Amélioration des méthodes de calcul de cœurs de réacteurs nucléaires dans APOLLO3 : décomposition de domaine en théorie du transport pour des géométries 2D et 3D avec une accélération non linéaire par la diffusion

Roland Lenain

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE 534 :

MODÉLISATION ET INSTRUMENTATION EN PHYSIQUE, ÉNERGIES, GÉOSCIENCES ET ENVIRONNEMENT

Laboratoire : *CEA/DEN/DANS/DM2S/SERMA Laboratoire de Protection, d'Etudes et de Conception*

THÈSE DE DOCTORAT

PHYSIQUE

Par

Roland LENAIN

Amélioration des méthodes de calcul de cœurs de réacteurs nucléaires dans APOLLO3 : décomposition de domaine en théorie du transport pour des géométries 2D et 3D avec une accélération non linéaire par la diffusion.

Date de soutenance : 15/09/2015

Composition du jury :

Président du jury :	Pierre DESESQUELLES	Professeur (Université Paris Sud)
Directeur de thèse :	Richard SANCHEZ	Directeur de Recherches (CEA)
Rapporteurs :	Jean RAGUSA	Professeur (Texas A&M University)
	Piero RAVETTO	Professeur (Politecnico di Torino)
Examinateurs :	Emiliano MASIELLO	Docteur, ingénieur (CEA)
	Jean-François VIDAL	Docteur, ingénieur (CEA)
Membre invité :	Frédéric DAMIAN	Docteur, ingénieur (CEA)

Amélioration des méthodes de calcul de cœurs de réacteurs nucléaires dans APOLLO3 : décomposition de domaine en théorie du transport pour des géométries 2D et 3D avec une accélération non linéaire par la diffusion

RESUME

Ce travail de thèse est consacré à la mise en œuvre d'une méthode de décomposition de domaine appliquée à l'équation du transport. L'objectif de ce travail est l'accès à des solutions déterministes *haute-fidélité* permettant de correctement traiter les hétérogénéités des réacteurs nucléaires, pour des problèmes dont la taille varie d'un motif d'assemblage en 3 dimensions jusqu'à celle d'un grand cœur complet en 3D. L'algorithme novateur développé au cours de la thèse vise à optimiser l'utilisation du parallélisme et celle de la mémoire. La démarche adoptée a aussi pour but la diminution de l'influence de l'implémentation parallèle sur les performances. Ces objectifs répondent aux besoins du projet APOLLO3, développé au CEA et soutenu par EDF et AREVA, qui se doit d'être un code portable (pas d'optimisation sur une architecture particulière) permettant de réaliser des modélisations haute-fidélité (*best estimate*) avec des ressources allant des machines de bureau aux calculateurs disponibles dans les laboratoires d'études. L'algorithme que nous proposons est un algorithme de Jacobi Parallèle par Bloc Multigroupe. Chaque sous domaine est un problème multigroupe à sources fixes ayant des sources volumiques (fission) et surfaciques (données par les flux d'interface entre les sous domaines). Le problème multigroupe est résolu dans chaque sous domaine et une seule communication des flux d'interface est requise par itération de puissance. Le rayon spectral de l'algorithme de résolution est rendu comparable à celui de l'algorithme de résolution classique grâce à une méthode d'accélération non linéaire par la diffusion bien connue nommée *Coarse Mesh Finite Difference*. De cette manière une scalabilité idéale est atteignable lors de la parallélisation. L'organisation de la mémoire, tirant parti du parallélisme à mémoire partagée, permet d'optimiser les ressources en évitant les copies de données redondantes entre les sous domaines. Les architectures de calcul à mémoire distribuée sont rendues accessibles par un parallélisme hybride qui combine le parallélisme à mémoire partagée et à mémoire distribuée. Pour des problèmes de grande taille, ces architectures permettent d'accéder à un plus grand nombre de processeurs et à la quantité de mémoire nécessaire aux modélisations haute-fidélité. Ainsi, nous avons réalisé plusieurs exercices de modélisation afin de démontrer le potentiel de la réalisation : calcul de cœur et de motifs d'assemblages en 2D et 3D prenant en compte les contraintes de discrétisation spatiales et énergétiques attendues.

Mots-clés : Neutronique, équation du transport des neutrons, méthode des caractéristiques courtes, IDT, décomposition de domaine, algorithme de Jacobi parallèle par bloc multigroupe, Coarse Mesh Finite Difference, parallélisme hybride, APOLLO3.

Contribution to the development of methods for nuclear reactor core calculations with APOLLO3 code: domain decomposition in transport theory for 2D and 3D geometries with nonlinear diffusion acceleration.

ABSTRACT

This thesis is devoted to the implementation of a domain decomposition method applied to the neutron transport equation. The objective of this work is to access high-fidelity deterministic solutions to properly handle heterogeneities located in nuclear reactor cores, for problems' size ranging from colorsets of assemblies to large reactor cores configurations in 2D and 3D. The innovative algorithm developed during the thesis intends to optimize the use of parallelism and memory. The approach also aims to minimize the influence of the parallel implementation on the performances. These goals match the needs of APOLLO3 project, developed at CEA and supported by EDF and AREVA, which must be a portable code (no optimization on a specific architecture) in order to achieve best estimate modeling with resources ranging from personal computer to compute cluster available for engineers analyses. The proposed algorithm is a Parallel Multigroup-Block Jacobi one. Each subdomain is considered as a multi-group fixed-source problem with volume-sources (fission) and surface-sources (interface flux between the subdomains). The multi-group problem is solved in each subdomain and a single communication of the interface flux is required at each power iteration. The spectral radius of the resolution algorithm is made similar to the one of a classical resolution algorithm with a nonlinear diffusion acceleration method: the well-known Coarse Mesh Finite Difference. In this way an ideal scalability is achievable when the calculation is parallelized. The memory organization, taking advantage of shared memory parallelism, optimizes the resources by avoiding redundant copies of the data shared between the subdomains. Distributed memory architectures are made available by a hybrid parallel method that combines both paradigms of shared memory parallelism and distributed memory parallelism. For large problems, these architectures provide a greater number of processors and the amount of memory required for high-fidelity modeling. Thus, we have completed several modeling exercises to demonstrate the potential of the method: 2D full core calculation of a large pressurized water reactor and 3D colorsets of assemblies taking into account the constraints of space and energy discretization expected for high-fidelity modeling.

Keywords: Neutronics, neutron transport equation, method of short characteristics, IDT, domain decomposition method, Parallel Multigroup-Block Jacobi algorithm, Coarse Mesh Finite Difference, hybrid parallelism, APOLLO3.

REMERCIEMENTS

Cette thèse a été réalisée au DM2S/SERMA/LPEC. Je tiens donc tout d'abord à remercier Monsieur Patrick Blanc-Tranchant (chef du SERMA) et Messieurs Alain Aggery (chef du LPEC) et Franck Gabriel (chef du LTSD) de m'avoir accueilli dans leurs laboratoires.

Je tiens aussi à remercier les membres du jury et en particulier Pr. Desesquelles pour m'avoir fait l'honneur de présider le jury. Les professeurs Piero Ravetto et Jean Ragusa ont toute ma reconnaissance pour avoir accepté d'être les rapporteurs de ma thèse. Je les remercie très sincèrement pour leur disponibilité et l'intérêt qu'ils ont porté à ce travail, ainsi que l'ensemble des membres du jury.

Je remercie aussi tout particulièrement Richard, mon directeur de thèse, ainsi que Frédéric et Emiliano, mes encadrants. Ils m'ont donné beaucoup de leur temps et de précieux conseils pour me permettre d'avancer dans mon travail. Tout au long de cette thèse, j'ai pu compter sur leur soutien au quotidien, sur leur excellence dans le domaine et sur leur immense patience (que j'ai dû mettre à l'épreuve quelques fois…). Cette équipe complémentaire m'a permis d'apprendre toujours plus. Je tiens aussi à remercier tous les ingénieurs du SERMA qui m'ont aidé pour le bon déroulement de la thèse.

Je ne pourrai certainement pas oublier Daniele, mon « coloc » de bureau, avec qui j'ai pu partager cette expérience. Ce fut réellement un plaisir de passer cette thèse en sa compagnie.

Finalement, j'ai une pensée pour mes proches, qui m'accompagnent depuis de longues années dans mes études. Un grand merci car sans eux, je ne serais certainement jamais arrivé là.

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

LISTE DES ACRONYMES ET ABREVIATIONS

	Définition	Page
AA	Aragones et Ahnert, fait référence à une formulation du CMFD	80
AFC	Average Flux Correction, fait référence à une formulation du CMFD	80
Colorset	Motif d'assemblages	4
Cellule	Crayon combustible et modérateur qui l'entoure	18
CMFD	Coarse Mesh Finite Difference	73
DD	Décomposition de Domaine	30
DDC	Début De Cycle	153
DDM	Méthode de Décomposition de Domaine	35
Direct	Fait référence à l'algorithme séquentiel classique de calcul transport	15
GB	GigaByte	
HCC	Heterogeneous Cartesian Cell	18
HPC	High Performance Computing	34
IML	Itération Multigroupe Locale	105
Intervalle de confiance	Intervalle à plus ou moins trois écarts types de la valeur moyenne pour un calcul Monte Carlo	145
IPL	Itérations de Puissance Locales	106
KBA	Koch-Baker-Alcouffe, Algorithme de balayage parallèle par front	66
MB	MegaByte	
MCNH	Moon, Cho, Noh et Hong, fait référence à une formulation du CMFD	80
MOC	Method Of Characteristics	15
MOSC	Method of Short Characteristics	17
MPI	Message Passing Interface	30
PB1G	Parallèle par Blocs dans 1 Groupe	35
PBJ	Parallel Block Jacobi	34
PMBG-S	Parallel Multigroup-Block Gauss-Seidel	61
PMBJ	Parallel Multigroup-Block Jacobi	42
RAM	Random Acces Memory.	26
RB	schéma Red & Black	66
REL	Réacteur à Eau Légère	
REP	Réacteur à Eau Pressurisée	
SBM	Simple Balayage Multigroupe	104
S_N	Méthode des ordonnées discrètes	13
T4	TRIPOLI-4®	153
TB	TeraByte	
UC	Unité de Calcul	116
UCE	Unité de Calcul Effective	116
UCG	Unité de Calcul Génératrice	116

Introduction

1. Contexte du travail de thèse et objectifs

Dans le cœur d'un réacteur nucléaire, le transport des neutrons est régi par l'équation du transport de Boltzmann. Cette équation est résolue en discrétisant le problème sur les variables d'espace, d'énergie et de direction angulaire. Le flux neutronique solution de cette équation dépendant d'un grand nombre de variables et les phénomènes physiques à prendre en considération allant de l'échelle du noyau jusqu'aux limites du cœur, il est donc impossible de rechercher directement la solution de cette équation sur le problème complet.

De ce fait, la méthodologie de calcul neutronique standard des cœurs de réacteurs nucléaire est basée sur une approche en deux étapes qui couvre plusieurs échelles. Lors de la première étape de calcul (résolution de quelques cm^3), on résout l'équation du transport sur une géométrie restreinte associée à des maillages fins en énergie et espace (plusieurs centaines de groupes d'énergie et maillage spatial hétérogène) en prenant en compte les spécificités des isotopes (autoprotection). Ce calcul est réalisé pour un assemblage, ou un motif d'assemblages, en milieu périodique et à deux dimensions. Le flux hétérogène en espace et en énergie ainsi calculé permet de produire des sections efficaces homogénéisées en espace et condensées en énergie qui seront utilisées dans la seconde étape de calcul. Cette dernière permet de résoudre le problème 3D à l'échelle du cœur sur des maillages plus larges en énergie et en espace (typiquement, dans les schémas de conception, il s'agit de quelques groupes d'énergie et d'un maillage spatial homogène par quart d'assemblage ou par crayon). Il s'agit du calcul du cœur du réacteur en théorie de la diffusion ou du transport avec un nombre de groupes en énergie réduit. Les deux étapes de calcul sont reliées via une base de données *ad-hoc* par milieu (crayon ou assemblage par exemple) appelée *bibliothèque de sections efficaces multi-paramétrées*. Afin de couvrir la gamme de fonctionnement du cœur d'un réacteur nucléaire (fonctionnement au cours du cycle et lors des transitoires accidentels), un grand nombre de calculs est réalisé afin de produire les sections efficaces dépendant des paramètres de fonctionnement. Les sections efficaces ainsi produites lors de la première étape, *calcul réseau*, et utilisées dans le seconde, *calcul cœur*, dépendent de divers paramètres tels que le taux de combustion, la température du combustible, la densité et la température du modérateur ou encore la concentration en bore soluble dans le caloporteur. Les sections efficaces en chaque point du cœur sont obtenues par interpolation au sein de la bibliothèque de sections efficaces multi-paramétrées.

Jusqu'à aujourd'hui, l'effort de réduction des biais de modélisation pour le calcul des cœurs de réacteurs a été majoritairement orienté vers une amélioration simultanée des modèles de calcul réseau et cœur. Ainsi, l'amélioration des modèles

d'autoprotection des sections efficaces, l'amélioration des solveurs de calcul du flux et l'augmentation du nombre de groupes en énergie ont permis de réduire les biais de calcul lors de l'étape réseau. En parallèle, l'augmentation du nombre de groupes en énergie, le calcul en théorie du transport, le raffinement dans la description géométrique du cœur, entre autres, ont permis de réduire les biais de calcul lors de l'étape cœur.

Toutefois, ces améliorations dans la modélisation ont toujours été réalisées en conservant une structure d'échange, la bibliothèque de sections efficaces multi-paramétrées, figée entre les deux étapes de calcul. Il n'y a alors pas de rétroaction entre les étapes réseau et cœur pour l'obtention des sections efficaces multigroupes aux conditions locales réelles du calcul. Ce découplage des deux étapes de calcul est à l'origine de biais méthodologiques notamment dans le cas où le cœur du réacteur présente de fortes hétérogénéités : l'hypothèse de périodicité (mode fondamental) retenue pour le calcul réseau est moins licite. Les hétérogénéités peuvent être des hétérogénéités de matière (différents types de combustibles dans un réacteur de puissance, hétérogénéités de densité du modérateur qui est également le caloporteur), des hétérogénéités géométriques (cœur de petite taille, réacteurs d'irradiation…) ou les deux simultanément (interface cœur-réflecteur dans les réacteurs à eau légère…).

L'objectif de cette thèse est de proposer une démarche de modélisation des cœurs de réacteur basée sur une approche innovante permettant d'intégrer directement les spécificités physiques locales dans les modélisations 3D détaillées. L'intérêt et la motivation d'une nouvelle approche dans la modélisation des cœurs est l'accès à une plus grande représentativité des grandeurs calculées. L'apport essentiel est l'obtention directe des grandeurs locales en même temps que les grandeurs globales conduisant à une évaluation plus aisée des incertitudes (en limitant les hypothèses de découplage).

La solution initialement envisagée pour obtenir ces résultats de « référence » est de considérer les assemblages de l'étape du calcul réseau dans leur environnement réel dans le cœur. Pour cela, il faut connaitre les conditions aux limites du calcul et donc avoir la solution à l'échelle du cœur. On entre alors dans un processus itératif où les calculs réseaux des assemblages fournissent les sections efficaces pour le calcul de cœur qui peut en retour donner les nouvelles conditions aux limites des assemblages. Le schéma itératif s'arrête alors lorsque les conditions aux limites des calculs réseaux sont convergées. Ainsi, le processus de génération des sections efficaces pour le calcul de cœur se trouve amélioré : la précision de la méthode dépend toutefois de la manière dont les conditions aux limites des calculs réseaux sont déterminées. Nous avons ainsi commencé par coupler les calculs réseaux sans insérer d'approximation à l'interface entre les assemblages, ce qui nous a naturellement amené au formalisme des méthodes de décomposition de domaine spatiale : chaque assemblage est alors un sous domaine. L'un des nombreux avantages de ces méthodes est qu'elles amènent naturellement au parallélisme, permettant d'accélérer la résolution du problème. De plus, elles permettent de diviser un problème de grande taille en plusieurs sous problèmes de

petite taille, plus faciles à résoudre et plus adaptés aux capacités des machines de calcul.

Ainsi, ce travail de thèse est dédié à la mise en œuvre d'un algorithme basé sur la décomposition de domaine permettant d'accéder aux grandeurs locales tout en prenant en compte les hétérogénéités, pour des problèmes de grande taille allant jusqu'à l'échelle du cœur en 3D.

Cette thèse s'inscrit dans le cadre des travaux de développement du code multifilières APOLLO3® [1]. Cet environnement de développement impose un certain nombre de prérequis concernant la démarche à mettre en œuvre. Tout d'abord, notre réalisation doit être portable et ne peut donc pas être optimisée sur une machine de calcul particulière. De plus, le nombre de processeurs à considérer est compris entre une dizaine (ordinateurs de bureau) et quelques milliers (calculateurs disponibles dans les laboratoires d'études), les architectures HPC n'étant pas toujours accessibles pour les utilisateurs d'APOLLO3. De cette contrainte dérive aussi un besoin essentiel d'optimisation de l'utilisation de la mémoire. Pour ces raisons, notre choix s'est porté sur un algorithme de résolution permettant de minimiser l'impact de l'implémentation parallèle sur les performances de la réalisation.

Dans ce document, nous nous attacherons plus particulièrement aux performances de l'algorithme de résolution lui-même en accordant une importance moindre à la qualité de l'implémentation parallèle. En effet, afin d'être performant, l'algorithme doit tout d'abord présenter des propriétés de convergence (rayon spectral) similaires à celles de l'algorithme de résolution de l'équation du transport classique. Ainsi, une grande partie de nos études portent sur des calculs séquentiels permettant de comparer l'algorithme développé à l'algorithme classique. De cette manière, nous pouvons séparer les différents effets et réaliser des tests élémentaires permettant d'identifier les manques et les avantages de l'algorithme comme l'ordre de convergence ou l'efficacité de l'utilisation des ressources de calcul. Ce n'est que lorsque les performances permettant d'envisager une scalabilité idéale seront atteintes, grâce à des méthodes d'accélération, que nous commencerons à investiguer le parallélisme. Aussi la part dédiée à l'implémentation parallèle est présentée à la fin du document.

Finalement, au cours de ce travail de thèse, nous n'avons pas limité notre démarche à tester l'algorithme sur des applications simples qui peuvent masquer les réelles difficultés des calculs haute-fidélité. La démonstration des capacités et des performances de l'algorithme développé a été faite jusqu'à des applications de calculs de cœur ou de motifs d'assemblages 2D et 3D avec les contraintes de discrétisation spatiales et énergétiques adaptées aux calculs haute-fidélité.

2. Organisation du document

Le Chapitre 1 est dédié à la présentation des éléments de neutroniques permettant de définir les notations utilisées dans cette thèse. Tout d'abord nous donnons des notions générales sur le transport des neutrons dans le domaine de la physique des réacteurs (§1.2). L'équation du transport est introduite (§1.3) ainsi que la méthode

standard pour la résoudre (§1.4). Finalement, une section (§1.5) est dédiée à la description de la méthode des caractéristiques courtes du solveur IDT [2] [3] utilisé pour la méthode de décomposition de domaine réalisée dans cette thèse.

