \\
 & \\
\phi_l^g = \psi_{2l}^g & \forall l \in [\![0, \frac{N-1}{2}]\!] \\
\overrightarrow{J_l^g} = \psi_{2l+1}^g \overrightarrow{x} & \forall l \in [\![0, \frac{N-1}{2}]\!]
\end{array}\right. ,$$

on obtient,

$$\left\{\begin{array}{ll}
\Sigma_{a,0}^g(x)\phi_0^g(x) + \mathrm{div}\,\overrightarrow{J_0^g}(x) = S_0^g(x) & \\
 & \\
\frac{1}{3}\overrightarrow{grad}\,\phi_0^g(x) + \Sigma_{a,1}^g(x)\overrightarrow{J_0^g}(x) + \frac{1}{3}\overrightarrow{grad}\,\phi_1^g(x) = \overrightarrow{S_0^g}(x) & \\
 & \\
\frac{2l}{4l+1}\mathrm{div}\,\overrightarrow{J_{l-1}^g}(x) + \Sigma_{a,2l}^g(x)\phi_l^g(x) + \frac{2l+1}{4l+1}\mathrm{div}\,\overrightarrow{J_l^g}(x) = S_l^g(x) & \forall l \in [\![1, \frac{N-1}{2}]\!] \\
\frac{2l+1}{4l+3}\overrightarrow{grad}\,\phi_l^g(x) + \Sigma_{a,2l+1}^g(x)\overrightarrow{J_l^g}(x) + \frac{2l+2}{4l+3}\overrightarrow{grad}\,\phi_{l+1}^g(x) = \overrightarrow{S_l^g}(x) & \forall l \in [\![1, \frac{N-3}{2}]\!] \\
 & \\
\frac{N}{2N+1}\overrightarrow{grad}\,\phi_{\frac{N-1}{2}}^g(x) + \Sigma_{a,N}^g(x)\overrightarrow{J_{\frac{N-1}{2}}^g}(x) = \overrightarrow{S_{\frac{N-1}{2}}^g}(x) &
\end{array}\right. . \tag{B.8}$$

En prenant la convention $J_{-1}^g = 0$ et $\phi_{(N+1)/2}^g = 0$, on obtient pour chaque harmonique l (avec $l \in [\![0, \frac{N-1}{2}]\!]$) les systèmes de diffusion couplés suivants :

$$\left\{\begin{array}{lcl}
\Sigma_{a,2l}^g(x)\phi_l^g(x) + \frac{2l+1}{4l+1}\mathrm{div}\,\overrightarrow{J_l^g}(x) & = & S_l^g(x) - \frac{2l}{4l+1}\mathrm{div}\,\overrightarrow{J_{l-1}^g}(x) \\
\frac{2l+1}{4l+3}\overrightarrow{grad}\,\phi_l^g(x) + \Sigma_{a,2l+1}^g(x)\overrightarrow{J_l^g}(x) & = & \overrightarrow{S_l^g}(x) - \frac{2l+2}{4l+3}\overrightarrow{grad}\,\phi_{l+1}^g(x)
\end{array}\right. . \tag{B.9}$$

Annexe C

Matrices de couplage

C.1 Notations des matrices de couplage

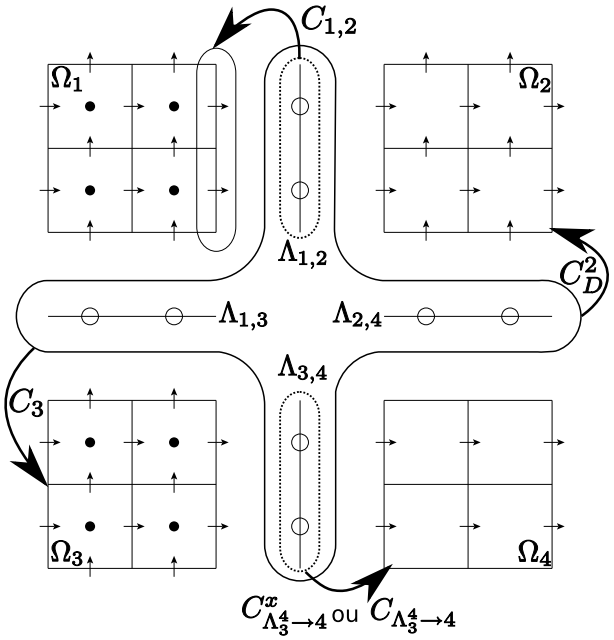


FIG. C.1 – Aide mémoire sur les matrices de couplage : représentation des espaces de départ et d'arrivée correspondant à chacune des notations pour les matrices de couplage.

Notation	introduction	espace de départ	espace d'arrivée
$C_{i,j}$	page 52	l'interface $\Gamma_{i,j}$	les degrés de liberté de courant du domaine Ω_i présents sur $\Gamma_{i,j}$
C_D^i	page 46 et 53	l'ensemble des interfaces Γ	les degrés de liberté de courant du sous-domaine Ω_i
C_i	page 53	l'ensemble des interfaces Γ	l'ensemble des degrés de liberté du sous-domaine Ω_i
$C^d_{\Lambda_i^j \to j}$	page 81	l'interface $\Gamma_{i,j}$	les degrés de liberté de courant dans la direction d du sous-domaine Ω_j

C.2 Calcul des matrices de couplage

Le calcul des matrices de couplage $C_{i\to j}$ dépend du choix des fonctions de base pour discrétiser les multiplicateurs de Lagrange. Suite à [48], on choisit de discrétiser chaque multiplicateur $\Lambda_{i,j}$ comme la trace des fonctions de base de courant associées au maillage le plus fin des maillages des sous-domaines Ω_i et Ω_j et à l'ordre de l'élément fini le plus élevé des ordres utilisés dans les sous-domaines Ω_i et Ω_j. Dans un premier temps, on se place dans le cas 2D, puis on montre qu'il est possible de ramener le cas 3D à deux calculs 2D. Enfin une numérotation des vecteurs d'interface permettant une implémentation performante des produits par les matrices $C_{i\to j}$ et ${}^tC_{i\to j}$ est présentée.