Le Chapitre 2 est dédié à la parallélisation de la résolution de l'équation du transport. Il commence par un état des lieux (§2.2). Tout d'abord, une revue des moyens de calcul actuels est faite (§2.2.1) en présentant les architectures des calculateurs et les paradigmes de parallélisation. Le §2.2.2 résume l'état de l'art des méthodes de parallélisation de la résolution de l'équation du transport puis sont présentées, de manière générale, les méthodes de décomposition de domaine (§2.2.3). Ensuite, nous introduisons l'approche proposée dans ce travail de thèse (§2.3) avant de poser précisément le formalisme de la méthode implémentée (§2.4). Ce chapitre se termine par une comparaison entre la méthode développée et la méthode classique de calcul transport déterministe (§2.5) dans le cadre du calcul d'un motif d'assemblages (appelé *colorset*) de type Réacteurs à Eau Pressurisé (REP) en 2D.

Le Chapitre 3 présente les méthodes d'accélération mises en œuvre pour accélérer l'algorithme initial. Pour cela, nous avons tout d'abord cherché à modifier l'algorithme de manière à transmettre au mieux la solution dans le domaine de calcul (§3.2). La seconde approche (§3.3) est une méthode d'accélération non linéaire, appelée *Coarse Mesh Finite Différence* (CMFD). Elle permet d'accélérer la transmission de la solution à travers le domaine de calcul global, mais aussi d'accélérer la convergence des sources. Cette méthode est présentée en détail et ses performances sont analysées sur des configurations de colorsets d'assemblages types REP. Finalement, une troisième méthode (§3.4) modifiant l'algorithme de résolution des sous domaines est présentée. Elle vise à accélérer la convergence des sources volumiques dans les sous domaines.

Le Chapitre 4 décrit l'architecture de la maquette et les méthodes de parallélisation qui ont été implémentées. Dans un premier temps (§4.2), nous détaillons la manière dont sont organisées les données dans la maquette développée avec le solveur IDT. Nous associons à cette présentation des tests illustratifs. Ensuite deux méthodes de parallélisation sont présentées : le parallélisme à mémoire partagée, illustrée par des mesures de performances parallèles (§4.3) et le parallélisme hybride mémoire partagée et distribuée (§4.4).

Le Chapitre 5 est dédié à l'analyse des performances de la méthode et aux comparaisons avec des calculs de référence. Dans un premier temps (§5.2) nous comparons les performances de la méthode pour le cas du benchmark C5G7 à 2D et 3D [4, 5]. Ensuite nous présentons un calcul de cœur de Réacteur à Eau Légère (REL) en 2D sans homogénéisation spatiale (§5.3). La distribution des taux de fission par crayon est comparée à la solution d'un calcul de référence Monte Carlo réalisé avec le code TRIPOLI-4® [6]. Le calcul d'un colorset d'assemblages en 3D (hauteur caractéristique d'un REP) est ensuite présenté (§5.4) afin de démontrer les nouvelles capacités de calcul transport 3D. Finalement, nous illustrons les capacités et les

limitations du solveur avec le calcul d'un grand cœur de REL complet 3D sans homogénéisation, mais avec un nombre restreint de groupes d'énergie (§5.5).

La dernière section conclut cette thèse et présente un panorama des pistes ayant retenu notre intérêt pour des recherches futures.

Chapitre 1. Méthodes numériques pour la modélisation du transport des neutrons

1.2 Introduction au transport des neutrons

La neutronique [7], ou transport des neutrons, consiste à étudier le cheminement des neutrons dans la matière et les réactions qu'ils y induisent. Cette section est dédiée à l'introduction de concepts de base pour la compréhension des modèles utilisés dans le cadre des calculs de réacteurs nucléaires. Un neutron (de masse m_n) est complètement caractérisé par les variables suivantes :

- $\boldsymbol{r}$: sa position dans l'espace.
- $\boldsymbol{\Omega}$: la direction de sa propagation dans l'espace.
- E : son énergie cinétique qui caractérise le module de sa vitesse $v = \sqrt{2E/m_n}$.
- t : l'instant considéré.

L'ensemble de ces données défini l'espace des phases. La grandeur utilisée pour décrire la population de neutrons est la densité neutronique $n(\boldsymbol{r}, \boldsymbol{\Omega}, E, t)$. Elle définit le nombre de neutrons contenue dans l'espace des phases $d^3\boldsymbol{r}d^2\boldsymbol{\Omega}dE$ à l'instant t. Il est plus pratique d'utiliser une autre grandeur que la densité de neutrons, le flux neutronique, $\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t)$, qui est exprimé comme le produit de la densité de neutrons par leur vitesse :

$$\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) = vn(\boldsymbol{r}, \boldsymbol{\Omega}, E, t).$$

Deux autres grandeurs sont aussi couramment utilisées en neutroniques. On définit ainsi le flux scalaire :

$$\phi(\boldsymbol{r}, E, t) = \int_{4\pi} \psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) d^2\boldsymbol{\Omega},$$

où 4π est la mesure de la sphère unité des directions S^2, ainsi que le courant :

$$\boldsymbol{J}(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) = \boldsymbol{\Omega}\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t),$$

dont le produit scalaire par $\boldsymbol{n}d^2\boldsymbol{r}$ donne le nombre de neutrons avec l'énergie E et la direction $\boldsymbol{\Omega}$ traversant la surface $d^2\boldsymbol{r}$ par unité de temps avec $\boldsymbol{n}$ la normale à la surface.

INTERACTION DES NEUTRONS AVEC LA MATIERE

Dans le cœur du réacteur, la population de neutrons est beaucoup moins dense que celle des molécules constituant la matière dans laquelle ils se propagent ($\leq 10^{15}$ neutrons/cm^3 contre ~10^{23} molécules/cm^3). Les interactions neutron-neutron sont donc

très peu probables. De ce fait, elles sont négligées pour la physique des réacteurs. De plus le temps moyen de désintégration spontanée d'un neutron est largement supérieur à sa durée de vie moyenne dans le cœur (~10 minutes contre 10^{-3}s à 10^{-7}s selon le type de réacteur). La décroissance radiative des neutrons est donc elle aussi négligée. Nous ne prenons donc en compte que l'interaction des neutrons avec la matière dans laquelle ils se propagent. La probabilité d'interaction des neutrons avec le milieu est exprimée par le biais de sections efficaces macroscopiques. La section efficace macroscopique $\Sigma(\boldsymbol{r}, \boldsymbol{\Omega}, E, t)$ est la probabilité par unité de distance parcourue par un neutron d'interagir avec le milieu qu'il traverse. Elle est calculée de la manière suivante :

$$\Sigma(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) = \sum_{is=1}^{A} \sigma_{is}(\boldsymbol{r}, \boldsymbol{\Omega}, E) \times n_{is}(\boldsymbol{r}, t).$$

Ici A est le nombre d'isotopes dans le milieu de propagation. $n_{is}(\boldsymbol{r}, t)$ est la densité de noyaux et $\sigma_{is}(\boldsymbol{r}, \boldsymbol{\Omega}, E)$ est la section efficace microscopique. La section efficace microscopique représente une « surface d'interaction » entre le neutron et le noyau et est généralement exprimée en barns (1b = 10^{-24}cm^2). Elle est déterminée par des mesures expérimentales et est considérée comme une donnée d'entrée pour la résolution de l'équation du transport. Il existe ainsi des bases de données régulièrement enrichies comme JEFF [8] ou ENDF [9] permettant d'avoir accès aux constantes nucléaires.

La dépendance angulaire de la section efficace microscopique permet de prendre en compte l'anisotropie du milieu. Dans le contexte des réacteurs nucléaires, les matériaux présents peuvent être considérés comme isotropes et de ce fait, la dépendance angulaire de la section efficace microscopique peut être négligée.

Les sections efficaces macroscopiques permettent de définir les probabilités des différents évènements pouvant se produire. Parmi ces évènements on relèvera en particulier la fission (Σ_f), l'absorption (Σ_a) et la diffusion élastique ou inélastique (Σ_s). La probabilité d'avoir un évènement quelconque est caractérisée par la section efficace totale (Σ_t). On introduit maintenant la notion de taux de réaction τ_i qui exprime le nombre d'évènements par unité de temps et de volume. Pour une réaction i on a l'expression suivante :

$$\tau_i(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) = \Sigma_i(\boldsymbol{r}, E, t)\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t).$$

Par exemple $\tau_f(\boldsymbol{r}, \boldsymbol{\Omega}, E, t) = \Sigma_f(\boldsymbol{r}, E, t)\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E, t)$ est le nombre de fissions par seconde dans le volume $d^3\boldsymbol{r}d^2\boldsymbol{\Omega}dE$. Pour les collisions de diffusion élastique ou inélastique (*scattering*), le taux de réaction est décrit de la manière suivante :

$$\tau_s(\boldsymbol{r}, \boldsymbol{\Omega}'.\boldsymbol{\Omega}, E' \to E, t) = \Sigma_s(\boldsymbol{r}, \boldsymbol{\Omega}'.\boldsymbol{\Omega}, E' \to E, t)\psi(\boldsymbol{r}, \boldsymbol{\Omega}', E', t).$$

Il s'agit du taux de réaction modifiant les neutrons de la position dans l'espace des phases $(\boldsymbol{r}, \boldsymbol{\Omega}', E', t)$ vers la position $(\boldsymbol{r}, \boldsymbol{\Omega}, E, t)$. L'angle entre les directions $\boldsymbol{\Omega}'$ et $\boldsymbol{\Omega}$ suffit à décrire la diffusion dans un milieu isotrope.

Les taux de réaction intégrés en énergie et en angle sont les grandeurs d'intérêt dans l'étude des réacteurs. Par exemple le taux de fission intégré en énergie et en angle multiplié par l'énergie dégagée par fission permet de connaitre la puissance dans un volume donné, et d'en déduire la distribution spatiale de puissance dans le cœur, ainsi que la position du point chaud qui est porte critère de sûreté. Ils permettent aussi de représenter d'autres phénomènes comme l'usure sous flux des composants du cœur ou de l'évolution isotopique des milieux. La détermination des taux de réaction est obtenue par la résolution de l'équation du transport.

1.3 Equation du transport

L'équation du transport fait état du bilan de neutrons dans le volume $d^3\boldsymbol{r}d^2\boldsymbol{\Omega}dE$ entre les instants t et $t+dt$. Elle décrit statistiquement l'interaction des neutrons avec la matière :

$$\frac{1}{v}\partial_t\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) + L\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) = H\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) + F\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) + S_{ext}(\boldsymbol{r},\boldsymbol{\Omega},E,t)\,. \tag{1.1}$$

Cette équation permet de connaitre la distribution des neutrons sur le domaine d'étude. Elle est complétée par des conditions initiale $\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t=0)$ et la condition aux limites notées $\psi^-(\boldsymbol{r},\boldsymbol{\Omega},E,t)$. Cette dernière est le flux angulaire défini pour les directions $\boldsymbol{\Omega}$ entrantes sur la surface notée Γ du domaine. On ne fait pas apparaitre le phénomène de neutrons retardés qui compliquerait inutilement le formalisme développé dans le cadre de cette thèse. Les différents termes qui composent l'équation du transport sont les suivants :

- $\frac{1}{v}\partial_t\,\psi$ est la variation du flux par rapport au temps. Ce terme est nul dans le cas du régime stationnaire.
- L est l'opérateur qui décrit les disparitions de neutrons :

$$L\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) = \big(\boldsymbol{\Omega}\cdot\boldsymbol{\nabla} + \Sigma_t(\boldsymbol{r},E,t)\big)\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t)\,. \tag{1.2}$$

  Le premier terme (streaming) $\boldsymbol{\Omega}\cdot\boldsymbol{\nabla}\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t)$ décrit les fuites de neutrons par la surface du volume $d^3\boldsymbol{r}$.

  Le second $\Sigma_t(\boldsymbol{r},E,t)\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t)$ est le taux de réaction décrivant la disparition des neutrons par interaction avec le milieu. Σ_t est la section efficace totale du milieu.

- H est l'opérateur de transfert, couramment appelé opérateur de *scattering*. Il décrit l'ensemble des neutrons qui arrivent dans $(\boldsymbol{\Omega},E)$ par diffusion. Il est exprimé de la manière suivante :

$$H\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) = \int_{4\pi} d\boldsymbol{\Omega}' \int_0^\infty dE'\Sigma_s(\boldsymbol{r},\boldsymbol{\Omega}'.\boldsymbol{\Omega},E'\to E,t)\psi(\boldsymbol{r},\boldsymbol{\Omega}',E',t). \tag{1.3}$$

Nous notons ici que le scattering ne dépend que du produit scalaire $\boldsymbol{\Omega}'.\boldsymbol{\Omega}$. Ceci est correct pour les milieux isotropes, ce qui est généralement le cas pour les matériaux utilisés dans les réacteurs nucléaires dans les conditions d'exploitation.

- F est l'opérateur de fission. Il permet de comptabiliser dans le bilan les neutrons produits par fission :

$$F\psi(\boldsymbol{r},\boldsymbol{\Omega},E,t) = \sum_{is}^{N_f} \frac{\chi_{is}(E)}{4\pi} \int_0^\infty dE'\, \upsilon_{is}(E')\Sigma_{f,is}(\boldsymbol{r},E',t)\psi(\boldsymbol{r},\boldsymbol{\Omega},E',t)\,, \tag{1.4}$$

où N_f est le nombre d'isotopes fissiles du milieu, υ_{is} est le nombre moyen de neutrons émis par fission et χ_{is} est le spectre d'émission des neutrons par fission.

- $S_{ext}(\boldsymbol{r},\boldsymbol{\Omega},E,t)$ est une source extérieure.

1.3.1 Evolution temporelle et approximation du régime stationnaire

L'exposition des matériaux à un flux de neutrons entraine des réactions nucléaires avec les noyaux des atomes qui composent la matière. Les caractéristiques des matériaux sont alors modifiées. Les réactions de capture neutronique et de fission modifient la composition isotopique des matériaux et provoquent l'apparition et la disparition de nucléides. Parmi eux, la plupart sont instables et sujets à la décroissance radioactive. Ainsi, l'état du combustible varie au cours du fonctionnement d'un réacteur et la modification des concentrations isotopiques entraine la modification des sections efficaces macroscopiques impliquées dans l'équation du transport. Cependant les modifications des concentrations sont suffisamment lentes (en comparaison avec le temps d'émission et de vie d'un neutron) pour pouvoir faire l'approximation du régime stationnaire dans la résolution de l'équation du transport. L'évolution des concentrations isotopiques est réalisée par la résolution des équations de Bateman [10]. Le calcul de flux et le calcul d'évolution isotopique sont traités le plus souvent par des méthodes de couplage explicites.

Par cette approximation, il est possible de décrire le fonctionnement d'un réacteur à un moment donné en considérant l'équation stationnaire du transport sans source externe. Pour que le problème stationnaire ait une solution non triviale, nous sommes amenés à considérer un problème à valeur propre :

$$\begin{cases} L\psi(\boldsymbol{r},\boldsymbol{\Omega},E) = H\psi(\boldsymbol{r},\boldsymbol{\Omega},E) + \dfrac{1}{\lambda}F\psi(\boldsymbol{r},\boldsymbol{\Omega},E), & \boldsymbol{r}\in D, \\ \psi^-(\boldsymbol{r},\boldsymbol{\Omega},E) = \beta\psi^+(\boldsymbol{r},\boldsymbol{\Omega},E), & \boldsymbol{r}\in \Gamma, \end{cases} \tag{1.5}$$

où D est le domaine étudié et Γ sa surface. Ce problème est complété par les conditions aux limites du flux ψ^- à la surface Γ du domaine. Il s'agit soit de condition de vide pour $\beta = 0$ soit de condition d'albédo pour $\beta \neq 0$. Cette dernière représente la fraction du flux sortant à travers la surface Γ réémis vers l'intérieur du domaine. Le

flux sortant du domaine, noté $\psi^+(\boldsymbol{r},\boldsymbol{\Omega},E)$, est le flux angulaire défini pour les directions $\boldsymbol{\Omega}$ sortantes sur la surface du domaine.

L'équation (1.5) a une infinité de solutions. En effet, si ψ est une solution alors $(k \times \psi\,, k \in \mathbb{R}^+)$ est aussi solution. Elle ne permet donc pas de déterminer le niveau de flux. Ce dernier peut par exemple être donné par la puissance dégagée dans le domaine. La puissance est définie par le taux de fission multiplié par l'énergie dégagée par fission et intégré en énergie, intégré en angle et intégré sur tout le domaine.

La plus grande valeur propre, λ, est appelée facteur de multiplication effectif et est souvent notée k_{eff}. Ce dernier exprime le facteur par lequel le nombre de fissions se trouve multiplié d'une génération de neutrons à la suivante. Si k_{eff} est plus grand que 1 le système est alors sur-critique et la population de neutrons croît. Quand k_{eff} est inférieur à 1 le système est sous-critique et la population de neutrons décroît. Enfin quand k_{eff} est égal à 1, le système est critique.

Dans la suite, nous ne nous intéresserons uniquement à la formulation stationnaire de l'équation du transport présenté à l'équation (1.5).

1.3.2 Développement de l'opérateur de scattering

L'opérateur de scattering est développé sur la base des harmoniques sphériques réelles [11] :

$$A_{m,l}(\boldsymbol{\Omega}) = A_{m,l}(\mu,\varphi) = \begin{cases} \alpha_l^m P_l^m(\mu)\cos m\varphi\,, & m \geq 0, \\ \alpha_l^{|m|} P_l^{|m|}(\mu)\sin|m|\varphi\,, & m \geq 0. \end{cases}$$

$P_l^m(\mu)$ sont les fonctions de Legendre et α_l^m sont des constantes de normalisation :

$$P_l^m(\mu) = \sqrt{1-\mu^2}\,\frac{d^m P_l(\mu)}{d\mu^m}\,,$$

$$\alpha_l^m = \sqrt{(2-\delta_{m0})\,\frac{(l-m)!}{(l+m)!}}\,,$$

où $P_l(\mu)$ sont les polynômes de Legendre d'ordre l :

$$P_l(\mu) = \frac{1}{2^l l!}\frac{d^l}{d\mu^l}(\mu^2-1)^l\,.$$

Ces harmoniques réelles satisfont la relation d'orthogonalité suivante :

$$\frac{1}{4\pi}\int_{4\pi} d\boldsymbol{\Omega}\; A_{l,m}(\boldsymbol{\Omega})A_{l',m'}(\boldsymbol{\Omega}) = \frac{1}{2l+1}\delta_{ll'}\delta_{mm'}\,,$$

et la relation de fermeture :

$$P_l(\boldsymbol{\Omega}'.\boldsymbol{\Omega}) = \sum_{m=-l}^{l} \mathrm{A}_{m,l}(\boldsymbol{\Omega})\mathrm{A}_{m,l}(\boldsymbol{\Omega}')\;.$$

Les sections efficaces de scattering sont développées sur les polynômes de Legendre jusqu'à l'ordre L, qui est le degré d'anisotropie du transfert pris en compte pour le calcul :

$$\Sigma_s(\boldsymbol{r}, \boldsymbol{\Omega}'.\boldsymbol{\Omega}, E' \to E) = \frac{1}{4\pi} \sum_{l=0}^{L} \Sigma_{s,l}(\boldsymbol{r}, E' \to E) P_l(\boldsymbol{\Omega}'.\boldsymbol{\Omega}), \tag{1.6}$$

De ce fait, puisque $P_l(\boldsymbol{\Omega}'.\boldsymbol{\Omega}) = \sum_{m=-l}^{l} \mathrm{A}_{m,l}(\boldsymbol{\Omega}) \mathrm{A}_{m,l}(\boldsymbol{\Omega}')$, on obtient la formulation suivante de l'opérateur de scattering en substituant (1.6) dans (1.3) :

$$H\psi(\boldsymbol{r}, \boldsymbol{\Omega}, E) = \sum_{l=0}^{L} \sum_{m=-l}^{l} A_{m,l}(\boldsymbol{\Omega}) \int_0^{\infty} dE' \Sigma_{s,l}(\boldsymbol{r}, E' \to E) \varphi_{l,m}(\boldsymbol{r}, E'), \tag{1.7}$$

où :

$$\varphi_{l,m}(\boldsymbol{r}, E') = \frac{1}{4\pi} \int_{4\pi} d\boldsymbol{\Omega}' \, A_{l,m}(\boldsymbol{\Omega}') \psi(\boldsymbol{r}, \boldsymbol{\Omega}', E') \tag{1.8}$$

sont les moments angulaires du flux.

1.4 Résolution de l'équation du transport

Deux approches différentes permettent d'accéder à une estimation de la solution de cette équation :

1. L'approche probabiliste par la méthode de Monte Carlo. Elle est basée sur l'estimation de valeurs moyennes des grandeurs d'intérêt obtenues en calculant un grand nombre de trajectoires de neutrons dans le système à étudier. Ces trajectoires sont simulées de manière aléatoire en respectant les lois qui régissent le déplacement des neutrons ainsi que les probabilités d'interaction avec les milieux traversés. Le point fort de cette méthode est qu'elle ne requiert a priori aucune approximation ni sur la géométrie, ni sur la physique mise en jeu. Cependant pour atteindre la précision requise, il faut simuler un très grand nombre de trajectoires (en lien avec la taille du problème et la précision requise), ce qui limite dans certains domaines son utilisation à des calculs de références. De plus pour un calcul de cœur en évolution, la propagation des incertitudes statistique rend le système encore plus long à faire converger.
2. L'approche déterministe consiste à considérer uniquement la valeur statistique moyenne du flux en chaque point de l'espace des phases. La solution de l'équation du transport (1.1) est discrétisée sur chacune des variables $(\boldsymbol{r}, \boldsymbol{\Omega}, E)$. Elle est obtenue en approximant les grandeurs neutroniques (flux et sections efficaces). Les approximations de cette discrétisation sont l'objet des paragraphes suivants.

1.4.1 Discrétisation énergétique : approche multigroupe

Avec les méthodes déterministes, la discrétisation de la variable énergétique est réalisée par le formalisme multigroupe. Ce dernier consiste à remplacer les variables continues en énergie (sections efficaces et flux) en variables discrétisées en domaines

d'énergie appelés groupes. L'espace en énergie est discrétisé en un nombre fini N_g de groupes à l'intérieur desquels les grandeurs neutroniques sont constantes et moyennées sur l'intervalle d'énergie ΔE_g associée au groupe.

L'objet de la résolution de l'équation du transport étant de calculer les taux de réaction, il faut s'assurer que l'approximation de la mise en groupes conserve les taux de réaction obtenus avec les sections continues en énergie. Les sections efficaces multigroupes sont définies de la manière suivante :

$$\Sigma^g(\boldsymbol{r},\boldsymbol{\Omega}) = \frac{\int_{E\in\Delta E_g} dE\ \Sigma(\boldsymbol{r},E)\psi(\boldsymbol{r},\boldsymbol{\Omega},E)}{\int_{E\in\Delta E_g} dE\ \psi(\boldsymbol{r},\boldsymbol{\Omega},E)}.$$

Cette définition amène alors deux problématiques. La première est que les sections efficaces macroscopiques ainsi définies dépendent de $\boldsymbol{\Omega}$ même pour un milieu isotrope, ce qui implique un problème multigroupe plus complexe à résoudre, les sections macroscopiques devenant anisotropes. La seconde est que la valeur de la section dépend du flux, qui est justement l'objet de la résolution de l'équation du transport. Pour les isotopes dont la section efficace varie lentement avec l'énergie, il est possible de remplacer le flux par un spectre avec des groupes assez fins, qui dépend du type de réacteur nucléaire. Cependant, il faut porter une attention particulière aux isotopes résonnants pour lesquels les sections efficaces peuvent varier de plusieurs ordres de grandeur dans un intervalle d'énergie très petit. De ce fait, à moins d'utiliser un nombre de groupe très élevé (~10^5), un traitement particulier doit être réalisé. Il s'agit du processus d'autoprotection [12, 13, 14]. Ce formalisme, que nous n'aborderons pas ici, permet de produire les sections efficaces spécifiques à chaque problème traité.

Dans le formalisme multigroupe, le flux est exprimé par l'ensemble discret :

$$\psi(\boldsymbol{r},\boldsymbol{\Omega},E) \equiv \left\{\psi^g(\boldsymbol{r},\boldsymbol{\Omega})\ ,\qquad 1 \le g \le N_g\right\},$$

et le flux scalaire :

$$\phi^g(\boldsymbol{r}) = \int_{4\pi} d\boldsymbol{\Omega}\,\psi^g(\boldsymbol{r},\boldsymbol{\Omega})\ .$$

FORMULATION MULTIGROUPE DE L'EQUATION DU TRANSPORT

Le formalisme multigroupe amène à un système d'équations couplées entre elles par un terme source. L'équation (1.5) devient :

$$\forall g \in \{1,\dots,N_g\} : \begin{cases} (L^g - H^{g\to g})\psi^g(\boldsymbol{r},\boldsymbol{\Omega}) = Q^g_{ext}, & \boldsymbol{r}\in D, \\ \psi^{g,-}(\boldsymbol{r},\boldsymbol{\Omega}) = \beta\psi^{g,+}(\boldsymbol{r},\boldsymbol{\Omega}), & \mathbf{r}\in\Gamma. \end{cases} \qquad (1.9)$$

où L^g et H^gsont respectivement les opérateurs de transport et de self-scattering dans le groupe g :

$$\begin{cases} L^g\psi^g(\boldsymbol{r},\boldsymbol{\Omega}) = \left(\boldsymbol{\Omega}\cdot\boldsymbol{\nabla} + \Sigma_t^g(\boldsymbol{r})\right)\psi^g(\boldsymbol{r},\boldsymbol{\Omega}), \\ H^{g\to g}\psi^g(\boldsymbol{r},\boldsymbol{\Omega}) = \int_{4\pi} d\boldsymbol{\Omega}'\Sigma_s^g(\boldsymbol{r},\boldsymbol{\Omega}'.\boldsymbol{\Omega})\psi^g(\boldsymbol{r},\boldsymbol{\Omega}'). \end{cases}$$

La source Q_{ext}^g couple les équations en prenant en compte la contribution des autres groupes d'énergie :

$$Q_{ext}^g = \left(\tilde{H}\psi\right)^g + \frac{1}{\lambda}(F\psi)^g.$$

Elle est composée de la source de scattering et la source de fission définis par :

$$\begin{cases} \left(\tilde{H}\psi\right)^g = \sum_{g'\neq g}^{N_g}\int_{4\pi} d\boldsymbol{\Omega}'\Sigma_s^{g'\to g}(\boldsymbol{r},\boldsymbol{\Omega}'.\boldsymbol{\Omega})\psi^{g'}(\boldsymbol{r},\boldsymbol{\Omega}'), \\ (F\psi)^g = \sum_{is}^{N_f}\frac{\chi_{is}^g}{4\pi}\sum_{g'}^{N_g}\left(\nu\Sigma_f\right)_{is}^{g'}(\boldsymbol{r})\phi^{g'}(\boldsymbol{r}). \end{cases}$$

1.4.2 Discrétisation angulaire

Nous nous plaçons maintenant dans le contexte de la résolution d'un problème monocinétique relatif à un groupe g quelconque pour aborder le traitement de la variable angulaire :

$$\begin{cases} (L-H)\psi(\boldsymbol{r},\boldsymbol{\Omega}) = Q_{ext}, & \boldsymbol{r}\in D, \\ \psi^-(\boldsymbol{r},\boldsymbol{\Omega}) = \beta\psi^+(\boldsymbol{r},\boldsymbol{\Omega}), & \boldsymbol{r}\in\Gamma, \end{cases} \tag{1.10}$$

où Q_{ext} est la source provenant des autres groupes (l'indice g est omis pour alléger les notations). Il existe plusieurs possibilités pour discrétiser la variable angulaire, parmi lesquelles les méthodes P_N et S_N.

La méthode P_N consiste à développer le flux angulaire sur une série limitée d'harmoniques sphériques. Les grandeurs caractéristiques sont les moments angulaires du flux qui représentent les composants du flux selon la base d'harmoniques sphériques. Le système d'équations à résoudre comporte un couplage entre les moments angulaires. La méthode P_N donne lieu à des systèmes d'équations de grande taille, difficiles à résoudre et coûteux en stockage mémoire. De plus l'ordre du développement du flux a besoin d'être relativement important pour prendre correctement en compte l'anisotropie du flux. Le gain est que l'effet de raies que l'on peut rencontrer dans les méthodes S_N est complètement éliminé.

La méthode des ordonnées discrètes (S_N) présentée ici est celle utilisée par le solveur IDT utilisé dans ce travail. Elle consiste à choisir un ensemble fini de directions et des poids associé ω_d (quadrature):

$$\{\boldsymbol{\Omega}_d;\omega_d\}_{d=1,\dots,N_d} : \sum_{d=1,\dots,N_d}\omega_d = 1,$$

où N_d est le nombre de directions discrètes engendrées par la quadrature S_N d'ordre N.

Le flux angulaire est défini de manière discrète pour chaque direction de la quadrature :

$$\psi_d(\boldsymbol{r}) = \psi(\boldsymbol{r}, \boldsymbol{\Omega_d}).$$

L'intégration angulaire est ainsi remplacée par la somme du flux sur chaque direction de la quadrature pondéré par le poids associé:

$$\phi(\boldsymbol{r}) = \int_{4\pi} d\boldsymbol{\Omega}\, \psi(\boldsymbol{r}, \boldsymbol{\Omega}) \approx \sum_{d}^{N_d} \omega_d \psi_d(\boldsymbol{r})\,.$$

Dans chaque groupe, on doit résoudre un système d'équations couplées par la source de self-scattering $q_d(\boldsymbol{r})$:

$$\forall\, d \in \{1, \dots, N_d\} : \begin{cases} L\psi_d(\boldsymbol{r}) = q_d(\boldsymbol{r}) + Q_{ext}(\boldsymbol{r}, \boldsymbol{\Omega_d}), & \boldsymbol{r} \in D, \\ \psi_d(\boldsymbol{r}) = \psi_d^-(\boldsymbol{r}) \quad , & \boldsymbol{r} \in \Gamma. \end{cases}$$

Où $\psi_d^-(\boldsymbol{r})$ est le flux entrant et $q_d(\boldsymbol{r})$ est la source de self-scattering :

$$q_d(\boldsymbol{r}) = \sum_{l=0}^{L} \Sigma_{s,l}(\boldsymbol{r}) \sum_{m=-l}^{l} A_{l,m}(\boldsymbol{\Omega_d}) \varphi_{l,m}(\boldsymbol{r}),$$

avec les moments du flux :

$$\varphi_{l,m}(\boldsymbol{r}) = \frac{1}{4\pi} \sum_{d}^{N_d} \omega_d \mathrm{A}_{l,m}(\boldsymbol{\Omega_d}) \psi_d(\boldsymbol{r})\,.$$

1.4.3 Discrétisation spatiale

Les méthodes de discrétisation de la variable spatiale sont nombreuses. Toutes les méthodes sont basées sur la partition du domaine de calcul en régions homogènes. La plupart des méthodes numériques pour le traitement de la variable spatiale sont de type projectif, c'est-à-dire que le flux est développé localement dans chacune des régions homogènes sur une base de fonctions. On considère simplement la dépendance spatiale pour une direction angulaire constante. En général, les méthodes de discrétisation spatiales utilisées avec la représentation angulaire S_N s'appuient sur deux équations pour la discrétisation de l'opérateur de transport. Il s'agit d'une équation de bilan qui sert au calcul du flux volumique et une équation de transmission pour la propagation du flux à travers les régions homogènes. Le flux dans la région homogène D_r et sur sa surface Γ_r sera discrétisé comme suit :

$$\begin{cases} \psi(\boldsymbol{r}) = \sum_{n=1}^{G_r} f_\alpha^n(\boldsymbol{r}) \psi_\alpha^n \quad , & \forall\, \boldsymbol{r} \in D_r, \\ \psi(\boldsymbol{r}) = \sum_{n=1}^{G_{s,r}} f_{s,r}^n(\boldsymbol{r}) \psi_{s,r}^n\,, & \forall\, \boldsymbol{r} \in \Gamma_r, \end{cases}$$

où r est l'indice de la région homogène D_r, tandis que G_r et $G_{s,r}$ sont respectivement les degrés de liberté associés à D_r et sa surface Γ_r. Les coefficients ψ_r^n et $\psi_{s,r}^n$ sont les

moments spatiaux du flux qui sont les inconnues à calculer. A cette classe de méthodes appartiennent les méthodes nodales, les Méthodes des Caractéristiques (MOC) structurées et non-structurées,… [15, 16]

1.4.4 Algorithme de résolution classique

Les méthodes numériques appliquées pour résoudre l'équation du transport sont basées sur des algorithmes itératifs visant à faire converger la valeur propre et la distribution de la source de fission sur le domaine étudié. Le bilan de neutrons est calculé indépendamment pour chaque groupe, partant du groupe le plus énergétique ($g = 1$) jusqu'au dernier groupe thermique ($g = N_g$). La matrice de scattering est divisée en trois parties :

- Le ralentissement des neutrons : $(H\psi)^{g'\to g}$ avec $g' < g$: les neutrons perdent de l'énergie par collision. Il s'agit de la partie triangulaire inférieure de la matrice. $(H\psi)_d^g$ est la source provenant de tous les groupes $g' < g$ (l'indice d signifiant ici $down\text{-}scattering$).
- Le self-scattering : $H^{g\to g}\psi^g$ qui décrit les collisions à l'intérieur du groupe. Le neutron change alors de direction sans changer de groupe d'énergie. C'est la diagonale de la matrice.
- L'up-scattering : $(H\psi)^{g'\to g}$ avec $g' > g$: les neutrons gagnent de l'énergie lors des collisions. Ces éléments sont dans la partie triangulaire supérieure de la matrice de scattering. $(H\psi)_u^g$ est la source provenant de tous les groupes $g' > g$.

Utilisant ces notations on reformule l'équation du transport pour mettre en évidence l'algorithme de résolution qui est formé par 3 niveaux d'itérations imbriqués :

$$\underbrace{L^g\psi = \overbrace{H^{g\to g}\psi}^{\text{Internes}} + \overbrace{(H\psi)_d^g + (H\psi)_u^g}^{\text{Thermiques}} + \overbrace{\frac{1}{\lambda}(F\psi)^g}^{\text{Externes}}}_{}\ , \tag{1.11}$$

Les itérations externes (indice e) permettent de faire converger la valeur propre et la distribution de la source de fission. Les itérations thermiques (indice th) visent à faire converger la source de scattering multigroupe des groupes thermiques, la matrice de scattering n'étant plus triangulaire inférieure. Enfin les itérations internes (indice i) permettent converger la source de self-scattering à l'intérieur d'un groupe et la distribution spatiale du flux angulaire. Ces différentes itérations sont expliquées dans les sections suivantes.

Le terme *Direct* sera utilisé dans la suite de ce manuscrit pour faire référence à l'algorithme (1.11).

1.4.4.a Itérations internes

Les itérations internes permettent de calculer la source de self-scattering et le flux angulaire. L'indice des itérations internes est noté i :

$$L^g \psi^{(i+1)} = H^{g\to g}\psi^{(i)} + Q^g_{ext}.$$

La source externe $Q^g_{ext} = (H\psi)^g_d + (H\psi)^g_u + \frac{1}{\lambda}(F\psi)^g$ est considérée comme source fixe. Le nouveau flux angulaire est calculé lors de ces itérations. Il permet de calculer les sources de scattering pour les autres groupes ainsi que le flux scalaire pour la source de fission.

1.4.4.b Balayage multigroupe et itérations thermiques

Les groupes sont divisés en deux catégories. Les groupes rapides pour lesquels la source de scattering ne provient que des groupes plus énergétiques ($(H\psi)^g_u = 0$). La matrice de transfert est triangulaire inférieure. Les groupes thermiques où l'up-scattering est possible (tous les groupes à partir de g : $(H\psi)^g_u \neq 0$) pour lesquels la matrice de transfert n'est pas triangulaire. Il en résulte deux équations selon que le groupe est rapide ou thermique :

$$\begin{cases} (L^g - H^{g\to g})\psi_g = (H\psi)^g_d + \dfrac{(F\psi)^g}{\lambda} & , g \leq N_{fast}, \\ (L^g - H^{g\to g})\psi_g = (H\psi)^g_d + (H\psi)^g_u + \dfrac{(F\psi)^g}{\lambda} & , g > N_{fast}. \end{cases}$$

N_{fast} est le nombre de groupes rapides et $(F\psi)^g$ est la source de fission dans le groupe g.

- Pour les groupes rapides, la solution ne dépend que des groupes $g' < g$. Un balayage allant dans l'ordre croissant des groupes permet d'inverser directement la matrice triangulaire inférieure :

$$(L^g - H^{g\to g})\psi_g^{(e+1)} = (H\psi)^{g\,(e+1)}_d + \frac{(F\psi)^{g\,(e)}}{\lambda^{(e)}}.$$

- Pour les groupes thermiques, la matrice n'est plus triangulaire inférieure. Les itérations thermiques permettent de faire converger la source d'up-scattering :

$$(L^g - H^{g\to g})\psi_g^{(th+1)} = (H\psi)^{g\,(th+1)}_d + (H\psi)^{g\,(th)}_u + Q^g_{fast}{}^{(e+1)} + \frac{(F\psi)^{g\,(e)}}{\lambda^{(e)}},$$

$$Q^g_{fast}{}^{(e+1)} = (H\psi)^{g\,(e+1)}_d, \qquad g' \leq N_{fast}$$

où $Q^g_{fast}{}^{(e)}$ est la source provenant des groupes rapides. L'algorithme utilisé est de type Gauss-Seidel, profitant de la résolution séquentielle des groupes. Finalement $\psi^{(e+1)}$ est le flux obtenu lors de la dernière itération thermique :

$$\psi_g^{(e+1)} = \psi_g^{(th=\infty)}, \qquad \forall g > N_{fast}$$

1.4.4.c Itérations externes

Les itérations externes ont pour but de converger la valeur propre du système ainsi que la distribution de la source de fission. La valeur propre est calculée de la manière suivante :

$$\lambda^{(e+1)} = \lambda^{(e)} \frac{\langle \boldsymbol{W}, \boldsymbol{F}^{(e+1)} \rangle}{\langle \boldsymbol{W}, \boldsymbol{F}^{(e)} \rangle},$$

où $\langle \cdot, \cdot \rangle$ est le produit scalaire tel que pour deux vecteurs $\boldsymbol{f}(\boldsymbol{r})$ et $\boldsymbol{g}(\boldsymbol{r})$ on a :

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \int d\boldsymbol{r}\, \boldsymbol{f}(\boldsymbol{r}) \cdot \boldsymbol{g}(\boldsymbol{r}),$$

et $\boldsymbol{W}(\boldsymbol{r})$ est une fonction de pondération. Les éléments du vecteur $\boldsymbol{F}(\boldsymbol{r})$ sont les intégrales de fission par isotope (indice is) définie par :

$$\boldsymbol{F}(\boldsymbol{r}) = \left\{ F_{is}(\boldsymbol{r}) = \sum_{g}^{N_g} \left(\nu\Sigma_f\right)_{is}^{g}(\boldsymbol{r}) \phi^{g}(\boldsymbol{r}) \right\}.$$

Dans notre cas on prend $\boldsymbol{W} = \boldsymbol{F}^{(e)}$.