Cas 2D

Sans perte de généralité, on considère une interface $\Gamma_{i,j}$ entre les sous-domaines Ω_i et Ω_j placée en $x = cte$. Dans la suite, le raisonnement fait pour Ω_i est équivalent à celui fait pour Ω_j. On note $(\chi^i_{l_i})_{0\leq l_i<N_i}$ les fonctions de base de J^i_x qui ne s'annulent pas sur $\Gamma_{i,j}$ et $(\mu_l)_{0\leq l<N_\Gamma}$ les fonctions de base de $\Lambda_{i,j}$, la matrice[50] $C_{i\to j}$ est donnée par :

$$\forall\, l_i \in [\![0, N_i - 1]\!], \quad \forall l \in [\![0, N_\Gamma - 1]\!], \quad \Big(C_{i\to j}\Big)_{l_i,l} = \int_{\Gamma_{i,j}} \chi^i_{l_i}\mu_l \, d\Gamma_{i,j}.$$

Soient k_i et h_i, respectivement k_Γ et h_Γ[51], l'ordre d'approximation et la taille des cellules suivant y du maillage (qu'on suppose uniforme) du sous-domaine Ω_i, respectivement du Lagrangien $\Lambda_{i,j}$. Comme on fait le choix $k_\Gamma = max\{k_i, k_j\}$ et $h_\Gamma = min\{h_i, h_j\}$, on se place dans le cas suivant : $k_i \leq k_\Gamma$, $h_\Gamma = h_i/r$ avec $r \geq 1$.
On choisit pour fonctions de base de $\Lambda_{i,j}$, la projection des fonctions de base de courant[52] associées à une taille de maillage h_Γ et un ordre k_Γ, auxquelles on applique un facteur α_Γ que l'on explicite par la suite[53]. On note m_i, respectivement $m_\Gamma = r.m_i$, le nombre de cellules dans la direction y du sous-domaine Ω_i, respectivement du Lagrangien $\Lambda_{i,j}$. Avec ces notations $N_i = m_i(k_i + 1)$ et $N_\Gamma = m_\Gamma * (k_\Gamma + 1)$.

La matrice $C_{i\to j}$ de taille $N_i \times N_\Gamma$ est une matrice diagonale par bloc constituée de m_i blocs identiques $C_{k_i,k_\Gamma,r}$ de taille $(k_i + 1) \times r(k_\Gamma + 1)$:

$$C_{i\to j} = \begin{pmatrix} C_{k_i,k_\Gamma,r} & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & C_{k_i,k_\Gamma,r} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & C_{k_i,k_\Gamma,r} \end{pmatrix}.$$

où $C_{k_i,k_\Gamma,r}$ ne dépend que des ordres k_i et k_Γ et du coefficient de raffinement r de l'interface $\Gamma_{i,j}$ par rapport au sous-domaine Ω_i. $C_{k_i,k_\Gamma,r}$ est donnée par :

[50] Notée C_i dans [48].
[51] Pour simplifier les notations, on confond Γ et $\Gamma_{i,j}$.
[52] Elles sont définies dans [16].
[53] Il peut être différent pour chaque fonction de base, mais ne doit dépendre que de données spécifiques à l'interface (h_Γ et k_Γ).

$\forall s \in [\![0, k_i]\!], \quad \forall t \in [\![0, k_\Gamma]\!], \quad \forall l \in [\![0, r-1]\!],$

$$\begin{aligned}
\left(C_{k_i,k_\Gamma,r}\right)_{s,l(k_\Gamma+1)+t} &= \alpha_\Gamma \int_{l.h_\Gamma}^{(l+1)h_\Gamma} \frac{2}{h_i w_s^{k_i}} l_s^{k_i}\left(\frac{2}{h_i}y - 1\right) \frac{2}{h_\Gamma w_t^{k_\Gamma}} l_t^{k_\Gamma}\left(\frac{2}{h_\Gamma}y - 1 - 2l\right) dy, \\
&= \frac{2\alpha_\Gamma}{h_i w_s^{k_i} w_t^{k_\Gamma}} \int_{-1}^{1} l_s^{k_i}\left(\frac{h_\Gamma}{h_i}(u+1+2l) - 1\right) l_t^{k_\Gamma}(u)\, du, \\
&= \frac{2\alpha_\Gamma}{h_i w_s^{k_i} w_t^{k_\Gamma}} \sum_{p=0}^{k_\Gamma} w_p^{k_\Gamma} l_s^{k_i}\left(\frac{1}{r}(x_p^{k_\Gamma} + 1 + 2l) - 1\right) l_t^{k_\Gamma}\left(x_p^{k_\Gamma}\right),
\end{aligned}$$

où :
- les réels $\left(x_p^k\right)_{1\leq p\leq k+1}$, désignent les points de Gauss d'ordre $k+1$;
- les réels $\left(w_p^k\right)_{1\leq p\leq k+1}$ désignent les poids de quadrature d'ordre $k+1$;
- la fonction l_i^k le i-ème polynôme de Legendre associé aux points de Gauss $\left(x_p^k\right)_{1\leq p\leq k+1}$.

Comme $l_t^{k_\Gamma}\left(x_p^{k_\Gamma}\right) = \delta_{t,p}$, en choisissant $\alpha_\Gamma = \dfrac{h_\Gamma}{2}$, il vient que :

$$\forall s \in [\![0, k_i]\!],\ \forall t \in [\![0, k_\Gamma]\!],\ \forall l \in [\![0, r-1]\!], \quad \left(C_{k_i,k_\Gamma,r}\right)_{s,l(k_\Gamma+1)+t} = \frac{1}{r w_s^{k_i}} l_s^{k_i}\left(\frac{1}{r}(x_t^{k_\Gamma} + 1 + 2l) - 1\right). \tag{C.1}$$

On a choisi α_Γ pour que l'expression de la matrice $C_{k_i,k_\Gamma,r}$ ne dépende que de données entières :
- l'ordre d'approximation du sous-domaine Ω_i ;
- l'ordre d'approximation de l'interface ;
- le coefficient de raffinement de la taille du maillage de l'interface par rapport à celle du sous-domaine Ω_i.

Dans le cas de maillages coïncidents ($h_\Gamma = h_i = h$) et avec des ordres d'élément fini identiques ($k_\Gamma = k_i = k$), l'expression (C.1) peut se simplifier :

$$\forall (r,s) \in [\![0, k+1]\!]^2, \quad \left(C_{k,k,1}\right)_{s,t} = \frac{2\alpha_\Gamma}{h w_s^k} l_s^k(x_t^k) = \frac{2\alpha_\Gamma}{h w_s^k}\delta_{s,t}.$$

En prenant, $\alpha_\Gamma = \dfrac{h w_s^k}{2}$, on a donc $C_{k,k,1} = I$.

Cas 3D

On considère une interface $\Gamma_{i,j}$ entre les sous-domaines Ω_i et Ω_j dirigée suivant l'axe d[54]. On note $(\chi_{l_i}^i)_{1\leq l_i\leq N_i^{d+1}.N_i^{d+2}}$ les fonctions de base de J_i qui ne s'annulent pas sur l'interface $\Gamma_{i,j}$ et $(\mu_l)_{0\leq l\leq N_\Gamma^{d+1}.N_\Gamma^{d+2}}$ les fonctions de base de $\Lambda_{i,j}$, la matrice $C_{i\to j}$ est donnée par :

$$\forall l_i \in [\![0, N_i^{d+1}.N_i^{d+2} - 1]\!], \quad \forall l \in [\![0, N_\Gamma^{d+1}.N_\Gamma^{d+2} - 1]\!], \quad \left(C_{i\to j}\right)_{l_i,l} = \int_{\Gamma_{i,j}} \chi_{l_i}^i \mu_l \, d\Gamma_{i,j}.$$

On note $k_i^{d'}$ et $h_i^{d'}$, respectivement $k_\Gamma^{d'}$ et $h_\Gamma^{d'}$, l'ordre d'approximation et la taille des cellules suivant l'axe d' du maillage (qu'on suppose uniforme) du sous-domaine Ω_i, respectivement du Lagrangien $\Lambda_{i,j}$. On se place dans le cas suivant : $\forall d' \in \{d+1, d+2\} \quad k_i^{d'} \leq k_\Gamma^{d'}$, $h_\Gamma^{d'} = h_i^{d'}/r^{d'}$ avec $r^{d'} \geq 1$.

[54] Les axes sont définis modulo 3.

On choisit pour fonction de base sur $\Lambda_{i,j}$, la projection des fonctions de base de courant du sous-domaine associé à la taille de maillage dans la direction $d+1$ (respectivement $d+2$) h_Γ^{d+1} (respectivement h_Γ^{d+2}) et à l'ordre d'approximation k_Γ^{d+1} (respectivement k_Γ^{d+2}), projection à laquelle on applique un facteur correctif $\alpha_\Gamma = \dfrac{h_\Gamma^{d+1} h_\Gamma^{d+2}}{4}$.

On note m_i^{d+1} respectivement $m_\Gamma^{d+1} = r^{d+1}.m_i^{d+1}$ le nombre de cellules dans la direction $d+1$ du sous-domaine Ω_i respectivement du Lagrangien $\Lambda_{i,j}$. Avec ces notations $N_i^{d+1} = m_i^{d+1}(k_i+1)$ et $N_\Gamma^{d+1} = m_\Gamma^{d+1}(k_\Gamma^{d+1}+1)$. On fait de même pour la direction $d+2$. On note *indice* une fonction permettant de passer d'une numérotation à deux indices à une numérotation à un seul indice.

La matrice $C_{i\to j}$ de taille $N_i^{d+1}.N_i^{d+2} \times N_\Gamma^{d+1}.N_\Gamma^{d+2}$ est une matrice diagonale par bloc constituée de $m_i^{d+1}.m_i^{d+2}$ blocs identiques $C_{k_i^{d+1},k_i^{d+2},k_\Gamma^{d+1},k_\Gamma^{d+2},r^{d+1},r^{d+2}}$ de taille $(k_i^{d+1}+1)(k_i^{d+2}+1) \times r^{d+1}r^{d+2}(k_\Gamma^{d+1}+1)(k_\Gamma^{d+2}+1)$.
On a : $\forall s^{d+1} \in [\![0, k_i^{d+1}]\!]$, $\forall t^{d+1} \in [\![0, k_\Gamma^{d+1}]\!]$, $\forall l^{d+1} \in [\![0, r^{d+1}-1]\!]$, $\forall s^{d+2} \in [\![0, k_i^{d+2}]\!]$, $\forall t^{d+2} \in [\![0, k_\Gamma^{d+2}]\!]$, $\forall l^{d+2} \in [\![0, r^{d+2}-1]\!]$,

$$
\begin{aligned}
& \Big(C_{k_i^{d+1},k_i^{d+2},k_\Gamma^{d+1},k_\Gamma^{d+2},r^{d+1},r^{d+2}}\Big)_{indice(s^{d+1},s^{d+2}),indice(l^{d+1}(k_\Gamma^{d+1}+1)+t^{d+1},l^{d+2}(k_\Gamma^{d+2}+1)+t^{d+2})} \\
= & \int_{l^{d+1}.h_\Gamma^{d+1}}^{(l^{d+1}+1)h_\Gamma^{d+1}} \int_{l^{d+2}.h_\Gamma^{d+2}}^{(l^{d+2}+1)h_\Gamma^{d+2}} \Big(\frac{4}{h_i^{d+1}h_i^{d+2} w_{s^{d+1}}^{k_i^{d+1}} w_{s^{d+2}}^{k_i^{d+2}}} l_{s^{d+1}}^{k_i^{d+1}}\big(\frac{2}{h_i^{d+1}}u_{d+1}-1\big) l_{s^{d+2}}^{k_i^{d+2}}\big(\frac{2}{h_i^{d+2}}u_{d+2}-1\big)\Big) \\
& \Big(\frac{1}{w_{t^{d+1}}^{k_\Gamma^{d+1}}} l_{t^{d+1}}^{k_\Gamma^{d+1}}\big(\frac{2}{h_\Gamma^{d+1}}u_{d+1}-1-2l^{d+1}\big) \frac{1}{w_{t^{d+2}}^{k_\Gamma^{d+2}}} l_{t^{d+2}}^{k_\Gamma^{d+2}}\big(\frac{2}{h_\Gamma^{d+2}}u_{d+2}-1-2l^{d+2}\big)\Big) du_{d+1}du_{d+2} \\
= & \frac{2}{h_i^{d+1} w_{s^{d+1}}^{k_i^{d+1}} w_{t^{d+1}}^{k_\Gamma^{d+1}}} \int_{l^{d+1}.h_\Gamma^{d+1}}^{(l^{d+1}+1)h_\Gamma^{d+1}} \Big(l_{s^{d+1}}^{k_i^{d+1}}\big(\frac{2}{h_i^{d+1}}u_{d+1}-1\big) l_{t^{d+1}}^{k_\Gamma^{d+1}}\big(\frac{2}{h_\Gamma^{d+1}}u_{d+1}-1-2l^{d+1}\big)\Big) du_{d+1} \times \\
& \frac{2}{h_i^{d+2} w_{s^{d+2}}^{k_i^{d+2}} w_{t^{d+2}}^{k_\Gamma^{d+2}}} \int_{l^{d+2}.h_\Gamma^{d+2}}^{(l^{d+2}+1)h_\Gamma^{d+2}} \Big(l_{s^{d+2}}^{k_i^{d+2}}\big(\frac{2}{h_i^{d+2}}u_{d+2}-1\big) l_{t^{d+2}}^{k_\Gamma^{d+2}}\big(\frac{2}{h_\Gamma^{d+2}}u_{d+2}-1-2l^{d+2}\big)\Big) du_{d+2} \\
= & \Big(C_{k_i^{d+1},k_\Gamma^{d+1},r^{d+1}}\Big)_{s^{d+1},l^{d+1}(k_\Gamma^{d+1}+1)+t^{d+1}} \Big(C_{k_i^{d+2},k_\Gamma^{d+2},r^{d+2}}\Big)_{s^{d+2},l^{d+2}(k_\Gamma^{d+2}+1)+t^{d+2}}.
\end{aligned}
$$

Le calcul des matrices de couplage en 3D est donc ainsi ramené aux calculs de matrices de couplage en 2D.