La source de fission multigroupe $(F\psi)^g$ est mise à jour comme suit :

$$(F\psi)^{g\,(e+1)} = \sum_{is}^{N_f} \frac{\chi_{is}^{g}}{4\pi} F_{is}^{(e+1)}.$$

1.4.4.d Critères de convergence

Les itérations s'arrêtent lorsque les critères de convergence sont atteints simultanément. Le premier critère est la tolérance ε_λ sur la variation de la valeur propre entre deux itérations :

$$err_\lambda = \left| 1 - \frac{\lambda^{(e+1)}}{\lambda^{(e)}} \right| < \varepsilon_\lambda.$$

Le second critère est la tolérance ε_f sur la variation locale de la source de fission pour chacune des régions du domaine de calcul:

$$err_f(\boldsymbol{r}) = \max_{\forall \boldsymbol{r}, \forall is} \left| 1 - \frac{{F_{is}}^{(e+1)}(\boldsymbol{r})}{{F_{is}}^{(e)}(\boldsymbol{r})} \right| < \varepsilon_f.$$

1.5 Méthode des caractéristiques courtes du solveur IDT d'APOLLO3®

La Méthode des Caractéristiques Courtes (MOSC : acronyme de l'anglais Method of Short Characteristics) permet d'obtenir une résolution plus précise que d'autres méthodes telles que les méthodes puisqu'elle est basée sur la résolution exacte de la transmission au travers de régions homogènes. Dans cette section nous considérons la résolution du problème suivant :

$$(\boldsymbol{\Omega_d} \cdot \boldsymbol{\nabla} + \Sigma_t)\psi(\boldsymbol{r}, \boldsymbol{\Omega_d}) = q_d(\boldsymbol{r}, \boldsymbol{\Omega_d}),$$

où q_d est la source de scattering ($(H^{g\to g}\psi)(\boldsymbol{r}, \boldsymbol{\Omega_d})$) plus la source externe et $\boldsymbol{\Omega_d}$ la direction de propagation étudiée. Dans la suite de la section, nous omettons l'indice $\boldsymbol{\Omega_d}$ de la direction discrète.

L'idée générale de la MOSC est d'obtenir un couple d'équations donnant le flux dans une région homogène ainsi que le flux sortant à partir des sources et du flux entrant dans la maille sous la forme :

$$\psi = Cq + I\psi^-,$$
$$\psi^+ = Eq + T\psi^-,$$

que nous expliciterons ci-après. La transmission est calculée par l'équation intégrale du transport que nous introduirons par la suite. Le domaine complet est résolu par un balayage dans la direction $\boldsymbol{\Omega_d}$ en calculant successivement le flux des différentes régions grâce à la continuité du flux à l'interface entre les mailles : Le flux sortant d'une maille est le flux entrant dans la maille suivante.

1.5.1 Discrétisation spatiale

Le MOSC a été implémentée dans le solveur IDT pour des mailles cartésiennes homogènes [2] où la transmission dans une maille peut être calculée analytiquement. La méthode a été étendue à des Cellules Cartésiennes Hétérogènes (HCC) [3]. Il s'agit de mailles rectangulaires contenant un nombre arbitraire d'anneaux concentriques comme le montre la Figure 1.1. Le terme *cellule* est utilisé dans la suite pour désigner ce type de géométrie. Dans un cœur de réacteur nucléaire, une cellule combustible est généralement constituée du crayon combustible et de la gaine dans les anneaux, entouré par le modérateur. Lorsqu'il n'y a pas d'anneaux (géométrie plane à plaque par exemple), nous retrouvons le cas homogène. Nous présenterons donc ici la méthode pour le calcul des HCC, le cas homogène pouvant être vu comme un cas particulier.

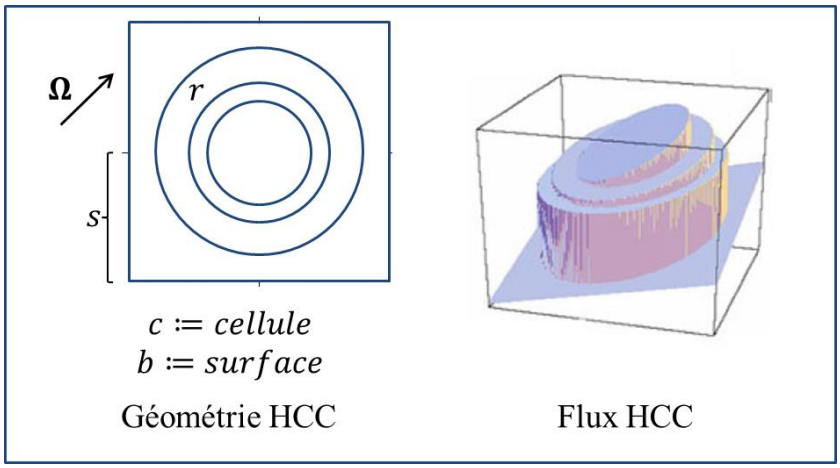


Figure 1.1. Discrétisation spatiale et représentation du flux d'une cellule cartésienne hétérogène (HCC) du solveur IDT.

Le MOSC repose sur l'équation intégrale du transport. Elle permet d'exprimer le flux dans la direction $\boldsymbol{\Omega}$ dans une cellule:

$$\psi(\boldsymbol{r},\boldsymbol{\Omega}) = e^{-\tau(0,l)}\psi^{-}(\boldsymbol{r}^{-},\boldsymbol{\Omega}) + \int_0^l dx e^{-\tau(x,l)} q(\boldsymbol{r}^{-} + x\boldsymbol{\Omega},\boldsymbol{\Omega}), \tag{1.12}$$

où $\psi^{-}(\boldsymbol{r}^{-},\boldsymbol{\Omega})$ est le flux entrant au point de la surface $\boldsymbol{r}^{-}$ et $\tau(x,l)$ est la distance optique entre $\boldsymbol{r}^{-} + x\boldsymbol{\Omega}$ et $\boldsymbol{r} = \boldsymbol{r}^{-} + l\boldsymbol{\Omega}$. L'épaisseur optique est exprimée par :

$$\tau(x,l) = \int_x^l dl' \Sigma(\boldsymbol{r}^{-} + l'\boldsymbol{\Omega})$$

Pour plus de simplicité, nous pouvons réécrire cette équation en introduisant les opérateurs T et K :

$$\psi(\boldsymbol{r},\boldsymbol{\Omega}) = (T\psi^{-} + Kq)(\boldsymbol{r},\boldsymbol{\Omega}). \tag{1.13}$$

Nous introduisons les développements des flux volumiques et surfaciques ainsi que de la source dans une cellule :

$$\begin{aligned}
\psi(\boldsymbol{r},\boldsymbol{\Omega}) &\sim \sum_{r\in c}\sum_{n=1,N_V} f_{r,n}(\boldsymbol{r})\psi_{r,n}(\boldsymbol{\Omega}) = \sum_{r\in c} \vec{f}_r(\boldsymbol{r})\vec{\psi}_r(\boldsymbol{\Omega}) \quad ,\forall\, \boldsymbol{r}\in D_c,\\
\psi^{\pm}(\boldsymbol{r},\boldsymbol{\Omega}) &\sim \sum_{s\in b}\sum_{n=1,N_s} f_{s,n}(\boldsymbol{r})\psi^{\pm}_{s,n}(\boldsymbol{\Omega}) = \sum_{s\in b} \vec{f}_s(\boldsymbol{r})\vec{\psi}^{\pm}_s(\boldsymbol{\Omega}) \quad ,\forall\, \boldsymbol{r}\in \Gamma^{\pm}_c,\\
q(\boldsymbol{r},\boldsymbol{\Omega}) &\sim \sum_{r\in c}\sum_{n=1,N_V} f_{r,n}(\boldsymbol{r})q_{r,n}(\boldsymbol{\Omega}) = \sum_{r\in c} \vec{f}_r(\boldsymbol{r})\vec{q}_r(\boldsymbol{\Omega}) \quad ,\forall\, \boldsymbol{r}\in D_c,
\end{aligned} \tag{1.14}$$

avec les notations de la Figure 1.1 : r représente l'indice de la région dans la cellule c ; s est une surface sur le bord de la cellule noté b; N_V et N_S sont respectivement les degrés de liberté dans les volumes et sur les surfaces de la cellule. Les $\psi_{r,n}(\boldsymbol{\Omega})$ sont les moments spatiaux du flux dans la région r et $\psi^{\pm}_{s,n}(\boldsymbol{\Omega})$ sont les moments spatiaux sur la surface s.

Pour utiliser l'approximation des équations (1.14) nous avons besoin d'introduire le produit scalaire définit par :

$$(f,g)_r = \int_{D_r} d\boldsymbol{r}\,(fg)(\boldsymbol{r}) \quad , \qquad A_s\langle f,g\rangle_s = \gamma_s(\boldsymbol{\Omega})\int_{\Gamma_s} d\Gamma(fg)(\boldsymbol{r}).$$

Dans la suite nous considérons que les fonctions sont orthonormées et respectent :

$$\left(f_{r,n},f_{r,m}\right)_r = V_r\delta_{nm}\ , \qquad \langle f_{s,n},f_{s,m}\rangle_s = \gamma_s(\boldsymbol{\Omega})\delta_{nm}\ ,$$

où V_r est le volume de la région r, A_s la surface du côté s, Γ_s le domaine du côté s de normale $\boldsymbol{n}_s$ et $\gamma_s(\boldsymbol{\Omega}) = |\boldsymbol{\Omega}.\boldsymbol{n}_s|$.

En projetant l'équation (1.13) sur $\vec{f}_r(\boldsymbol{r})$ on obtient pour chaque région de la cellule :

$$V_r\vec{\psi}_r(\boldsymbol{\Omega}) = \sum_{s'\in b} I_{r,s'}(\boldsymbol{\Omega})\vec{\psi}^-_{s'}(\boldsymbol{\Omega}) + \sum_{r'\in c} C_{r,r'}(\boldsymbol{\Omega})\vec{q}_{r'}(\boldsymbol{\Omega}).$$

$I_{r,s'}(\boldsymbol{\Omega}) = \left(\vec{f}_r, T\vec{f}_{s'}\right)_r$ est la matrice d'entrée depuis la surface s' vers la région r et $C_{r,r'}(\boldsymbol{\Omega}) = \left(\vec{f}_r, K\vec{f}_{r'}\right)_r$ est la matrice de collision de la région r' vers la région r. Cette équation permet d'obtenir le flux dans la région r de la cellule. De manière similaire, le flux sortant est obtenu par l'équation suivante :

$$\gamma_s(\boldsymbol{\Omega})\vec{\psi}^+_s(\boldsymbol{\Omega}) = \sum_{s'\in b} T_{s,s'}(\boldsymbol{\Omega})\vec{\psi}^-_{s'}(\boldsymbol{\Omega}) + \sum_{r\in c} E_{s,r}(\boldsymbol{\Omega})\vec{q}_r(\boldsymbol{\Omega}).$$

$T_{s,s'}(\boldsymbol{\Omega}) = \langle\vec{f}_s, T\vec{f}_{s'}\rangle_s$ est la matrice de transmission de la surface s' à la surface s et $E_{s,r}(\boldsymbol{\Omega}) = \langle\vec{f}_s, K\vec{f}_r\rangle_s$ est la matrice de fuite de la région r par la surface s.

Les fonctions $\vec{f}_r$ et $\vec{f}_s$ implémentées dans IDT sont des polynômes. Un développement jusqu'à l'ordre bilinéaire est utilisée pour les flux volumiques et jusqu'à l'ordre linéaire pour les flux surfaciques. Ce développement spatial permet une meilleure représentation du flux et permet ainsi de réduire le nombre de mailles à la discrétisation annulaire.

Finalement, les moments angulaires du flux nécessaires pour le calcul des sources (§1.3.2) sont calculés de la manière suivante :

$$\varphi_{r,n,l,m}(\boldsymbol{r}) = \frac{1}{4\pi}\sum_{d=1,\dots,N_d} A_{l,m}(\boldsymbol{\Omega}_d)\,\omega_d \sum_{n=1,N_V} f_{r,n}(\boldsymbol{r})\psi_{r,n}(\boldsymbol{\Omega}_d) \qquad (1.15)$$

où N_d est le nombre de directions angulaires et ω_d le poids associé à chaque direction. l est l'ordre d'anisotropie, $m = -l,\dots,l$. N_V est le nombre de moments spatiaux du flux.

La Figure 1.2 présente les discrétisations volumiques et surfaciques disponibles dans IDT. Les mailles volumiques sont constituées de tubes pour les anneaux et une seule maille pour la partie externe à l'anneau le plus grand. En ce qui concerne les surfaces,

un nombre de mailles dans chaque direction, $N_{subx}, N_{suby}, N_{subz}$ permet de discrétiser les surfaces.

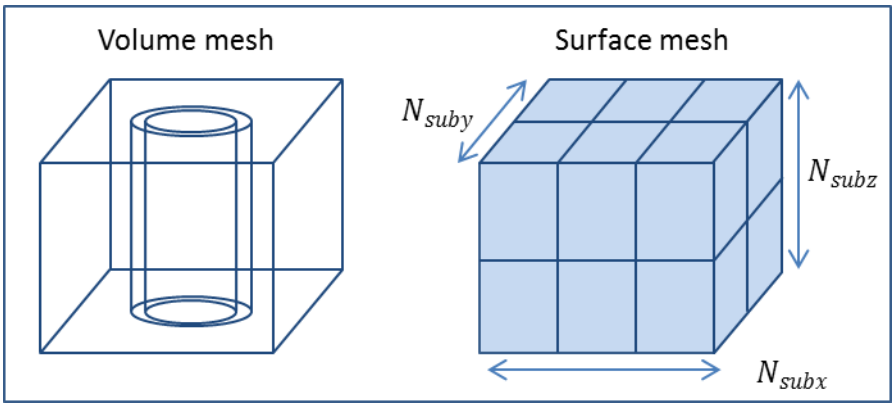


Figure 1.2. Discrétisation volumique et surfacique des cellules hétérogènes pour la méthode des caractéristiques courtes du solveur IDT.

1.5.2 Coût en mémoire

Le stockage des matrices I, C, T, E constitue une part importante de l'occupation mémoire d'un calcul IDT. Il peut être évalué de la manière suivante pour une cellule :

$$\begin{aligned} N_{coeff} &= N_G \times N_D \times (N_I + N_C + N_T + N_E), \\ &= N_G \times N_D \times \left(N_R \times nc + N_{S/2} \times nb\right)^2, \end{aligned}$$

qui est calculé de la manière suivante :

$$\begin{aligned} N_I &= N_R \times nc \times N_{S/2} \times nb \quad , \quad le\ nombre\ d'éléments\ de\ la\ matrice\ d'entrées, \\ N_C &= (N_R \times nc)^2 \quad , \quad le\ nombre\ d'éléments\ de\ la\ matrice\ de\ collisions, \\ N_T &= \left(N_{S/2} \times nb\right)^2 \quad , \quad le\ nombre\ d'éléments\ de\ la\ matrice\ de\ transfert, \\ N_E &= N_R \times nc \times N_{S/2} \times nb \quad , \quad le\ nombre\ d'éléments\ de\ la\ matrice\ de\ sorties, \end{aligned}$$

où :

$$\begin{aligned} N_G &\quad , \quad le\ nombre\ de\ groupes \\ N_D &\quad , \quad le\ nombre\ de\ directions \\ N_R &\quad , \quad le\ nombre\ de\ régions\ dans\ la\ cellule \\ N_{S/2} &\quad , \quad le\ nombre\ de\ surfaces\ entrantes \\ &\quad = N_{suby} \times N_{subz} + N_{subx} \times N_{subz} + N_{subx} \times N_{suby} \\ nc &\quad , \quad le\ nombre\ de\ moments\ spatiaux \\ nb &\quad , \quad le\ nombre\ de\ moments\ surfaciques \end{aligned}$$

Le coût en mémoire a aussi été optimisé par cette méthode. Pour cela, un typage des cellules a été réalisé. Les cellules ayant la même géométrie (nombre d'anneaux et leurs diamètres respectifs identiques) ainsi que les mêmes matériaux, dans chaque région, sont de même type. Les coefficients sont calculés une seule fois par type de cellule. Cette méthode permet d'épargner de la mémoire lorsqu'un motif est répété plusieurs fois dans le domaine de calcul.

1.5.3 Calcul des matrices

1.5.3.a Calcul des matrices – mailles homogènes

Lorsqu'une maille est homogène ($N_{subx} = N_{suby} = N_{subz} = N_R = 1$), les matrices sont calculées analytiquement. Pour calculer les fuites et la transmission, la surface de sortie est décomposée en 3 parties, chacune d'elles correspondant à la surface sortante d'un volume traversé par les trajectoires provenant des 3 surfaces entrantes (Figure 1.3).

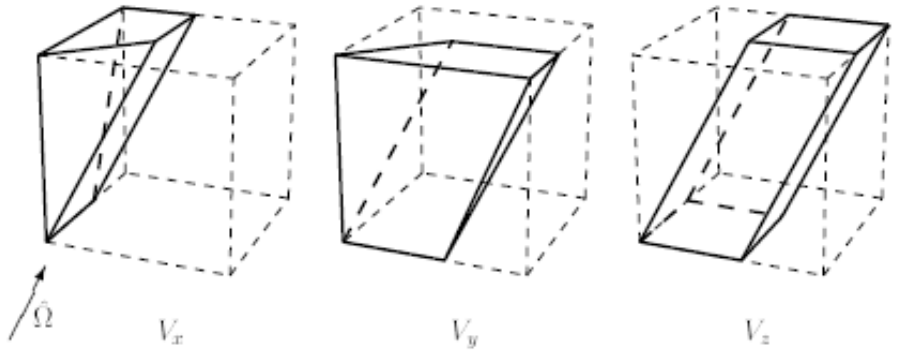


Figure 1.3. Décomposition de la surface sortante pour le calcul des coefficients de Fuites et de Transmission

Les longueurs des trajectoires l dans chaque volume V_u, peuvent s'exprimer en fonction d'une seule coordonnée spatiale $u = x, y, z$:

$$l(u) = au + b\,,$$

où a et b dépendent de la direction angulaire et des dimensions de la maille. Ainsi, les intégrales pour les matrices de fuite et de transmission sont calculées analytiquement.

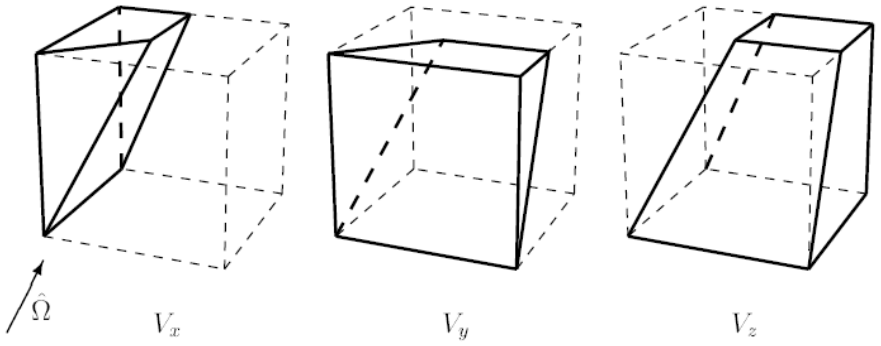


Figure 1.4. Décomposition du volume de la maille pour le calcul des coefficients de Collision et d'Entrée.

Il en est de même pour les coefficients d'entrée et de collision. Le volume de la maille est décomposé en 3 parties correspondant au volume traversé par les trajectoires provenant des 3 faces entrantes (Figure 1.4). La longueur des trajectoires s'exprime de

manière similaire aux coefficients de transmission et de fuite, permettant là aussi une intégration analytique.

Le calcul des matrices pour les mailles homogènes est peu coûteux. De ce fait, selon la mémoire disponible, il est possible de les calculer à la volée pour ne pas avoir à les stocker.

1.5.3.b Calcul des matrices – mailles hétérogènes

Pour les mailles hétérogènes ($N_{subx} \neq 1\ ou\ N_{suby} \neq 1\ ou\ N_{subz} \neq 1\ ou\ N_R \neq 1$), les intégrations sont réalisées numériquement. En projetant toutes les discontinuités géométriques (Figure 1.5) de la cellule sur la surface transverse, le domaine d'intégration est décomposé en plusieurs bandes. Ensuite une quadrature est définie de manière à sous-diviser chaque bande en M_G sous-bandes de même largeur définis par un pas de traçage. Enfin, une formule de Gauss-Legendre d'ordre N_G est appliquée à chaque sous-bande.

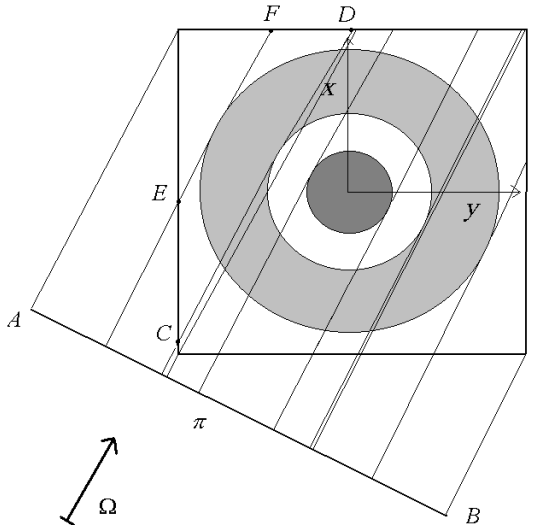


Figure 1.5. Projection des discontinuités de la cellule sur la surface perpendiculaire.

Finalement, l'intégrale le long de la trajectoire est évaluée analytiquement pour chaque bande.

Cette méthode d'intégration est sensiblement plus coûteuse que dans le cas des cellules homogènes. De ce fait, afin de réduire l'occupation mémoire, il est préférable de stocker les matrices dans des fichiers plutôt que de les recalculer à la volée.

Chapitre 2. Parallélisation de la résolution de l'équation du transport

2.1 Introduction

Dans ce chapitre, nous commencerons par faire un état des lieux des moyens disponibles pour paralléliser la résolution de l'équation du transport. Nous discuterons dans un premier temps des moyens de calcul existants pour la simulation et de leurs architectures. Nous présenterons les paradigmes de parallélisation à mémoire partagée et à mémoire distribuée. Les moyens de mesurer les performances parallèles d'une implémentation seront aussi introduits. Ensuite nous ferons une revue des méthodes de parallélisation de l'équation du transport présentées dans la littérature de manière à positionner notre nouvelle approche. Cette dernière sera présentée en détail dans ce chapitre.

Tout d'abord, le nouvel algorithme de résolution sera brièvement introduit de manière à le comparer aux algorithmes utilisés dans la littérature afin d'identifier les avantages et inconvénients qu'il engendre. Ensuite, nous introduirons le formalisme utilisé pour décrire précisément le nouvel algorithme de résolution. Finalement nous comparerons cet algorithme avec l'algorithme Direct à l'aide de résultats de comparaisons numériques obtenues sur un colorset d'assemblages de REL.

2.2 Un état des lieux

2.2.1 Les moyens de calculs disponibles pour la simulation et leurs architectures

Dans le domaine de la simulation numérique, l'outil expérimental est composé d'un algorithme et d'un calculateur, ce que nous appelons couramment le processeur, qui exécute l'algorithme.

Dans cette section, nous présenterons les caractéristiques des calculateurs couramment utilisés pour la simulation et comment leur architecture peut conditionner les algorithmes de résolution. Nous commencerons par un bref historique et les raisons qui amènent aux architectures actuelles, orientées vers le calcul parallèle. Nous présenterons ensuite deux types d'architectures de calculateurs basées sur l'utilisation de processeurs : architectures à mémoire partagée et à mémoire distribuée. Les concepts de parallélisme de base associés à ces architectures seront introduits. Il s'agit du parallélisme à mémoire partagée et du parallélisme à mémoire distribuée. Enfin, nous terminerons par la présentation de moyens d'évaluer les performances d'un calcul parallèle.

Les performances (temps de calcul) d'un outil de simulation sont directement liées à la puissance de calcul des processeurs [17, 18]. Il y a encore peu de temps,

l'amélioration des performances des outils de simulation reposait principalement sur l'augmentation de la fréquence des processeurs. En d'autres termes, l'amélioration des processeurs permettait d'accélérer les calculs, suivant plus ou moins la loi de Moore. Cette loi, empirique, fut formulée pour la première fois en 1965 dans « Electronics Magazine » par Gordon Moore. Elle suggère que le nombre de transistors présents dans les processeurs double tous les deux ans à un prix constant. Ainsi, il était possible de se contenter de l'augmentation de la puissance des processeurs pour améliorer la performance des outils de simulation.

Cependant, cette loi d'évolution est remise en cause pour les années à venir pour la simple raison que les limites physiques de la technologie actuelle ne permettent plus cette évolution exponentielle. En effet, l'augmentation de la densité des transistors et de la fréquence d'horloge engendre des problèmes de dissipation thermique. Les effets de ces limites sont déjà visibles : depuis quelques années, la fréquence d'horloge n'augmente plus. Une nouvelle voie est alors prise vers un autre type d'architecture dont le principe repose alors sur la multiplication des unités de calcul (cœurs) au sein d'un même processeur (multi-cœur). Malgré cela, le problème de dissipation de la chaleur n'est pas complètement résolu. De plus l'augmentation de la complexité des processeurs rend ce type d'architecture très coûteuse. Une autre approche est alors développée : elle consiste à utiliser plusieurs processeurs multi-cœurs, reliés les uns aux autres par un réseau de communication rapide. Les tâches à effectuer par les programmes sont réparties sur les différents cœurs et réalisées en parallèle. Cette approche s'est généralisée depuis plus de dix ans et les clusters de calculs basés sur ce concept sont de plus en plus employés. C'est par le parallélisme, tant au niveau du processeur que du calculateur entier, que la loi de Moore continue d'être suivie à l'heure actuelle.

D'autres voies d'augmentation des performances des unités de calcul sont explorées, telles que l'utilisation des cartes graphiques (GPU) par Nvidia [19] et AMD/ATI ou des coprocesseurs d'Intel (Many Integrated Cores) [20]. Ceux-ci permettent d'atteindre un haut degré de parallélisme par la disponibilité de nombreux cœurs. Des langages de programmation, comme OpenACC [21], ont pour ambition d'être performant sur tout type d'architecture. Cependant nous ne nous tournons pas vers ces architectures. Ce travail de thèse ayant pour objectif de développer une nouvelle méthodologie, nous préférons nous tourner vers des architectures plus classiques afin de mener plus aisément notre recherche.

Afin que les outils de simulation profitent de l'évolution des architectures, il est nécessaire que la programmation évolue pour tirer parti des nombreux cœurs disponibles. Les algorithmes actuels doivent être adaptés ou modifiés de façon à pouvoir profiter du calcul parallèle.

Dans la suite de cette section, nous abordons 3 points particulièrement importants vis-à-vis de la réalisation de ce travail de thèse. Tout d'abord, une analyse du fonctionnement du processeur et son lien avec la mémoire va nous permettre de comprendre les possibilités qu'offre le calcul parallèle mais aussi l'importance de

l'organisation des données dans un code. Ensuite nous nous intéresserons à deux types d'architectures de calculateurs parallèles : les architectures à mémoire partagée et les architectures à mémoire distribuée. Finalement, nous aborderons les méthodes de parallélismes associées à ces architectures et nous présenterons des moyens d'évaluer les performances d'un calcul parallèle.

2.2.1.a Fonctionnement et architecture des processeurs multi-cœurs

Dans ce paragraphe nous nous intéressons à l'architecture d'un processeur ainsi que son lien avec la mémoire. Ce paragraphe n'a pas pour objectif d'expliquer dans le détail l'architecture complète d'un processeur, mais d'identifier les principes de base qui permettent d'optimiser un code. La vitesse d'un calcul dépend principalement de deux paramètres : la vitesse à laquelle le processeur réalise les opérations, qui dépend de sa fréquence ainsi que le nombre d'opérations qu'il est capable d'exécuter par cycle, mais aussi la vitesse d'accès à la mémoire principale. Il est important de noter que la performance de l'unité de calcul est beaucoup plus importante que celle de la mémoire centrale (RAM) où sont stockées les données. Il faut compter plusieurs centaines de cycles du processeur pour accéder une donnée en mémoire. Et cette différence relative de performance ne fait qu'augmenter.

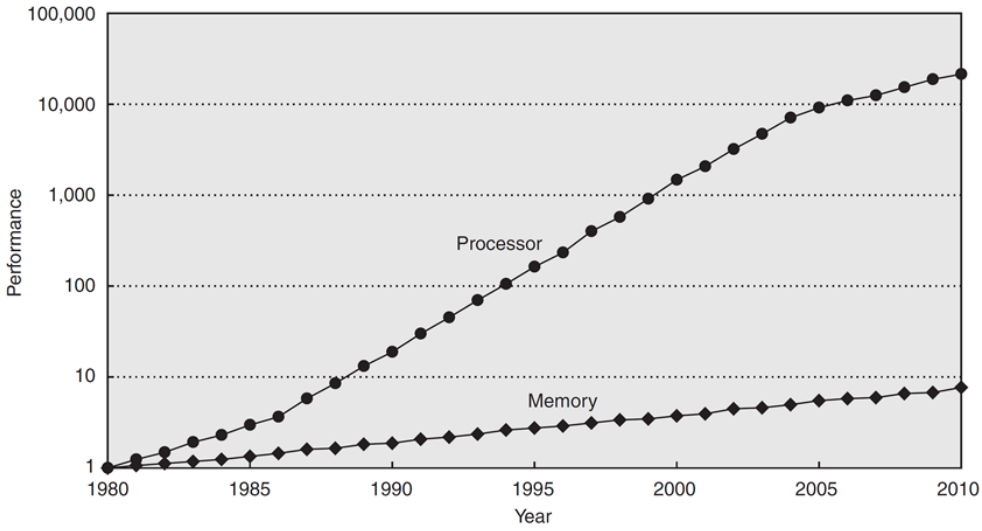


Figure 2.1. Evolution des performances des processeurs et de la mémoire [22].

De ce fait, le chargement des données dans le cœur entraîne une latence pendant laquelle celui-ci ne peut pas réaliser d'opération. Pour pallier à cette différence de vitesse, une mémoire « rapide », la mémoire *cache*, est insérée entre la mémoire RAM et le cœur. La latence passe alors à quelques cycles par accès au cache. Cette mémoire, dans laquelle le cœur va piocher les données dont il a besoin pour réaliser les opérations, a une taille limitée à quelques MegaByte (MB) ou dizaines de MB. La mémoire cache coûte cher, ce qui explique sa taille limitée. Dans les architectures des processeurs actuels la mémoire cache est hiérarchisée, souvent en 3 niveaux. Pour les architectures multi-cœurs, il est très fréquent que le dernier niveau de cache soit partagé par plusieurs cœurs.

Le cache est divisé en blocs, appelés *ligne de cache*, qui sont des copies temporaires des lignes de la RAM. Le transfert des données de la RAM vers le cache se fait par ligne entière. Lorsque le processeur doit accéder à une donnée, il interroge le cache pour savoir s'il contient cette donnée. Si elle est présente (*cache hit*) il peut alors la charger dans ses registres immédiatement, sinon (*cache miss*) une nouvelle ligne de cache contenant la donnée demandée est chargée depuis la RAM. Avant de charger une nouvelle ligne, il faut libérer de la place et donc renvoyer en mémoire centrale une des lignes. Le choix du remplacement est souvent de retirer la ligne utilisée le moins récemment. Plusieurs situations peuvent entraîner un cache miss. Tout d'abord s'il s'agit du premier accès à la donnée, celle-ci doit nécessairement être chargée. Ensuite dans les architectures multi-cœurs, la gestion de la cohérence des caches peut entraîner des caches miss pour deux raisons. La première est que le cache est partagé entre plusieurs cœurs et donc cela peut entrainer une migration de données du cache du fait d'une requête venant d'un autre cœur. La seconde est que la cohérence des caches locaux à chacun des cœurs doit être vérifiée, entrainant une communication entre les mémoires. Enfin, la taille limitée du cache fait qu'une donnée précédemment utilisée a pu être effacée pour libérer de la place. Pendant ces opérations de déchargement et de chargement de lignes de cache, le processeur est en attente sur plusieurs dizaines de cycles, entrainant un ralentissement dans l'exécution du programme. Afin de minimiser le nombre de cache miss, il faut se reposer sur les principes de localité temporelle et de localité spatiale. Améliorer la localité temporelle consiste à faire en sorte que lorsqu'un programme accède à une donnée, il y accède un maximum de fois dans un futur proche. De cette manière il est moins probable que la donnée ait été déchargée du cache pour libérer de la place. De même, la localité spatiale est améliorée en faisant en sorte que lorsqu'une donnée est utilisée, les données voisines le sont aussi. En procédant ainsi, l'ensemble de la ligne de cache chargée est utilisée. Par cet aspect, l'algorithme d'un programme (localité temporelle) et l'organisation du stockage de ses données (localité spatiale) ont une influence importante sur l'efficacité de l'utilisation d'un processeur.

2.2.1.b Architecture à mémoire partagée

Les architectures à mémoire partagée consistent à avoir un espace de mémoire centrale visible par tous les processeurs. Les processeurs ont leur propre mémoire cache (le dernier niveau pouvant être partagé entre plusieurs cœurs) dans laquelle est copiée la partie de la mémoire globale utilisée pour le calcul. Il y a deux types d'architecture à mémoire partagée. La première est appelée Uniform Memory Access (UMA). Elle contient un seul élément de mémoire centrale, accessible de la même manière par tous les processeurs. Pour la seconde, Non-Uniform Memory Access (NUMA), chaque processeur, ou groupe de processeurs, possède son propre élément de mémoire centrale, reliés les uns aux autres par un réseau. Tous les processeurs sont capables d'accéder à la totalité de la mémoire mais les temps d'accès ne sont pas les mêmes selon la localisation des données par rapport au processeur. Ces architectures possèdent un certain nombre d'avantages. Tout d'abord, la gestion de la cohérence des caches est le plus souvent transparente pour l'utilisateur. De plus l'adressage global,

visible par tous les processeurs rend la programmation en parallèle facile et permet de mutualiser des données entre les tâches réalisées en parallèle. Enfin, les communications entre les tâches sont transparentes et rapides. Cependant de telles machines sont coûteuses et les performances décroissent vite lorsque le nombre de processeurs devient important à cause du trafic sur les chemins d'accès aux données en mémoire et à la gestion de la cohérence des caches. Le nombre de cœurs est généralement limité à quelques dizaines.

2.2.1.c Architecture à mémoire distribuée

Les architectures à mémoire distribuée sont constituées de plusieurs processeurs ayant chacun leur propre mémoire centrale, isolée des autres. Les processeurs sont reliés entre eux à travers un réseau d'interconnexion. Les accès à la mémoire entre processeurs doivent alors se faire explicitement par échanges de messages à travers le réseau. Ce dernier détermine la vitesse d'accès aux données d'un processeur distant. La qualité du réseau liée à sa bande passante, à sa latence ainsi qu'à sa topologie, est très importante pour ce type d'architecture et conditionne les performances du calculateur. Les échanges de données entre processeurs doivent être gérés par le programmeur, entrainant un effort de développement supplémentaire. Le programmeur doit veiller à limiter les latences dues aux synchronisations entre les processeurs et le coût lié à l'envoie des messages. Malgré ces contraintes les architectures à mémoire distribuée ont l'avantage primordial de procurer un grand nombre de processeurs à un moindre coût par rapport à l'architecture en mémoire partagée. Le nombre de processeurs n'est alors à priori pas limité. De même la quantité de mémoire croit proportionnellement au nombre de processeurs et les accès à la mémoire locale restent rapides indépendamment du nombre de processeurs.

Aujourd'hui les calculateurs combinent les architectures à mémoire distribuée et à mémoire partagée. Ils sont constitués d'éléments à mémoire partagée, appelés nœuds, reliées entre eux par un réseau d'interconnexion. Ces architectures mixtes permettent de tirer profit des avantages de la mémoire partagée et des avantages de la mémoire distribuée.

2.2.1.d Programmation en Calcul Parallèle

NOTION DE GRANULARITE

La parallélisation d'un programme peut être réalisée à différents niveaux. Ces niveaux correspondent à ce que l'on appelle la granularité du parallélisme. Le parallélisme à gros grain, ou *coarse grained*, consiste à exécuter des tâches comportant un grand nombre d'instructions en parallèle avec un minimum de communication entre celles-ci. La limitation des communications entre les tâches permet de limiter le surcoût lié à la parallélisation. A l'opposé le parallélisme à grain fin, ou *fine grained*, consiste à paralléliser des boucles ou des opérations les plus élémentaires sur les variables. Le parallélisme fine grained permet de paralléliser l'algorithme à l'aide d'un grand nombre de tâches regroupant peu d'instructions, l'inconvénient étant alors l'augmentation des surcoûts liés à la gestion des tâches est

des communications entre celles-ci. De manière générale, il est préférable que le grain de parallélisme soit de même niveau que le grain de la machine utilisée. Etant donné que désormais les architectures proposent différents niveau de parallélisme (cluster de machines à mémoire partagée composant de nœuds de calculs multiprocesseurs multi cœurs), les algorithmes doivent utilisés les différents grains de parallélisme afin d'exploiter au mieux les capacités des architectures.

PARALLELISME A MEMOIRE PARTAGEE

Le principe du *multithreading* est utilisé pour le parallélisme à mémoire partagée. Cela consiste à exécuter un programme qui peut activer plusieurs processus légers, appelés *threads*. Ces derniers sont distribués sur les différents cœurs de la machine à mémoire partagée par le système d'exploitation pour pouvoir s'exécuter en parallèle. L'exécution des threads est réalisée dans l'espace mémoire du processus initial qui est alors partagée entre tous les threads. A celle-ci s'ajoute une mémoire locale à chaque thread. Cette mémoire est privée et n'est pas visible par les autres threads.