C.3 Numérotation des interfaces

Les seules opérations sur les vecteurs d'interface, qui demandent de connaître précisément la numérotation des vecteurs d'interface, sont les produits par les matrices de couplage ou leurs transposées. Cela permet donc de choisir une numérotation des vecteurs d'interface qui soit la plus adaptée possible à ces opérations. Comme les matrices de couplage $C_{i\to j}$, sont des matrices diagonales par bloc, on propose donc une numérotation qui respecte cette structure bloc. La récupération des données sur les interfaces nécessite de manipuler plusieurs numérotations :

- N_i^d : la numérotation locale des degrés de liberté de courant de Ω_i suivant la direction d ;
- $N_{i,j}^d$: la numérotation des degrés de liberté de courant de Ω_i suivant la direction d présents sur $\Gamma_{i,j}$;
- $N_{i,j}^\Gamma$: la numérotation des degrés de liberté du multiplicateur de Lagrange associé à l'interface $\Gamma_{i,j}$.

On explicite ces trois numérotations, en se placant dans le cas 3D.

Numérotation N_i^d

Cette numérotation est déjà définie dans le code existant. On pose les notations suivantes[55] :

- n_x, n_y, n_z correspondent aux nombres de cellules dans chaque direction ;
- k_x, k_y, k_z déterminent l'ordre de l'élément fini choisi dans chaque direction[56] ;
- N_x, N_y, N_z représentent, pour chaque direction, le nombre de degrés de liberté de flux sur une ligne de couplage : $\forall\, d \in \{x, y, z\} \quad N_d = n_d.(k_d + 1)$;
- M_x, M_y, M_z représentent, pour chaque direction, le nombre de degrés de liberté de courant sur une ligne de couplage : $\forall\, d \in \{x, y, z\} \quad M_d = N_d + 1$.

Avec ces notations, le j_d^{ieme} degré de liberté, placé sur la ligne de courant de direction d repérée par les indices cartésiens i_{d+1} et i_{d+2},[57] se voit attribuer l'indice suivant :

$$I_d = j_d + M_d i_{d+1} + M_d N_{d+1} i_{d+2}.$$

Cette formalisation correspond à une numérotation suivant la direction d, puis $d+1$ et enfin $d+2$[58]. Cette numérotation est illustrée en noir sur la figure Fig. C.2. Pour des informations plus détaillées sur la numérotation locale des degrés de liberté d'un sous-domaine, on renvoie le lecteur à la note [83].

Numérotation $N_{i,j}^d$

On introduit deux indices afin que la matrice de couplage $C_i^{i,j}$ soit diagonale par bloc, ce qui revient à rendre consécutifs les degrés de liberté de courant sur l'interface appartenant à la même classe d'équivalence pour la relation R : *être situé dans la même maille grossière.*
Le premier indice correspond ainsi au numéro de la maille grossière, numérotée d'abord sur la direction $d+1$, puis sur la direction $d+2$. Cet indice varie donc entre 0 et le nombre de cellules du sous-domaine maillé le plus grossièrement ayant une frontière commune avec l'interface.

Le second indice correspond à la numérotation des degrés de liberté de courant, internes à chaque cellule grossière, avec toujours une numérotation suivant $d+1$ puis $d+2$. Si Ω_i est le sous-domaine maillé finement, cet indice appartient donc à $[\![0, r_{d+1} * r_{d+2} * (o_{d+1}+1) * (o_{d+2}+1) - 1]\!]$ avec r_d le rapport dans la direction d du nombre de cellules du sous-domaine grossier par celui du nombre de cellules du sous-domaine fin. En revanche, si le sous-domaine est maillé grossièrement cet indice appartient à $[\![o_{d+1} * o_{d+2} - 1]\!]$. Cette numérotation est illustrée en vert[59] sur la figure Fig. C.2.

Numérotation $N_{i,j}^\Gamma$

On choisit parmi les numérotations $N_{i,j}^d$ et $N_{j,i}^d$, celle correspondant au sous-domaine maillé le plus finement (voir figure Fig. C.2 en rouge[60]). En revanche, on doit expliciter pour celle-ci la correspondance entre la numérotation à deux indices et une à un seul indice, car on stocke en mémoire l'ensemble de l'interface. Si on note $Cellule$ le premier indice et ddl le second, l'indice global s'obtient donc naturellement par la formule $Indice = ddl + Cellule.nbddlParCellule$ où

[55] Pour clarifier les notations, on omet l'indice i du sous-domaine.
[56] Il est en effet possible de mixer les différents ordres d'élément fini.
[57] Les directions étant définies modulo 3.
[58] On rappelle que le flux est numéroté selon xyz.
[59] Cette numérotation figure en dessous des flèches représentant les degrés de liberté de courant.
[60] Cette numérotation figure sur l'interface, en dessus des flèches représentant les degrés de liberté du multiplicateur.

nbddlParCellule correspond au nombre de degrés de liberté du sous-domaine maillé finement dans une cellule du sous-domaine maillé grossièrement.

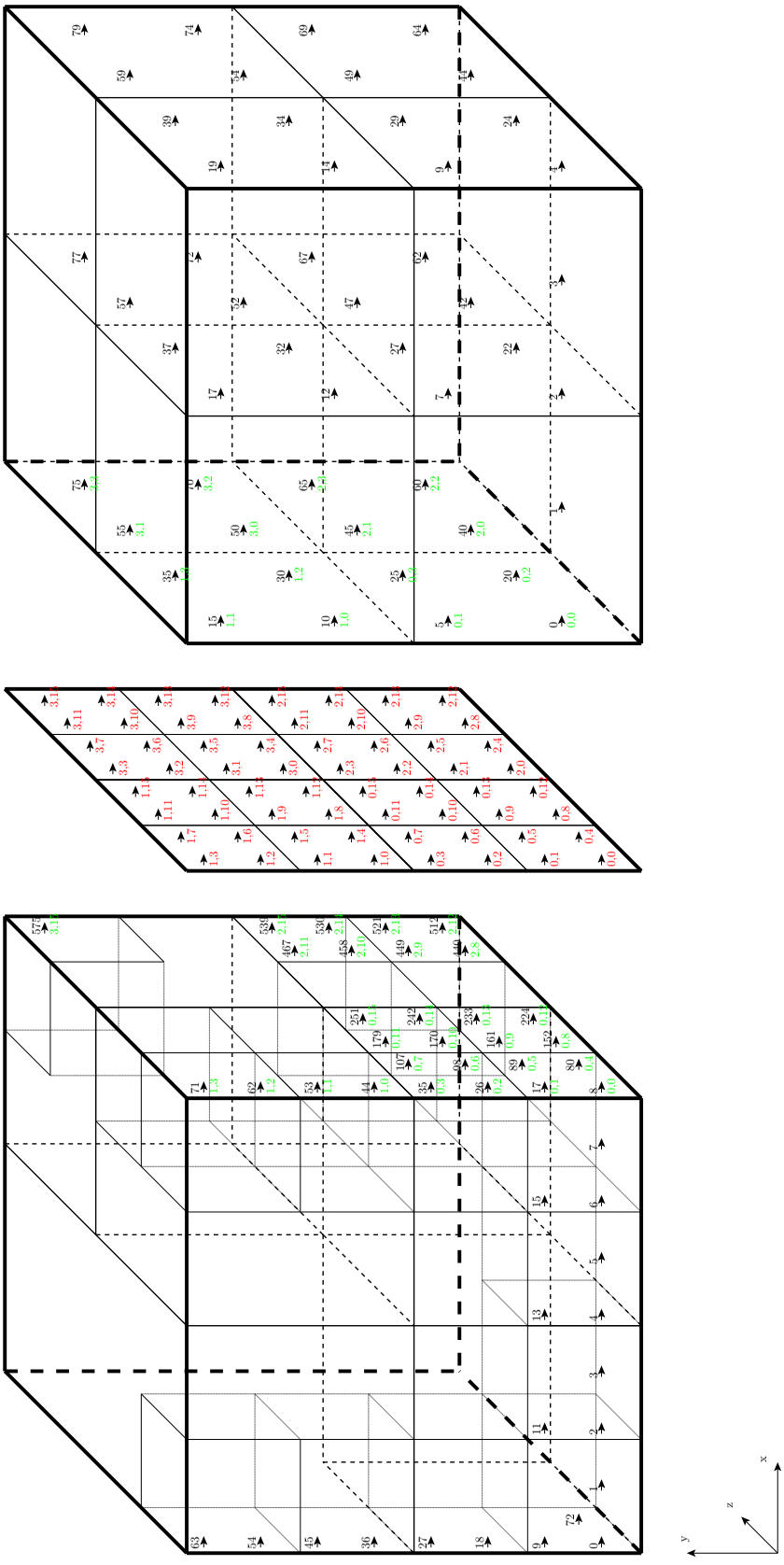


FIG. C.2 – Illustration des différentes numérotations

Annexe D

Machine utilisée : cluster `rendvous`

Configuration

Pour les résultats numériques, le clusteur `rendvous` d'EDF R&D est utilisé. Il est constitué de 208 nœuds :

- 180 noeuds avec 4 Go de Ram ;
- 20 noeuds avec 8 Go de Ram ;
- 8 noeuds avec 16 Go de Ram.

Sur chacun de ces nœuds, se trouvent :

- 2 processeurs Intel XEON 3.4 Ghz avec 2Mo de cache L2 (référence SL7ZD) ;
- une carte réseau Infiniband reliée à un switch Infiniband (`Voltaire IB DDR 288P - 264 ports`).

L'implémentation MPI est `mvapich-0.9.9` avec la bibliothèque Infiniband `openib-2.0.5` et le compilateur `g++ (GCC) 4.1.2 20061115 (prerelease) (Debian 4.1.1-21)`.

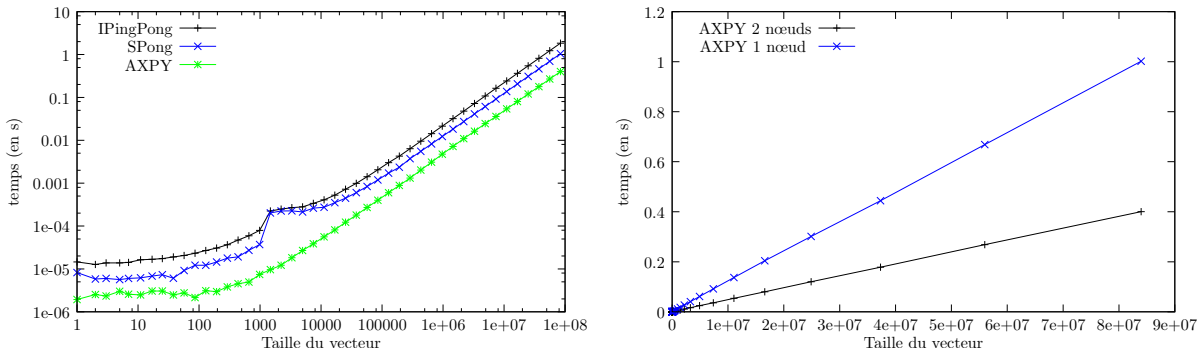
Mesures de référence

La latence et le débit du réseau rapide sont obtenus à partir d'un programme IPingPong écrit en mpi avec des appels asynchrones effectuant un *ping pong* entre deux nœuds ; la latence est de l'ordre de $6.10^{-6}\ s$ et le débit de $580\ Mo.s^{-1}$. L'exécution d'un AXPI, codé par une simple boucle en C, est 4 à 5 fois plus rapide que l'échange par ping-pong d'un vecteur de même taille (cf. figure Fig. D.1(a)). Sur cette figure on remarque également qu'avec le programme SPong (S pour Simultané), où l'on commence les deux communications simultanément, un gain de performance est obtenue en comparaison au simple programme IPingPong.

Pour des codes limités par les accès mémoires (telle que l'opération `Y=Y+X`), il n'est pas efficace d'utiliser deux processus par nœud en raison du bus mémoire commun aux deux processeurs. Sur la figure Fig. D.1(b), on compare l'exécution parallèle avec deux processus de l'opération `Y=Y+X` en plaçant les deux processus :

- sur des nœuds différents ;
- sur un même nœud.