Pour implémenter ce modèle de programmation, l'interface de programmation Open Multi Processing (OpenMP) [23] est couramment utilisée et s'est imposé comme standard de base. Elle est composée de directives (permettant de définir le travail à partager, les synchronisations ou encore le statut privé ou partagé des données), de sous-programmes (faisant partie d'une bibliothèque) et de variables d'environnement pour gérer l'exécution du programme (nombre de threads,…). L'utilisation des directives permet de créer et/ou détruire des régions parallèles pendant l'exécution du programme. A l'entrée d'une région parallèle (`!$OMP PARALLEL`) la tâche maîtresse génère des threads qui disparaissent à la fin de la région parallèle (`!$OMP END PARALLEL`) alors que la tâche maîtresse poursuit l'exécution du programme.

Chaque thread a accès à la mémoire globale partagée en plus d'une mémoire privée qui lui est propre. Les échanges de données entre les threads se font via la mémoire partagée. La mémoire cache du processeur étant une copie temporaire de la mémoire centrale, la gestion de la cohérence du cache doit être faite avec une attention particulière. Il faut en effet s'assurer que lorsqu'un thread lit une donnée depuis le cache de son processeur, celle-ci a bien la même valeur que sa copie stockée dans la mémoire centrale. La cohérence du cache peut être gérée par des directives explicites ou implicites. La vérification de cohérence des données d'un thread est la partie la plus coûteuse du parallélisme à mémoire partagée. La gestion de la mémoire d'un code ainsi que l'implémentation des algorithmes parallélisés doivent être adaptée de manière à limiter les besoins de synchronisation de la mémoire entre les threads.

La parallélisation fine grained d'un programme en OpenMP est très aisée et ne nécessite que peu de modifications du code séquentiel. Des directives dédiées à la parallélisation des boucles (`!$OMP DO`) ou de tâches (`!$OMP SECTIONS`) sont fournies dans la bibliothèque. En revanche un parallélisme coarse grained est nettement plus complexe à développer. La nature partagée ou privée des variables, les synchronisations et les communications entre les threads doivent être gérées avec

attention. Cependant OpenMP a été développée de manière à supporter aussi bien le coarse grained que le fine grained. Elle est la bibliothèque la plus répandue pour paralléliser en mémoire partagée. D'autres bibliothèques comme Intel TBB [24] ou pthread [25] sont disponibles pour paralléliser en mémoire partagée.

PARALLELISME A MEMOIRE DISTRIBUEE

Pour le parallélisme à mémoire distribuée, le modèle utilisé est celui de l'échange de messages. Le programme est exécuté en lançant plusieurs processus indépendants qui pourront communiquer entre eux via un mécanisme d'échange de message. Chaque processus exécute un programme et gère sa propre mémoire sans connaissance de celle des autres processus. De ce fait, toutes les variables du programme sont des variables locales pour chaque processus. Les processus peuvent alors s'échanger des messages afin de transférer des données ou bien de se synchroniser. Ces communications sont décrites explicitement par le programmeur. L'interface de programmation couramment utilisée est Message Passing Interface (MPI) [26]. Elle fournit une bibliothèque de fonctions permettant de gérer l'environnement de l'exécution du programme parallèle, les communications entre les processus,… Les communications peuvent avoir lieu entre deux processus (communication point-à-point : `MPI_SEND` ou `MPI_RECV` par exemple) ou alors entre plusieurs processus (communications collectives : `MPI_BCAST`, `MPI_ALLREDUCE`, `MPI_ALLGATHER`,…). Pour chaque type de communication, le message est envoyé au(x) destinataire(s). Celui-ci doit alors pouvoir interpréter le message qui lui a été adressé. Ainsi avec l'envoi des données, un processus doit aussi communiquer son identifiant, celui du destinataire et surtout le type de données envoyé et la longueur du message. Ces informations complémentaires constituent en quelque sorte l'enveloppe du message et ont un coût. Une parallélisation efficace devant minimiser le coût des communications par rapport à celui du calcul, l'envoi d'un très grand nombre de petits messages n'est pas adapté à ce type de parallélisation. C'est pourquoi MPI est couramment utilisé dans le contexte d'un parallélisme coarse grained et en particulier dans les méthodes de décomposition de domaine (DD). La parallélisation en MPI a l'avantage d'être très portable et d'offrir un cadre facilitant la parallélisation en mémoire distribuée ; elle s'est naturellement imposée comme la norme. Le nombre de processeurs et la mémoire totale disponible ne sont à priori pas limités. Le temps d'exécution d'un programme peut alors être grandement réduit en parallélisant sur un grand nombre de processeurs. De plus certaines simulations demandent une grande quantité de mémoire qui n'est alors disponible que sur des architectures à mémoire distribuée. Cependant cette parallélisation peut faire face à de sérieux problèmes. Pour des programmes dont le volume de données dépend directement du nombre de processus, la quantité de mémoire requise devient vite très importante. De plus les performances des communications collectives se dégradent avec l'augmentation du nombre de processeurs. Malgré tout, MPI reste l'outil le plus utilisé sur les architectures à mémoire distribuée ou les architectures mixtes.

Finalement un modèle de programmation hybride mémoire partagée / mémoire distribuée permet de profiter des avantages des deux approches de la parallélisation. L'utilisation simultanée des bibliothèques MPI et OpenMP permet par exemple de réduire le nombre de processus MPI et donc de diminuer le nombre de réplication de certaines données en mémoire, le nombre de communications collectives, le nombre de messages,… et ainsi de lever des limites de scalabilité. MPI permet d'avoir accès au nombre de processeurs et à la quantité de mémoire souhaité alors que OpenMP permet de limiter les défauts du parallélisme à mémoire distribuée. Cette complémentarité est donc un réel avantage.

2.2.1.e Mesure des performances du calcul parallèle

Le but de la parallélisation est d'accélérer l'exécution d'un algorithme en multipliant les processeurs ou cœurs. Dans une situation idéale, l'utilisation de N cœurs permettrait de résoudre le problème N fois plus vite. Malheureusement, ce n'est pas toujours le cas pour plusieurs raisons. L'algorithme parallélisable peut être moins performant que l'algorithme séquentiel ou bien l'implémentation parallèle dégrade les performances. Pour caractériser le gain lié à la parallélisation, on définit le facteur d'accélération, ou *speedup*, et l'efficacité. Le *speedup* $S(N)$ est défini comme le rapport entre le temps d'exécution du programme en séquentiel et le temps d'exécution du même programme parallélisé :

$$S(N) = \frac{t(1)}{t(N)},$$

où $t(N)$ est le temps d'exécution avec N unités de calcul. Idéalement, $S(N) = N$, c'est-à-dire que l'accélération augmente comme le nombre d'unités de calcul. L'efficacité $E(N)$ est définie comme le speedup divisé par le nombre d'unités de calcul :

$$E(N) = \frac{S(N)}{N}.$$

Toutefois, même si ces deux grandeurs renseignent sur la qualité de la parallélisation, elles ne permettent pas de mesurer la qualité de l'algorithme parallèle utilisé. C'est pourquoi il est aussi important de comparer le temps d'exécution du programme parallélisé avec le temps du « meilleur algorithme séquentiel » pour définir une accélération *utile*. En effet, c'est elle qui indiquera à l'utilisateur le gain réel procuré par la parallélisation du programme.

D'autre part, le speedup que l'on peut obtenir sur un algorithme peut être fortement affecté par les parties de code qui ne sont pas parallélisées dans l'algorithme. Les parties exécutées en séquentiel représentent un goulot d'étranglement qui freine l'accélération par la parallélisation. Ceci est exprimé par la loi d'Amdahl :

$$S_{Amdahl} = \frac{1}{(1-s) + \frac{s}{N}},$$

où s est la fraction du temps concernée par la parallélisation. En supposant un nombre infini de processeurs, la loi d'Amdahl donne la limite maximale de l'accélération :

$$S_{max} = \frac{1}{(1-s)}.$$

Cette loi donne une limite théorique du meilleur speedup atteignable avec un algorithme. Cependant elle ne prend pas en compte un certain nombre d'éléments pouvant dégrader les performances.

En effet, plusieurs causes peuvent être à l'origine de pertes de performances. Le premier élément à prendre en compte est la qualité de l'algorithme parallélisable. En effet, le facteur d'accélération *utile* consiste à comparer l'algorithme séquentiel à l'algorithme parallélisable. Si, exécuté sur un processeur, ce dernier est moins bon que l'algorithme séquentiel, une scalabilité idéale ne pourra pas être atteinte. D'autre part, comme le montre la loi d'Amdahl, la fraction de code séquentielle donne une limite théorique à l'accélération de l'algorithme. Il est donc important de veiller à ce que la partie séquentielle du code soit réduite au maximum. Par ailleurs, le *déséquilibrage de charge* peut faire chuter le speedup. Si les tâches sont mal réparties, les processeurs ayant le moins de travail devront attendre que les autres finissent leurs tâches. Les performances seront alors dégradées. De plus, le surcoût des opérations liées au parallélisme peut faire chuter l'efficacité de la parallélisation d'un algorithme. Enfin, la perturbation du programme par l'OS ou d'autres processus peuvent perturber une tâche parallèle et dégrader les performances de l'ensemble du calcul.

Nous terminerons par l'introduction de la notion de *scalabilité*. Elle désigne la capacité d'un système à s'adapter à un changement d'ordre de grandeur. En parallélisme, elle consiste à étudier le comportement d'un algorithme en fonction du nombre de processeurs. Lorsque le nombre de processeurs augmente, des pertes de performances peuvent apparaître. Par exemple, l'algorithme peut se dégrader ou bien les surcoûts dus à l'implémentation peuvent se manifester. Pour évaluer la scalabilité d'un programme, deux tests sont souvent réalisés. Le *weak scaling* consiste à accroitre la taille du problème global proportionnellement au nombre de processeurs. Ce test permet d'estimer jusqu'à quelle taille de problème l'algorithme restera performant. Le *strong scaling* consiste à résoudre un problème unique avec un nombre croissant de processeurs. Il permet de voir jusqu'à quel point la résolution d'un problème peut être accélérée avec l'algorithme parallélisé.

2.2.2 Revue des méthodes de parallélisation de l'équation du transport

Nous venons de voir qu'il n'est plus possible de se reposer sur l'augmentation continue de la vitesse des cœurs des processeurs et de la quantité de mémoire pour améliorer les performances de calcul ainsi que d'étendre le champ des problèmes solvables.

En ce qui concerne les méthodes déterministes pour la résolution de l'équation du transport, celles-ci doivent s'appuyer sur les possibilités offertes par le parallélisme. Il faut alors distribuer des tâches sur chaque cœur du calculateur parallèle. Ce

changement dans l'architecture des calculateurs impose une évolution des algorithmes de façon à pouvoir profiter du calcul parallèle. De manière générale, la parallélisation consiste à distribuer des tâches sur chaque processeur et à les effectuer en parallèle. La méthode classique de parallélisation des méthodes déterministes consiste à partitionner la discrétisation énergétique, angulaire et spatiale en blocs de taille réduite. Les différents processeurs seront alors en charge du calcul de ces blocs, permettant de paralléliser le calcul.

DENOVO [27] est un code 3D basé sur la méthode des ordonnées discrètes avec plusieurs discrétisations spatiales disponibles. Ce code est développé pour permettre une décomposition spatiale et angulaire dans l'objectif de paralléliser les calculs sur un grand nombre de processeurs. Pour réaliser le balayage spatial, DENOVO emploie la décomposition de domaine avec l'algorithme de Koch-Baker-Alcouffe (KBA) parallèle par front [28]. Cette méthode qui permet d'inverser directement l'opérateur de transport a montré ses limites de scalabilité pour un grand nombre de processeurs. Dans les travaux présentés par Evans [27], la parallélisation de la variable énergétique n'est pas présentée. En effet, l'algorithme de Gauss-Seidel utilisé traditionnellement pour calculer les sources de scattering, nativement séquentiel, ne permet pas une parallélisation efficace de la variable énergétique. Néanmoins dans une publication récente [29], les auteurs proposent un solveur avancé basé sur une méthode de Krylov pour paralléliser la variable énergétique. Cependant la principale limitation de ce solveur vient du fait qu'il repose sur un maillage cartésien structuré, rendant très chère une discrétisation fine des géométries circulaires. Ceci est un problème non négligeable pour l'analyse des cœurs de réacteurs à eau légère ou les réacteurs à neutrons rapides composés principalement de géométries circulaires.

UNIC [30] utilise la méthode des éléments finis tétraédriques ou hexaédriques pour mailler la géométrie. Plusieurs méthodes de résolution spatiale-angulaire sont disponibles. PN2ND et SN2ND sont basés sur la formulation pair-impaire de l'équation du transport. Dans PN2ND la variable angulaire est projetée sur une base d'harmoniques sphériques alors que SN2ND utilise la méthode collocative des ordonnées discrètes. Ces deux solveurs sont parallélisés sur l'espace par décomposition de domaine (DD) et sur la variable angulaire. Seul le parallélisme à mémoire distribuée a été utilisé avec la bibliothèque MPI. Les performances parallèles de PN2ND n'ont pas été analysées méthodiquement, contrairement à celles de SN2ND. Pour ce dernier la scalabilité est très bonne (>80%) jusqu'à plus de 150,000 cœurs. Un 3ème solveur, MOCFE est basé sur la Méthode des Caractéristiques (MOC) pour des géométries 2D et 3D. Les variables d'angle et d'espace (décomposition de domaine) sont parallélisées en utilisant un algorithme GMRES de PETSc [31] au lieu de l'algorithme de Jacobi par bloc classique. Les performances parallèles semblent moins bonnes que pour SN2ND à cause du déséquilibrage de charge dû au fait que le nombre d'intersections avec les surfaces des sous domaines ne croît pas proportionnellement avec le nombre de segments des lignes caractéristiques [32]. Les auteurs soulignent la nécessité de plus amples recherches afin d'obtenir une meilleure

scalabilité pour des Calculs à Haute Performance (HPC, acronyme de l'anglais High Performance Computing).

Très récemment dans MPACT [33], Kochunas a proposé une parallélisation de la Méthodes des Caractéristiques (MOC) en 3D par décomposition des variables d'espace et d'angle pour des calculs HPC. En plus de cette décomposition, la méthode des caractéristiques permet de paralléliser le calcul des trajectoires. Cette parallélisation a été réalisée en utilisant le parallélisme à mémoire distribuée pour les variables d'espace et d'angle, alors que le calcul des trajectoires est parallélisé en mémoire partagée. Cette décomposition est appliquée uniquement à la résolution de l'équation du transport dans un seul groupe d'énergie, la variable énergétique n'ayant pas été parallélisée. Dans cette méthode, les communications entre les processeurs sont présentes à plusieurs niveaux. Tout d'abord une opération de réduction doit être réalisée pour calculer le flux scalaire : somme sur les trajectoires et les directions. L'approximation des sources isotropes est utilisée. Cette réduction intervient dans la parallélisation des trajectoires en mémoire partagée et dans la parallélisation sur les directions en mémoire distribuée. Un algorithme de Jacobi Parallèle par Bloc (PBJ, acronyme de l'anglais Parallel Block Jacobi) est utilisé pour maximiser la parallélisation. Ainsi dans un sous domaine, une synchronisation est nécessaire entre toutes les directions et tous les calculs de trajectoire pour effectuer cette opération. Ensuite, des communications ont lieu entre les sous domaines de la décomposition spatiale. Afin de préserver la qualité de la solution, le flux aux surfaces des sous domaines doit être conservé pour chaque point d'impact des trajectoires pour le processus gérant un angle dans un sous domaine. De cette manière, il peut être transmis au sous domaine voisin sans insertion d'approximation. Dans ce travail, Kochunas montre que l'efficacité de la méthode est très bonne (>90%) en termes de performances parallèles. Le problème de déséquilibrage de charge identifié dans MOCFE n'est pas présent ici. Le traçage modulaire utilisé impose une grille structurée. Dans le cadre des REL, cette grille est cartésienne et le réseau est régulier. De ce fait le nombre de segments et d'intersections est similaire dans chaque sous domaine. Cette méthode à l'avantage de permettre un équilibrage de charge maitrisé mais restreint l'application au cas comprenant une grille structurée. Cependant l'auteur note qu'un calcul sans DD requiert nettement moins d'itérations. Dans son travail de thèse [34] Kochunas présente cette méthode en accélérant les itérations externes par le CMFD (cf. §3.3). En appliquant cette méthode au Benchmark de Takeda [35], l'auteur montre que pour plusieurs décompositions spatiales, le nombre d'itérations externes accélérées par le CMFD reste constant et identique au calcul sans DD.

Finalement, OpenMOC [36] propose une toute autre approche en se focalisant sur une parallélisation à mémoire partagée. Cette parallélisation est réalisée sur les angles azimutaux et les trajectoires. L'algorithme classique a été revu pour optimiser la parallélisation. Le balayage multigroupe est réalisé dans la boucle sur les segments des caractéristiques, partie la plus interne du code. Cette méthode de balayage énergétique ne permet plus de bénéficier de l'algorithme de Gauss-Seidel classique

utilisé pour faire converger la source de scattering multigroupe. Les tests de scalabilité montrent une scalabilité idéale jusqu'à 12 cœurs lorsque les barrières de synchronisation des variables sont évitées en dupliquant le flux scalaire dans chaque thread et en effectuant l'opération de réduction hors de la zone parallèle. Cependant aucune comparaison n'est réalisée avec l'algorithme classique des itérations de puissance. Ces tests de scalabilité sont donc purement une évaluation de la qualité de la parallélisation. Au-delà des performances du solveur, l'auteur relève un certain nombre d'éléments pertinents à propos de la parallélisation. En particulier la nécessité de hiérarchiser les niveaux de parallélisme et de s'adapter à l'architecture des machines de calcul. De nos jours, même les architectures à mémoire partagée possèdent plusieurs cœurs par nœud (~12-64). La plupart des codes utilisent le parallélisme à mémoire distribuée, entrainant la copie redondante des données sur plusieurs processus, alors que la mémoire par processeur a plutôt tendance à décroitre. L'optimisation de la mémoire ainsi que celle du cache sont pointées comme argument de l'utilisation du parallélisme à mémoire partagée. OpenMOC est orienté vers l'optimisation du niveau fine grained de la parallélisation, laissant ouvertes d'autres voies pour une parallélisation coarsed grained de plus haut niveau.

Pour conclure sur les méthodes de parallélisation de l'équation de transport, nous relèverons les points suivants. A l'exception d'OpenMOC, l'ensemble des méthodes de parallélisation reposent sur des décompositions de domaines spatiales et angulaires. Ces décompositions de domaines sont réalisées de manière à modifier le moins possible les propriétés de convergence de l'algorithme initial. L'algorithme KBA de DENOVO permet de paralléliser le balayage spatio-angulaire en reproduisant l'algorithme classique. Kochunas [33] utilise l'algorithme PBJ qui se détache de l'algorithme séquentiel en rompant le balayage spatio-angulaire. Pour améliorer la convergence de l'algorithme, les itérations sont accélérées par le CMFD. De manière générale les performances parallèles sont bonnes pour la parallélisation des variables spatiales et angulaires. Nous pouvons cependant noter que l'ensemble de ces méthodes de parallélisation sont réalisées par décomposition des variables de l'équation à un groupe d'énergie (cf. équation (1.10)). A notre connaissance, il n'y a pas de nom dans la littérature pour spécifier que la DD n'est appliquée que pour le balayage spatio-angulaire. Afin d'éviter toute confusion, un nom spécifique est introduit ici pour désigner les algorithmes faisant référence aux méthodes de DD appliquées au niveau du balayage spatio-angulaire : algorithme Parallèle par Bloc dans un Groupe (PB1G). Seul OpenMOC remet en question l'algorithme classique en incluant le balayage multigroupe dans le balayage spatial.