Ces mesures ne prennent en compte que le calcul et ne sont donc pas influencées par les communications réseaux. On observe qu'en raison de conflits d'accès sur le bus mémoire, la deuxième solution est plus que deux fois plus lente que la première.



(a) Comparaison Calcul / Communication

(b) Mise en évidence de la contention mémoire

FIG. D.1 – Mesure de performances sur le cluster `rendvous`

Annexe E

Liste des publications

Publications dans une conférence avec comité de sélection

Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A Domain Decomposition Method Applied to the Simplified Transport Equations », dans *Proceedings of Computational Science and Engineering (CSE)*, p. 91–97, 2008.

Barrault(M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A Non Overlapping Parallel Domain Decomposition Method Applied to The Simplified Transport Equations », dans *Proceedings of Mathematics, Computational Methods & Reactor Physics*, 2009.

Publication dans un journal

Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « Efficient Parallel Resolution of The Simplified Transport Equation in Mixed-Dual Formulation », En préparation.

Communications orales dans une conférence internationale

Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A domain decomposition method for the resolution of an eigenvalue problem in neutron physics », IMACS 2008.

Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A domain decomposition method applied to large eigenvalue problems in neutron physics », PMAA 2008.

Bibliographie

[1] Plagne (L.) et Ponçot (A.), « Generic programming for deterministic neutron transport codes », dans *Proceedings of Mathematical and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Palais des Papes, Avignon, France*, 2005. cité p. 4, 13

[2] Ponçot (A.), « Projet DESCARTES : notice théorique du solveur SPN-GLASS ». Note interne n° HI-23/06/003/P, EDF R&D, 2006. cité p. 4, 13, 105

[3] Lautard (J. J.) et Moreau (F.), « A fast 3D Parallel Solver Based on the Mixed Dual Finite Element Approximation », dans *Mathematics & Computation and Super Computing in Nuclear Applications*, 1993. cité p. 4, 13

[4] Baudron (A. M.), Lautard (J. J.) et Schneider (D.), « Mixed dual methods for neutronic reactor core calculations in the CRONOS system », dans *Reactor Physics and Environmental Analysis of Nuclear Systems*, 1999. cité p. 4, 13

[5] Lewis (E. E.) et Miller (W. F.), *Computational Methods of Neutron Transport*. Wiley, New York, 1984. cité p. 4, 12

[6] Reuss (P.), *Précis de neutronique*. EDP sciences, 2003. cité p. 4

[7] Pinchedez (K.), *Calcul parallèle pour les équations de diffusion et de transport homogènes en neutronique*. Thèse de doctorat, Université Paris XI, 1999. cité p. xv, 4, 12, 39, 40, 105

[8] Schneider (D.), *Éléments finis mixtes duaux pour la résolution numérique de l'équation de la diffusion neutronique en géométrie hexagonale*. Thèse de doctorat, Université Paris VI, 2001. cité p. 4, 47

[9] Guérin (P.), *Méthodes de décomposition de domaine pour la formulation mixte duale du problème critique de la diffusion des neutrons*. Thèse de doctorat, Université Paris VI, 2007. cité p. 4, 40, 41, 95

[10] Chauvet (S.), *Méthode multi-échelle pour la résolution des équations de la cinétique neutronique*. Thèse de doctorat, Université de Nantes, 2008. cité p. 4

[11] Barrault (M.), « Notice théorique sur la mise en œuvre de l'accélération de Tchebychev en neutronique ». Note Interne n° H-I23-2006-01614-FR, EDF R&D, 2006. cité p. 9, 66

[12] Barrault (M.) et Gilles (E.), « Mise en œuvre de l'accélération de Tchebychev en neutronique ». Note Interne n° H-I23-2008-00225-FR, EDF R&D, 2008. cité p. 9, 66

[13] Coulomb (F.) et Fedon-Magnaud (C.), « Mixed and mixed-hybrid elements for the diffusion equation », *Nuclear Science and Engineering*, vol. 100, n° 3, 1988, p. 218 – 225. cité p. 13, 47

[14] Nédélec (J. C.), « A new family of mixed finite elements in $\mathbb{R}^3$ », *Numerische Mathematik*, vol. 50, 1986, p. 57–81. cité p. 14

[15] Raviart (P. A.) et Thomas (J. M.), « A mixed finite element method for second order elliptic problems », *Lecture Notes in Mathematics*, 1977. cité p. 14

[16] Barrault (M.), « Extention du code SPn-GLASS à l'élément fini RTk ». Note Interne, EDF R&D, 2006. cité p. 19, 110

[17] Douglas (J.), Durin (R.) et Pietra (P.), « Alternating-direction iteration for mixed finite element methods », *Computing methods in applied sciences and engineering*, vol. VII, 1986, p. 181 – 196. cité p. 21

[18] Douglas (J.), Durin (R.) et Pietra (P.), « Formulation of alternating-direction iterative methods for mixed methods in three space », *Numerical approximation of partial differential equations*, 1987, p. 21 – 29. cité p. 21

[19] Gu (H.) et Wu (X.), « Alternating-direction iterative technique for mixed finite element methods for Stokes equations », *Applied Mathematics and Computation*, 2005. cité p. 21

[20] Van den Eshof (J.), Sleijpen (G. L. G.) et Van Gijzen (M. B.), « Relaxation strategies for nested Krylov methods. ». Technical Report n° TR/PA/03/27, CERFACS, 2003. cité p. 22

[21] Bouras (A.) et Frayssé (V.), « A relaxation strategy for inexact matrix-vector products for Krylov methods ». Technical Report n° TR/PA/00/15, CERFACS, 2000. Soumis à SIAM Journal in Matrix Analysis and Applications. cité p. 22

[22] Bouras (A.) et Frayssé (V.), « A relaxation strategy for the Arnoldi method in eigenproblems ». Technical Report n° TR/PA/00/16, CERFACS, 2000. cité p. 22

[23] Bouras (A.), Frayssé (V.) et Giraud (L.), « A relaxation strategy for inner-outer linear solvers in domain decomposition methods ». Technical Report n° TR/PA/00/17, CERFACS, 2000. cité p. 22

[24] Chaitin-Chatelin (F.) et Meškauskas (T.), « Inner-outer iterations for mode solvers in structural mechanics : application to the Code Aster ». Contract Report n° TR/PA/01/85, CERFACS, 2001. cité p. 22

[25] Simoncini (V.) et Szyld (D. B.), « Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing », *SIAM Journal on Scientific Computing*, 2003, p. 454–477. cité p. 22

[26] Van den Eshof (J.), Sleijpen (G. L. G.) et Van Gijzen (M. B.), « Iterative linear system solvers with approximate matrix-vector products ». Technical Report n° TR/PA/04/133, CERFACS, 2004. cité p. 22

[27] Lai (Y.-L.), Lin (K.-Y.) et Lin (W.-W.), « An Inexact Inverse Iteration for Large Sparse Eigenvalue Problems », *Linear Algebra and its Applications*, vol. 4, 1997, p. 425–437. cité p. 22

[28] Golub (G. H.) et Ye (Q.), « Inexact Inverse Iterations for the Generalized Eigenvalue Problems », *BIT Numerical Mathematics*, vol. 40, 1999, p. 672–684. cité p. 22

[29] Micheelsen (B.), « 3D IAEA Benchmark Problem ». Rapport technique, IAEA, 1977. cité p. 24

[30] Schwartz (N.), Couyras (D.), Ponçot (A.) *et al.*, « Projet N3Cv2 - Schéma de calcul Industriel - Description et élément de validation sur GR514 ». Note Interne n° H-I27-2007-02879-FR, EDF R&D, 2007. cité p. 30

[31] Guillo (M.), Couyras (D.) et Hoareau (F.), « Projet N3Cv2 - Schéma de calcul Industriel - Etude de faisabilité d'un calcul 3D de cœur crayon par crayon ». Note Interne n° H-I27-2008-01726-FR, EDF R&D, 2008. cité p. 30

[32] Boiteau (O.), « Décomposition de domaine et parallélisme en mécanique des structures : État de l'art et benchmark pour une implémentation raisonnée dans Code-Aster ». Notes internes n° HI-23/03/009/A et HI-23/07/03174, EDF R&D, 2003 et 2008. cité p. 34

[33] Smith (B. F.), Bjørstad (P. E.) et Gropp (W. D.), *Domain decomposition : parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, New York, NY, USA, 1996. cité p. 34

[34] Magoules (F.), *Mesh Partitioning Techniques and Domain Decomposition Methods*. Saxe-Coburg Publications, 2008. cité p. 34

[35] Chan (T. F.) et Mathew (T. P.), « Domain Decomposition Algorithms », dans *Acta Numerica*, p. 61–143. Cambridge University Press, 1994. cité p. 34

[36] Le Tallec (P.), « Domain decomposition methods in computational mechanics », dans Oden (J. T.), éditeur, *Computational Mechanics Advances*, vol. 1 (2), p. 121–220. North-Holland, 1994. cité p. 34, 38

[37] Xu (J.) et Zou (J.), « Some Nonoverlapping Domain Decomposition Methods », *SIAM Review*, vol. 40, n° 4, 1998, p. 857–914. cité p. 34

[38] Nataf (F.). « Interface Connections in Domain Decomposition Methods ». 2006. cité p. 35

[39] Farhat (C.) et Roux (F.-X.), « An Unconventional Domain Decomposition Method for an Efficient Parallel Solution of Large-Scale Finite Element Systems », *Journal on Scientific and Statistical Computing*, vol. 13, 1992, p. 379 – 396. cité p. 38

[40] Coulomb (F.), « Domain Decomposition and Mixed Finite for The Neutron Diffusion Equation », dans *Second International Symposium on Domain Decomposition Methods*, Philadelphia, PA, 1988. SIAM. cité p. 38

[41] Guérin (P.), Baudron (A. M.), Lautard (J. J.) et Van Criekingen (S.), « Component mode synthesis methods for 3-D heterogeneous core calculations applied to the mixed-dual finite element solver MINOS », *Nuclear Science and Engineering*, vol. 155, n° 2, 2007, p. 264–275. cité p. 40

[42] Guérin (P.), Baudron (A. M.) et Lautard (J. J.), « Domain Decomposition methods for core calculations using the MINOS solver », dans *Joint International Topical Meeting on Mathematics & Computation and super Computing in Nuclear Applications*, 2007. cité p. 40

[43] Vaudescal (J. L.), « Mise en œuvre d'un algorithme de décomposition de domaines pour les calculs crayon par crayon tridimentionnels en diffusion multigroupe ». Note Interne n° HI-72/97/010/0, EDF R&D, 1997. cité p. 41

[44] Vaudescal (J. L.), « Une méthode de décomposition de domaines pour le raffinement local en diffusion neutronique ». Note Interne, EDF R&D, 1995. cité p. 41

[45] Warin (X.), « Raffinement local en transport neutronique 3D ». Note Interne, EDF R&D, 1998. cité p. 41

[46] Caramel (G.), « Document de synthèse portant sur l'analyse des résultats lors de l'utilisation de l'algorithme GMRES préconditionné dans le code SPn séquentiel et parallèle ». Note Interne, Inria Bordeaux Sud-Ouest/EDF R&D, 2007. cité p. 42

[47] Saad (Y.), *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003. cité p. 42, 63, 64

[48] Barrault (M.) et Tonnel (N.), « Une méthode de décomposition de domaine pour le traitement SP1 de l'équation de transport des neutrons ». Note Interne n° HI-23/05/035/A, EDF R&D, 2005. cité p. 44, 51, 110

[49] Ben Belgacem (F.), *Discrétisations 3D non conformes pour la méthode de décomposition de domaine des éléments avec joints : analyse mathématique et mise en oeuvre pour le problème de Poisson*. Thèse de doctorat, EDF, 1993. cité p. 46, 47

[50] Bernardi (C.), Maday (Y.) et Patera (A. T.), « A New Non Conforming Approach to Domain Decomposition : The Mortar Element Method », dans Brezis (H.) et Lions (J.-L.), éditeurs, *Collège de France Seminar*. Pitman, 1994. cité p. 46, 47

[51] Brezzi (F.) et Fortin (M.), *Mixed and hybrid finite element methods*. Springer-Verlag New York, Inc., 1991. cité p. 47

[52] Glowinski (R.) et Wheeler (M. F.), « Domain Decomposition and Mixed Finite Element Methods for Elliptic Problems », dans Glowinski (R.), Golub (G. H.), Meurant (G. A.) et Périaux (J.), éditeurs, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*. SIAM, 1988. cité p. 47

[53] Cowsar (L. C.) et Wheeler (M. F.), « Parallel Domain Decomposition Method for Mixed Finite Elements for Elliptic Partial Differential Equations », dans Glowinski (R.) *et al.*, éditeurs, *Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*. SIAM, 1991. cité p. 47

[54] Cowsar (L. C.), Mandel (J.) et Wheeler (M. F.), « Balancing Domain Decomposition for Mixed Finite Elements », *Mathemactics of Computation*, vol. 64, n° 211, 1995, p. 989–1015. cité p. 47