2.2.3 Introduction aux Méthodes de Décomposition de Domaine

La première formulation de la décomposition de domaine est attribuée à H.A. Schwarz [37] qui propose une méthode itérative pour la résolution de problèmes elliptiques sur un domaine géométriquement complexe (Figure 2.2). Les Méthodes de Décomposition de Domaine (DDM) consistent à diviser un problème de grande taille en une suite de sous-problèmes de taille plus petite.

Les principaux avantages des DDM sont qu'elles amènent naturellement au parallélisme et qu'elles permettent de diviser un problème de grande taille en plusieurs sous problèmes de petite taille, plus faciles à résoudre et plus adaptés aux capacités des machines de calcul. C'est pour ces raisons que nous nous tournons vers les DDM.

2.2.3.a Méthodes avec recouvrement

On considère un domaine D et l'équation aux dérivées partielles suivantes :

$$\begin{cases} Lu = f, & dans\ X, \\ u = g, & sur\ \Gamma\ . \end{cases} \tag{2.1}$$

L étant un opérateur différentiel et Γ la frontière du domaine. L'idée de la décomposition de domaine repose sur une décomposition du domaine X en plusieurs sous domaines tel que :

$$X = X_1 \cup X_2, \qquad X_1 \cap X_2 \neq \emptyset,$$

Il s'agit de résoudre itérativement sur chacun d'eux un problème identique au problème initial en utilisant une condition aux limites de Dirichlet sur la frontière $\Gamma_i \cap X_j$ pour prendre en compte la présence des sous domaines voisins. Une représentation schématique de la DDM est présentée Figure 2.2 :

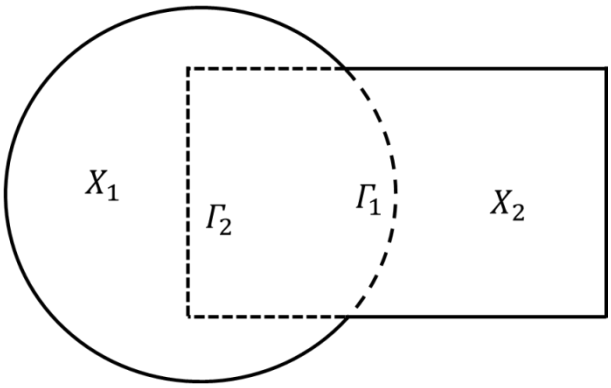


Figure 2.2. Décomposition de domaine avec recouvrement.

Le schéma de résolution est alors le suivant : en partant des approximations $u_1^{(0)}$ et $u_2^{(0)}$ pour la solution initiale dans les sous domaines, on a pour $n = 1,2,3, \ldots$

Algorithme 2.1. Algorithme de Schwarz.

$Jusqu'$à $convergence, répéter$:

$$résoudre : \begin{cases} Lu_1^{(n+1)} = f_1\ , & dans\ X_1, \\ u_1^{(n+1)} = g\ , & sur\ \Gamma_1 \cap \Gamma, \\ u_1^{(n+1)} = u_2^{(n)}, & sur\ \Gamma_1 \cap X_2. \end{cases}$$

$$résoudre : \begin{cases} Lu_2^{(n+1)} = f_2\ , & dans\ X_2, \\ u_2^{(n+1)} = g\ , & sur\ \Gamma_2 \cap \Gamma, \\ u_2^{(n+1)} = u_1^{(n')}, & sur\ \Gamma_2 \cap X_1. \end{cases}$$

A chaque itération n on résout les deux sous problèmes sur chaque sous domaine ce qui permet d'obtenir une suite de solutions approchées $(u_1^{(n)}, u_2^{(n)})$. Ce processus itératif, dont Schwarz a montré la convergence [37], permet d'obtenir la solution du problème initial.

En fonction du type de mise à jour $u_2^{(n+1)} = u_1^{(n')}$, la méthode se décline en deux versions. L'algorithme dit additif lorsque l'on prend $n' = n$ et l'algorithme dit multiplicatif lorsque $n' = n + 1$. On peut faire une analogie directe de ces deux méthodes avec les algorithmes de Jacobi par bloc et de Gauss-Seidel pour la résolution d'un système linéaire. Si l'avantage de l'algorithme multiplicatif est de converger plus rapidement, celui de l'algorithme additif est qu'il est directement parallélisable puisque la résolution de chaque sous domaine est indépendante.

Les avantages d'une telle décomposition de domaine sont nombreux. La géométrie initiale est composée de formes plus simples (Figure 2.2) et la taille de chaque problème a une dimension réduite par rapport au problème initial. D'autre part les propriétés de convergence des méthodes de décomposition de domaine avec recouvrement sont bonnes et la parallélisation de l'algorithme de Schwarz, dans sa version additive, est très aisée.

Cependant, le principal défaut de cette approche est le coût en mémoire. Sur les zones de recouvrement, l'ensemble des données doivent être dupliquées : le flux, les sources, les sections efficaces, les coefficients du transport,... Augmentant ainsi les besoins en mémoire.

De plus une méthode sans recouvrement à l'avantage de faciliter la réalisation de la DDM. En effet, le domaine est généralement composé d'éléments de base qui ne sont pas arbitraires. Il est donc plus facile d'un point de vue de la géométrie de manipuler et mettre en relation des objets géométriques par leurs surfaces externes. De même, une méthode sans recouvrement ne nécessite pas de calculer de données surfaciques à l'intérieur d'un sous domaine pouvant être difficiles à récupérer.

2.2.3.b Méthode sans recouvrement

Pour pallier aux inconvénients énoncés dans la section précédente, nous nous tournons vers une méthode de décomposition de domaine sans recouvrement (Figure 2.3). Le seul lien entre les sous domaines se réduit à la surface qu'ils partagent :

$$X = X_1 \cup X_2, \qquad X_1 \cap X_2 = \emptyset, \qquad \Gamma_1 \cap \Gamma_2 \neq \emptyset.$$

Avec ces méthodes basées sur le complément de Schur, on cherche à déterminer la solution au niveau de l'interface.

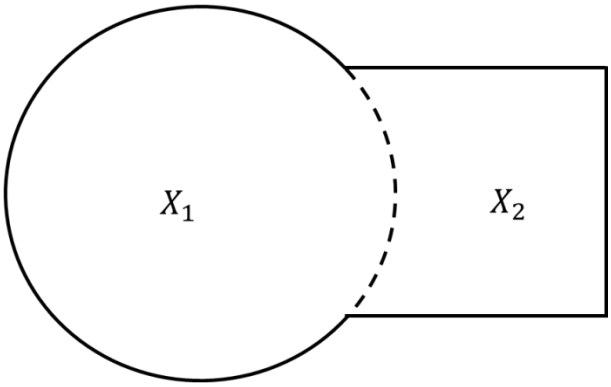


Figure 2.3. Décomposition de domaine sans recouvrement.

Dans chaque sous domaine, on résout le problème suivant $i = \{1,2\}$:

$$\begin{cases} Lu_i = f_i \ , & dans\ X_i, \\ u_i = g \ , & sur\ \Gamma_i \cap \Gamma, \\ u_i = u_\Gamma, & sur \Gamma_i \cap \Gamma_j. \end{cases} \tag{2.2}$$

On cherche la valeur de u sur l'interface $\Gamma_i \cap \Gamma_j$. En réordonnant habilement les inconnues, c'est-à-dire en regroupant les degrés de liberté internes au sous domaine X_1, puis les degrés de liberté internes au sous domaine X_2 et enfin ceux de l'interface $\Gamma_i \cap \Gamma_j$, la matrice associée présente une structure par bloc :

$$\begin{bmatrix} A_1 & 0 & -B_{1\Gamma} \\ 0 & A_2 & -B_{2\Gamma} \\ -D_{\Gamma 1} & -D_{\Gamma 2} & C_{\Gamma_{12}} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_{\Gamma_{12}} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ g \end{bmatrix}, \tag{2.3}$$

où u_i représente le vecteur des inconnues dans chaque sous domaine et $u_{\Gamma_{ij}}$ les inconnues à l'interface $\Gamma_i \cap \Gamma_j$. A_i et $C_{\Gamma_{12}}$ représentent les restrictions de l'opérateur L dans les sous domaines X_i et à l'interface de couplage $\Gamma_i \cap \Gamma_j$. Les $B_{i\Gamma}$ et $D_{\Gamma i}$ sont les matrices mettant en relation les inconnues à l'intérieur des sous domaines avec les inconnues liées à l'interface.

Ce système peut être ramené à un problème linéaire sur les inconnues aux interfaces :

$$S u_{\Gamma_{12}} = b$$

avec :

$$\begin{cases} S = C_{\Gamma_{12}} - D_{\Gamma 1} A_1^{-1} B_{1\Gamma} - D_{\Gamma 2} A_2^{-1} B_{2\Gamma}, \\ b = g + D_{\Gamma 1} A_1^{-1} f_1 + D_{\Gamma 2} A_2^{-1} f_2, \end{cases} \tag{2.4}$$

où S est le complément de Schur. Une fois le système sur les interfaces résolu par une méthode directe ou itérative, on obtient la solution dans les sous domaines par ($i = \{1,2\}$):

$$u_i = A_i^{-1}\left(f_i - B_{i\Gamma} u_{\Gamma_{12}}\right) \tag{2.5}$$

La construction du complément de Schur demande l'inversion des matrices A_{ii}. Cela revient à inverser le problème local pour chaque sous domaine qui peut être résolu par une inversion directe ou itérative.

2.3 Approche proposée : Décomposition de domaine par Bloc Multigroupe

2.3.1 Rappel du contexte

Dans le cadre de ce travail de thèse, nous cherchons à accélérer la résolution de l'équation du transport pour des calculs d'étude ou d'ingénierie visant à obtenir des résultats de « référence » rapides pour des problèmes de grande taille. Même si l'objectif n'est pas la parallélisation pour du calcul HPC, dont l'accès est encore limité, l'algorithme recherché a aussi l'ambition d'être performant sur ce type d'architecture. Pour notre travail, le nombre de processeurs que nous considérons est compris entre une dizaine (ordinateurs de bureau) et jusqu'à l'ordre du millier (calculateurs disponibles dans les laboratoires d'études). Par ailleurs un point essentiel pour satisfaire les utilisateurs d'APOLLO3® est la portabilité du code. Ainsi nous ne pouvons pas nous focaliser sur l'optimisation de la méthode sur une architecture donnée. Ces arguments nous amènent à considérer que toutes les variables, espace-angle-énergie, ne pourront pas être partitionnées sur si peu de processeurs. De ce fait, notre approche a comme principal objectif de paralléliser le calcul tout en optimisant l'utilisation de la mémoire et en limitant les copies redondantes de données. La Décomposition de Domaine (DD) spatiale semble alors le meilleur moyen de répondre à la problématique posée. En effet, les données physiques (géométrie, matériaux, flux,…) peuvent être intégralement distribuées par sous domaine. Par ailleurs, les solveurs actuels ont été développés pour être performants sur des calculs séquentiels avec un seul processeur, en optimisant notamment l'utilisation du cache par la localité temporelle et spatiale des données. Ces optimisations sont bonnes pour des calculs à l'échelle de l'assemblage en 2D, cependant lorsque la taille du problème spatial croit, les performances des codes se dégradent. La DD permet de découper le problème global en sous problèmes de taille réduite, pouvant se rapprocher de celle de l'assemblage, ce qui permet de retrouver le bon comportement des solveurs pour le calcul des sous domaines.

Enfin nous cherchons à minimiser l'overhead dû à la DD. Il s'agit alors de limiter le nombre de synchronisations et la quantité de données à communiquer entre les sous domaines. De cette manière nous espérons pouvoir réduire le temps d'exécution et minimiser l'impact de l'architecture du calculateur sur les performances.

2.3.2 Méthode proposée : décomposition de domaine spatiale par bloc multigroupe et algorithme de Jacobi par bloc combinés

2.3.2.a Description du problème à résoudre

Nous cherchons à résoudre l'équation du transport pour un problème posé sur un domaine D :

$$\begin{cases} A\psi = Q, & dans\ X, \\ \psi^{-} = 0, & sur\ \Gamma. \end{cases} \tag{2.6}$$

ψ est le flux à l'intérieur du domaine et ψ^{-} est le flux entrant nul sur la frontière Γ. A est l'opérateur de transport et Q est la source de fission normalisée. L'espace des phases est représenté comme suit :

$$\begin{aligned} X &\equiv (\boldsymbol{r} \in D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma^{\pm} &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_+(\boldsymbol{r}) \gtrless 0, E \in \mathbb{R}^+), \end{aligned} \tag{2.7}$$

où $\boldsymbol{r}$ est la position spatiale, $\boldsymbol{\Omega}$ la direction, et E est l'énergie. ∂D est la surface du domaine D. $\boldsymbol{n}^+(\boldsymbol{r})$ est la normale sortant du domaine D au point $\boldsymbol{r} \in \partial D$. Γ^+ et Γ^- sont respectivement les frontières sortantes et entrantes de l'espace des phases X. Notons que l'espace des phases est discrétisé dans le cadre de ce travail de thèse. Ces notations sont utilisées par simplicité.

2.3.2.b Décomposition de domaine et algorithme de résolution

L'approche proposée consiste à diviser le domaine spatial global en sous domaines sans recouvrement. Le choix du non recouvrement répond à l'attente d'optimisation de la mémoire. Prenons pour exemple le cas simple de la décomposition spatiale avec deux sous domaines D_1 et D_2 présenté sur la Figure 2.4.

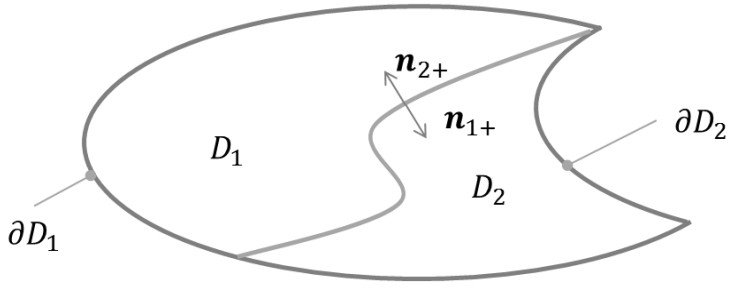


Figure 2.4. Exemple de décomposition de domaine sans recouvrement.

L'espace des phases de chaque sous domaine est défini par les notations suivantes :

$$\begin{aligned} X_i &\equiv (\boldsymbol{r} \in D_i, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma_i &\equiv (\boldsymbol{r} \in \partial D_i, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma_i^{\pm} &\equiv (\boldsymbol{r} \in \partial D_i, \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_+(\boldsymbol{r}) \gtrless 0, E \in \mathbb{R}^+), \end{aligned} \tag{2.8}$$

Dans chaque sous domaine $i = \{1,2\}$, on résout le problème suivant :

$$\begin{cases} A_i\psi_i = Q_i \ , & dans\ X_i, \\ \psi_i^- = 0 \ , & sur\ \Gamma_i \cap \Gamma, \\ \psi_i = \psi_{\Gamma_{12}}, & sur\ \Gamma_i \cap \Gamma_j, \end{cases} \tag{2.9}$$

où ψ_i est la restriction du flux angulaire multigroupe sur le domaine i et $\psi_{\Gamma_{12}}$ le flux angulaire à l'interface entre les sous domaines $\Gamma_{ij} \equiv \Gamma_i \cap \Gamma_j$. A_i et Q_i sont respectivement les restrictions de l'opérateur de transport et la source de fission au sous domaine i. Les conditions aux limites de vide sur Γ sont représentées par le flux entrant nul (2ème ligne). La solution globale est obtenue en imposant la continuité du flux angulaire à l'interface entre les sous domaines (3ème ligne). Nous appliquons la décomposition de domaine spatiale sans recouvrement telle que présentée à la section 2.2.3.b, équation (2.3), qui conduit à la formulation matricielle suivante :

$$\begin{bmatrix} A_1 & 0 & -B_{1\Gamma} \\ 0 & A_2 & -B_{2\Gamma} \\ -D_{\Gamma 1} & -D_{\Gamma 2} & C_{\Gamma_{12}} \end{bmatrix} \cdot \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_{\Gamma_{12}} \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ 0 \end{bmatrix}. \tag{2.10}$$

$B_{i\Gamma}$, et $D_{\Gamma i}$ sont les opérateurs de projection entre le flux dans le sous domaine et le flux à l'interface. $C_{\Gamma_{12}}$ est l'opérateur de couplage entre les flux d'interfaces.

Pour introduire correctement l'algorithme de résolution nous exprimons le flux d'interface par :

$$\psi_{\Gamma 1} = \begin{cases} \psi_{\Gamma 1,+}, & \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_{1+}(\boldsymbol{r}) > 0, \\ \psi_{\Gamma 1,-}, & \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_{1+}(\boldsymbol{r}) < 0, \end{cases}$$

et :

$$\psi_{\Gamma 2} = \begin{cases} \psi_{\Gamma 2,+}, & \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_{2+}(\boldsymbol{r}) > 0, \\ \psi_{\Gamma 2,-}, & \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_{2+}(\boldsymbol{r}) < 0. \end{cases}$$

Le flux à l'interface $\psi_{\Gamma_{12}}$ est dupliqué en $\psi_{\Gamma 1}$ et $\psi_{\Gamma 2}$ pour être représenté dans chaque sous domaine. Ici $\boldsymbol{n}_{i+}$ est la normale sortante du sous domaine i à l'interface entre les sous domaines. Le problème (2.10) devient alors :

$$\left[\begin{array}{cc|cccc} A_1 & 0 & 0 & 0 & -\tilde{B}_{1\Gamma} & 0 \\ 0 & A_2 & 0 & 0 & 0 & -\tilde{B}_{2\Gamma} \\ \hline -\tilde{D}_{\Gamma 1} & 0 & I & 0 & -\tilde{C}_1 & 0 \\ 0 & -\tilde{D}_{\Gamma 2} & 0 & I & 0 & \tilde{C}_2 \\ 0 & 0 & 0 & -I & I & 0 \\ 0 & 0 & -I & 0 & 0 & I \end{array}\right] \cdot \left[\begin{array}{c} \psi_1 \\ \psi_2 \\ \hline \psi_{\Gamma 1,+} \\ \psi_{\Gamma 2,+} \\ \psi_{\Gamma 1,-} \\ \psi_{\Gamma 2,-} \end{array}\right] = \left[\begin{array}{c} Q_1 \\ Q_2 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right], \tag{2.11}$$

La matrice (2.11) illustre les avantages de la DDM proposée. Les deux premières lignes sont les restrictions de l'équation du transport multigroupe à chaque sous domaine. Ces lignes sont indépendantes, donc l'inversion de l'opérateur de transport dans les sous domaines est parfaitement parallélisable. Les deux lignes suivantes sont

le calcul du flux sortant de chaque sous domaine à l'interface. Enfin les deux dernières lignes sont les transmissions des flux d'interface, seul point de communication entre les sous domaines.

Cette formulation permet de présenter l'algorithme de résolution de type Schwarz additif proposé dans cette thèse :

Algorithme 2.2. Algorithme de Schwarz additif appliqué à l'équation du transport.