[55] Arbogast (T.), Cowsar (L. C.), Wheeler (M. F.) et Yotov (I.). « Mixed Finite Element Methods on Non-Matching Multiblock Grids », 1996. cité p. 47

[56] Arbogast (T.) et Yotov (I.), « A Non-Mortar Mixed Finite Element Method For Elliptic Problems On Non-Matching Multiblock Grids », *Computer Methods in Applied Mechanics and Engineering*, vol. 149, 1997, p. 255–265. cité p. 47

[57] Gaiffe (S.), Glowinski (R.) et Masson (R.). « Domain Decomposition and Splitting Methods for Mortar Mixed Approximations to Parabolic Problems », 2001. cité p. 47

[58] Kaasschieter (E. F.) et Huijben (A. J. M.), *Mixed-hybrid finite elements and streamline computation for the potential flow problem*, vol. 8. 1992. cité p. 47

[59] Razafindrakoto (E.), « Approximation des systèmes de Darcy et de Navier-Stokes à masse volumique variable et avec surface libre par la méthode Eléments Finis Mixtes Hybrides ». Note Interne n° HI-P74-2008-03487, EDF R&D, 2008. cité p. 47

[60] Chen (Z.), Ewing (R. E.) et Lazarov (R.), « Domain Decomposition Algorithms For Mixed Methods For Second Order Elliptic Problems », *Mathemactics of Computation*, vol. 65, 1996, p. 467–490. cité p. 47

[61] Vaudescal (J. L.), « Présentation d'algorithmes et proposition d'études numériques dans le cadre de coccinelle ». Note Interne n° HI-72/94/017/0, EDF R&D, 1994. cité p. 47

[62] Van Criekingen (S.) et Beauwens (R.), « Mixed-hybrid discretization methods for the P1 equations », *Applied Numerical Mathematics*, vol. 57, n° 2, 2007, p. 117 – 130. cité p. 47

[63] Douglas (J.) et Huang (C. S.), « Accelerated Domain Decomposition Iterative Procedures for Mixed Methods Based on Robin Transmission Conditions ». Technical Report, 2002. cité p. 47

[64] Hoppe (R.), et Wohlmuth (B. I.), « Adaptive Mixed Hybrid and Macro-Hybrid Finite Element Methods », *Acta Numerica*, 1998, p. 159–179. cité p. 47

[65] Girardi (E.), *Couplage de méthodes et décomposition de domaine pour la résolution des équations de transport*. Thèse de doctorat, Université d'Evry Val-d'Essonne, 1999. cité p. 56

[66] Makarenko (A.), *Parallélisation de la méthode nodale variationnelle pour l'équation du transport neutronique*. Thèse de doctorat, Université de Provence, 1997. cité p. 56

[67] Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A Non Overlapping Parallel Domain Decomposition Method Applied to The Simplified Transport Equations », dans *Proceedings of Mathematics, Computational Methods & Reactor Physics*, May 2009. cité p. 60

[68] Benzi (M.), Golub (G. H.) et Liesen (J.), « Numerical solution of saddle point problems », *Acta Numerica*, vol. 14, 2005, p. 1–137. cité p. 63

[69] Shewchuk (J. R.), « An Introduction to the Conjugate Gradient Method Without the Agonizing Pain ». Rapport technique, Pittsburgh, PA, USA, 1994. cité p. 64

[70] Bertrand (F.) et Tanguy (P. A.), « Krylov-Based Uzawa Algorithms for the Solution of the Stokes Equations Using Discontinuous-Pressure Tetrahedral Finite Elements », *Journal of Computational Physics*, vol. 181, 2002, p. 617–638. cité p. 65

[71] Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « A Domain Decomposition Method Applied to the Simplified Transport Equations », dans *Proceedings of Computational Science and Engineering (CSE)*, p. 91–97, 2008. cité p. 65

[72] Barrault (M.), Lathuilière (B.), Ramet (P.) et Roman (J.), « Efficient Parallel Resolution of The Simplified Transport Equation in Mixed-Dual Formulation », En préparation. cité p. 78

[73] Jamelot (E.) et Lautard (J. J.), « Domain decomposition method fore core calculations using MINOS solver in the APOLLO3 code ». Rapport DM2S n° SERMA/LLPR/RT/4755/A, CEA, 2009. cité p. 95

[74] Kirschenmann (W.), Plagne (L.) et Vialle (S.), « Multi-target c++ implementation of parallel skeletons », dans *POOSC '09 : Proceedings of the 8th workshop on Parallel/High-Performance Object-Oriented Scientific Computing*, p. 1–10. ACM, 2009. cité p. 95

[75] Pastorini (S.), « Description fonctionnelle des calculs-type du code de cœur COCAGNE du projet N3CV2 ». Note Interne n° H-I27-2007-01206, EDF R&D, 2007. cité p. 96

[76] Ozdoba (H.), « Vers une méthode de décomposition de domaine en neutronique massivement parallèle ». Rapport de stage, EDF R&D, 2009. cité p. 98

[77] Hoareau (F.), « COCAGNE : impact des éléments finis RTk différents par direction sur les calculs crayon par crayon 3D ». Note Interne n° CR-I27-2009-77, EDF R&D, 2009. cité p. 98

[78] Akherraz (B.), Ponçot (A.) et Picard (J. F.), « DESCARTES : Spécifications du solveur transport unifié NEO : implémentation 0, cas cartésien ». Note interne n° HI-XX/2002/XXX/X ou CEA-DEN-SERMA/LENR/RT/02-XXXX, EDF, CEA, 2004. NEO a été renommé Domino. cité p. 102

[79] Févotte (F.), Darmet (G.) et Guérin (P.), « Algorithme itératif pour la résolution du transport neutronique ». Note Interne n° H-I23-2009-03442-FR, EDF R&D, 2009. cité p. 102

[80] Kirschenmann (W.), Plagne (L.), Ploix (S.) *et al.*, « Massively parallel solving of 3D simplified P_N equations on graphic processing units », dans *Proceedings of Mathematics, Computational Methods & Reactor Physics*, 2009. cité p. 102

[81] Gelbard (E. M.), « Simplified spherical harmonics equations and their use in shielding problems ». Technical Report n° WAPD-T-1182, Bettis Atomic Power Laboratory, 1961. cité p. 105

[82] Pomraning (C. G.), « Asymptotic and Variational Derivation of the Simplified Pn Equations », *American Nuclear Energy, 20, 9*, 1993, p. 623–637. cité p. 105

[83] Plagne (L.), « SPN-GLASS : Documentation informatique ». Rapport technique, EDF R&D, 2005. cité p. 113