$Jusqu'$à $convergence, répéter$:

$$\left|\begin{array}{ll} résoudre: & \begin{cases} A_1\psi_1^{(e+1)} = Q_1 + \tilde{B}_{1\Gamma}\psi_{\Gamma 1,-}^{(e)}, & dans\ X_1, \\ \psi_{\Gamma 1,+}^{(e+1)} = \widetilde{D}_{\Gamma 1}\psi_1^{(n+1)} + \tilde{C}_1\psi_{\Gamma 1,-}^{(e)}, & sur\ \ \Gamma_1 \cap \Gamma_2. \end{cases} \\ résoudre: & \begin{cases} A_2\psi_2^{(e+1)} = \ Q_2 + \tilde{B}_{2\Gamma}\psi_{\Gamma 2,-}^{(e)}, & dans\ X_2, \\ \psi_{\Gamma 2,+}^{(e+1)} = \widetilde{D}_{\Gamma 2}\psi_2^{(e+1)} + \tilde{C}_2\psi_{\Gamma 2,-}^{(e)}, & sur\ \Gamma_1 \cap \Gamma_2. \end{cases} \\ nouveau\ flux\ entrant: & \begin{cases} \psi_{\Gamma 1,-}^{(e+1)} = \psi_{\Gamma 2,+}^{(e+1)}, & sur\ \Gamma_1 \cap \Gamma_2, \\ \psi_{\Gamma 2,-}^{(e+1)} = \psi_{\Gamma 1,+}^{(e+1)}, & sur\ \Gamma_1 \cap \Gamma_2. \end{cases} \end{array}\right.$$

La résolution est itérative : le problème transport multigroupe complet est inversé dans chaque sous domaine. Pour les interfaces entre sous domaines, le flux angulaire multigroupe entrant est imposé comme condition aux limites du problème transport local. Ce flux entrant est le flux sortant des sous domaines voisins, calculé lors de l'itération externe précédente. Nous nommons cette méthode algorithme de Jacobi Parallèle par Bloc Multigroupe (PMBJ, acronyme de l'anglais Parallel Multigroup-Block Jacobi). En effet, non seulement il s'agit d'un algorithme de Jacobi puisque le flux d'interface est obtenu à l'itération précédente, mais en plus les blocs sont multigroupes. La mise à jour des flux n'est réalisée qu'à la fin de l'itération externe, d'où l'appellation PMBJ.

La méthode des caractéristiques courtes d'IDT (cf. §1.5), solveur S_N, a été utilisée pour l'inversion du problème transport. Ce solveur permet de représenter un réseau cartésien avec des crayons combustibles décrits finement. Ainsi son application au calcul de REL est tout à fait pertinente. De plus, le flux angulaire à la surface est représenté par projection sur des fonctions d'essai constantes ou linéaires. De ce fait son stockage a un coût réduit par rapport au MOC qui requiert de garder tous les points d'impact de trajectoires sur les interfaces entre les sous domaines. La DD peut alors être appliquée sans besoin d'une approximation à l'interface entre les sous domaines et, donc, à un moindre coût. L'identité de la solution de la méthode de DD avec celle obtenue par le solveur IDT standard est aussi assurée. C'est un outil non négligeable pour obtenir des éléments de vérification lors du développement du code.

2.3.3 Comparaison aux méthodes classiques de Décomposition de Domaine

DEFAUTS DE L'ALGORITHME PMBJ

Le défaut majeur de l'algorithme PMBJ vient du fait qu'il détériore considérablement les propriétés de convergence de l'algorithme classique. Considérons le problème à l'intérieur d'un groupe, c'est-à-dire dans les itérations internes. L'inversion directe de l'opérateur de transport réalisée par le balayage spatial est rompue à l'interface entre les sous domaines puisque nous utilisons un algorithme de Jacobi. De plus, le flux à l'interface entre les sous domaines n'est jamais mis à jour au cours des itérations internes. Ainsi, la solution obtenue, même si les itérations internes ont convergé dans chaque sous domaine, n'est alors pas la solution convergée du problème global. Il faut plusieurs itérations externes pour converger la solution dans un groupe. De plus cette erreur dans les itérations internes se propage dans la résolution des autres groupes : la source de scattering multigroupe est alors entachée de l'erreur sur le flux. De ce fait, les sources de scattering ne sont convergées que localement, dans les sous domaines. Ainsi, même pour les groupes rapides, plusieurs itérations multigroupes sont nécessaires pour converger la solution à l'échelle globale. De même pour les groupes thermiques, l'algorithme de Gauss-Seidel ne permet de converger les sources de scattering que localement. Ces considérations nous amènent évidemment à considérer qu'il sera nécessaire d'accélérer l'algorithme pour espérer obtenir une scalabilité idéale vis-à-vis de l'algorithme classique de résolution de l'équation du transport décrit à la section 1.4.4.

De plus, un overhead est associé au stockage du flux angulaire à la surface des sous domaines. Si le coût du stockage des quantités surfaciques aux frontières des sous domaines est associé à toutes les DDM spatiales, dans le cas de l'algorithme PMBJ l'overhead est plus important. Le flux doit être stocké pour tous les groupes d'énergie. Cependant il faut comparer cet overhead au coût de stockage des quantités volumiques. Pour un calcul d'évolution, le grand nombre de matériaux à décrire engendre une quantité de mémoire importante requise pour le stockage des matrices de transfert. Prenons l'exemple d'un problème donné sur géométrie cartésienne 3D comprenant N_x mailles par direction, N_g groupes d'énergie et un développement angulaire avec N_h moments. La mémoire requise pour le stockage des matrices de transfert, notée M_v, peut être estimée de la manière suivante :

$$M_v = N_x^3 \times N_g \times N_e \times N_h \, ,$$

où le nombre moyen d'éléments par ligne dans la matrice de transfert est noté N_e. La quantité de mémoire pour les données surfaciques, notée M_s, est estimée par :

$$M_s = 6N_x^2 \times N_g \times N_d \times N_b .$$

Ici, N_d est le nombre de directions et N_b le nombre de moments spatiaux sur les surfaces. Pour le calcul d'un assemblage de REL, on a généralement $N_x = 17$, $N_g = 281$, $N_e \cong 160$, $N_d \cong 100$, $N_b = 3$ (développement spatial linéaire) et $N_h = 16$

(ordre d'anisotropie P3). L'overhead, qui est définit par M_s/M_v, dû au stockage des flux angulaires multigroupes sur les surfaces est de l'ordre de 2.5%. L'overhead de la DDM ne sera donc pas le facteur limitant pour réaliser un tel calcul. Cependant, pour des modélisations simplifiées où le nombre de matériaux est réduit (i.e. le nombre de matrices de transfert est réduit), le stockage du flux angulaire peut devenir la principale cause de l'occupation mémoire avec les moments du flux multigroupe.

AVANTAGES DE L'ALGORITHME PMBJ

L'algorithme PMBJ a plusieurs avantages par rapport aux méthodes de DD présentées à la section 2.2.2. Je rappelle qu'un nom spécifique a été introduit pour désigner les algorithmes faisant référence aux méthodes de DD appliquées au niveau du balayage spatio-angulaire : algorithme Parallèle par Blocs dans 1 Groupe (PB1G). Dans cette partie nous ne considérons que la DD spatiale pour la comparaison avec l'algorithme PMBJ. Nous faisons l'hypothèse que les itérations internes et que les itérations thermiques sont convergées de manière à estimer les gains apportés par l'algorithme PMBJ. Les différents avantages qui nous ont orientés vers l'algorithme PMBJ sont les suivants :

- Réduction du nombre de synchronisations par itération externe
- Réduction du nombre et du volume de communications par itération externe
- Influence sur le temps de calcul d'une itération externe
- Implémentation
- Parallélisme

Ces avantages sont détaillés ci-dessous :

1. Réduction du nombre de synchronisations par itération externe
   - Bilan des synchronisations pour les algorithmes PB1G.

L'algorithme de KBA [28] permet d'inverser directement l'opérateur de transport à l'échelle du domaine global. Les synchronisations sont alors provoquées par la contrainte d'ordre de calcul des sous domaines pour chaque direction discrète : le sous domaine qui commence le calcul est différent pour chaque octant. D'autre part l'algorithme PBJ permet de réduire le nombre de synchronisation au prix de la dégradation des propriétés de convergence. Une seule synchronisation est nécessaire pour chaque itération interne. Le nombre de synchronisations N_{PBJ}^{sync} est alors :

$$N_{PBJ}^{sync} = \sum_{g=1}^{N_{fast}} N_{in}(g) + \sum_{th=1}^{N_{th}} \sum_{g=N_{fast}+1}^{N_g} N_{in}(g)\,,$$

où N_{fast} est le nombre de groupes rapides, N_{th} et $N_{in}(g)$ sont respectivement le nombre d'itérations thermiques nécessaires pour converger et le nombre d'itérations internes nécessaires pour converger dans le groupe g. Dans la suite nous considérons l'algorithme PBJ comme optimal pour la réduction du nombre de synchronisations avec les méthodes PB1G.

- Bilan des synchronisations pour l'algorithme PMBJ.

Une seule synchronisation est requise avec l'algorithme PMBJ: lors du calcul de la valeur propre et de la source de fission. A cette exception, les sous domaines sont calculés complètement indépendamment les uns des autres. On a alors $N_{PMBJ}^{sync} = 1$.

Le nombre de synchronisations est très fortement réduit par l'algorithme PMBJ. Il est aussi important de noter que le nombre de synchronisations ne dépend ni du nombre de groupes, ni du nombre d'itérations thermiques, ni du nombre d'itérations internes, contrairement aux algorithmes PB1G.

2. Réduction du nombre et du volume de communications par itération externe

Le nombre de communications est égal au nombre de synchronisations, drastiquement réduit par l'algorithme PMBJ. Par contre le volume de communications est calculé différemment. Nous ne considérons ici que les communications liées à la transmission du flux à l'interface entre les sous domaines.

- Volume de communication pour les algorithmes PB1G.

Le volume de communication pour l'algorithme PB1G est donné par :

$$V_{PB1G} = \sum_{g=1}^{N_{fast}} V_s N_{in}(g) + \sum_{th=1}^{N_{th}} \sum_{g=N_{fast}+1}^{N_g} V_s N_{in}(g)\,,$$

où V_s est le nombre total d'inconnues du flux angulaire d'un groupe pour toutes les interfaces entre les sous domaines. Notons que $V_{PB1G} = V_s N_{PBJ}^{sync}$.

- Volume de communication pour l'algorithme PMBJ.

Pour l'algorithme PMBJ, le volume de communication est exprimé par :

$$V_{PMBJ} = V_s N_g$$

Il ne dépend pas du nombre d'itérations. A moins que $N_{in}(g)$ soit égal à 1 pour tous les groupes et qu'il n'y ait pas de groupe thermique, le volume de communication de l'algorithme PMBJ est inférieur à celui de l'algorithme PB1G. Nous pouvons aussi noter que le volume par communication est différent. Pour les algorithmes PB1G, le volume V_s est engagé par communication. Par contre pour l'algorithme PMBJ, le volume $(V_s N_g)$ est communiqué en une seule fois. Avec une implémentation MPI, la réduction du nombre de communications permet d'éviter les surcouts liés à l'envoi de chaque message.

3. Influence sur le temps de calcul d'une itération externe

Nous considérons ici le temps de calcul d'une itération externe en supposant une décomposition en α sous domaines, parallélisée sur α processeurs. Dans l'évaluation du temps de calcul nous ne considérons que le temps lié aux itérations internes. Le temps du balayage spatio-angulaire, noté T_α^{sw}, est constant dans un sous domaine.

- Temps de calcul pour les algorithmes PB1G.

Avec ces méthodes, le nombre d'itérations internes et thermiques sont les mêmes pour chaque sous domaine. Le nombre de balayages spatio-angulaire N^{sw} dans une itération externe est :

$$N^{sw} = \sum_{g=1}^{N_{fast}} N_{in}(g) + \sum_{th=1}^{N_{th}} \sum_{g=N_{fast}+1}^{N_g} N_{in}(g). \tag{2.12}$$

Le temps de calcul pour l'algorithme PB1G est donné par :

$$T_{PB1G} = T_{max}^{sw} \times N^{sw}, \qquad T_{max}^{sw} = \max_{\forall \alpha}(T_{\alpha}^{sw}), \tag{2.13}$$

où T_{max}^{sw} est le temps maximal de balayage spatio-angulaire parmi tous les sous domaines. Le temps de calcul est obtenu multipliant T_{max}^{sw} par N^{sw}. De ce fait les méthodes PB1G sont très sensibles au déséquilibrage de charge. La perte de temps due au déséquilibrage augmente avec le nombre de groupes, d'itérations thermiques et d'itérations internes par groupe.

De plus le nombre d'itérations internes et le nombre d'itérations thermiques dépend du rayon spectral du problème global. Prenons l'exemple du calcul dans un groupe g quelconque : puisque $N_{in}(g)$ est contraint par le plus grand rayon spectral parmi tous les sous domaines, on a $N_{in}(g) \geq \max_{\forall \alpha}(N_{in}^{\alpha}(g))$. $N_{in}^{\alpha}(g)$ est le nombre d'itérations internes nécessaires pour converger dans le sous domaine α isolé. Le temps de calcul dans le groupe g est alors :

$$T_{PB1G}^{g} \geq \max_{\forall \alpha}(N_{in}^{\alpha}(g)) \times \max_{\forall \alpha}(T_{\alpha}^{sw}). \tag{2.14}$$

Et de même pour le nombre d'itérations thermiques : $N_{th} \geq \max_{\forall \alpha}(N_{th}^{\alpha})$.

- Temps de calcul pour les algorithmes PMBJ.

Pour l'algorithme PMBJ, le nombre d'itérations internes $N_{in}^{\alpha}(g)$ et le nombre d'itérations thermiques N_{th}^{α} nécessaires pour converger sont propres à chaque sous domaine. Ils sont conditionnés par le rayon spectral du sous domaine. Il en découle que le nombre de balayages spatio-angulaire par sous domaine et pour une itération externe, noté N_{α}^{sw}, est estimé par :

$$N_{\alpha}^{sw} = \sum_{g=1}^{N_{fast}^{\alpha}} N_{in}^{\alpha}(g) + \sum_{th=1}^{N_{th}^{\alpha}} \sum_{g=N_{fast}^{\alpha}+1}^{N_g} N_{in}^{\alpha}(g) \tag{2.15}$$

Le temps de calcul d'une itération externe est alors donné par :

$$T_{PMBJ} = \max_{\forall \alpha}[T_{\alpha}^{sw} \times N_{\alpha}^{sw}] \tag{2.16}$$

La définition locale du nombre d'itérations internes et du nombre d'itérations thermiques permet d'éviter de recalculer les sous domaines où les sources de scattering sont déjà convergées. Pour le calcul dans un groupe. On a alors :

$$T_{PMBJ}^{g} = \max_{\forall \alpha}(N_{in}^{\alpha}(g) \times T_{\alpha}^{sw}) \leq T_{PB1G}^{g}. \quad (2.17)$$

Dans l'algorithme PMBJ, le déséquilibrage de charge entrainé par le sous domaine $\beta: T_{\beta}^{sw} = \max_{\forall \alpha}(T_{\alpha}^{sw})$ est réduit si $N_{in}^{\beta}(g) \neq \max_{\forall \alpha}(N_{in}^{\alpha}(g))$. Le même raisonnement est applicable aux itérations thermiques.

De plus puisque la seule synchronisation de l'algorithme PMBJ est à la fin de l'itération externe, les itérations multigroupes sont entièrement désynchronisées. Le temps de calcul dans un groupe $T_{\alpha,g}$ variant d'un groupe à l'autre et entre les sous domaines, l'algorithme PMBJ permet d'épargner du temps par rapport aux algorithmes PB1G :

$$T_{PMBJ} = \max_{\forall \alpha}\left(\sum_{g=1}^{N_g} T_{\alpha,g}\right) \leq \; T_{PB1G} = \sum_{g=1}^{N_g} \max_{\forall \alpha}(T_{\alpha,g}) \quad (2.18)$$

4. Implémentation

En ce qui concerne l'implémentation des algorithmes PB1G et PMBJ, les contraintes sont différentes.

Les algorithmes PB1G interfèrent dans le balayage spatio-angulaire qui est la partie de l'algorithme où le calcul est intensif. Il faut donc veiller avec attention à ce que les directives parallèles dégradent au minimum les performances.

Au contraire, les directives parallèles pour l'algorithme PMBJ sont placées dans les itérations externes. De ce fait, elles sont nettement moins soumises à l'intensité du calcul et moins susceptibles de dégrader les performances. La fréquence des appels aux directives parallèles étant bien inférieure, une implémentation non optimale est moins pénalisante. Pour une application dont l'objectif est d'être portable sur une quelconque architecture, cet argument est important puisque les implémentations optimales sont généralement machine-dépendantes. L'approche par l'algorithme PMBJ permet aussi de minimiser les modifications dans un code séquentiel. Les routines réalisant un calcul multigroupe classique peuvent directement être réutilisées. Seul le calcul de la valeur propre globale doit être modifié et des routines dédiées aux communications entre les sous domaines doivent être ajoutées. Paralléliser un solveur séquentiel par cette méthode est alors très simple, même sans maîtriser les sources du solveur. De même, les méthodes d'accélération de calculs à sources déjà présentes dans le solveur peuvent être réutilisées (accélération des itérations internes et thermiques) pour accélérer la convergence des itérations locales. Finalement, la parallélisation intrinsèque au solveur peut être utilisée dans un parallélisme hiérarchisé. L'algorithme PMBJ permet de réaliser la décomposition de domaine alors que le solveur lui-même peut être parallélisé sur la variable angulaire dans les itérations internes.

5. Parallélisme

L'un des objectifs étant de proposer une méthode optimisée pour les machines de bureau, le parallélisme à mémoire partagée a été notre premier axe de développement. Une parallélisation coarse grained présente l'avantage de nécessiter bien moins de créations et destructions de threads, opérations impliquant systématiquement un overhead. Dans notre cas, l'algorithme a été implémenté avec seulement 2 zones parallèles et presqu'aucune directive en dehors de la routine réalisant les itérations externes. Une parallélisation à mémoire partagée coarse grained est souvent décrite comme difficile à implémenter. Le statut privé ou partagé des variables doit être suivi avec beaucoup d'attention, de même que la synchronisation de la mémoire entre les threads. Avec l'algorithme PMBJ, chaque sous domaine est considéré comme un solveur indépendant. Toutes les variables ont alors le statut privé, sans avoir à les différencier. Les détails de l'implémentation pour le parallélisme à mémoire partagée sont donnés à la section 4.3.

Un calcul de cœur en début de cycle à 281 groupes d'énergie et en 2D requiert quelques GigaBytes (GB) de données, principalement en raison du stockage du flux. Cette taille de problème est adaptée aux ordinateurs de bureau actuels. Cependant, pour un calcul en évolution, le nombre de matériaux pouvant évoluer différemment est proche du nombre de régions. Ainsi, la quantité de mémoire due au stockage des sections efficaces demande plusieurs centaines de GB et va jusqu'à quelques TeraBytes (TB) en comptabilisant le coût du stockage des coefficients transport. Pour réaliser ce genre de calcul, le parallélisme à mémoire distribuée a été implémenté. Il permet d'accéder non seulement à plus de mémoire, mais aussi à un nombre plus important de processeurs. La section 4.4 est dédiée à la description de cette implémentation.

Dans cette section, nous avons cherché à minimiser au maximum les communications tout en optimisant la distribution de la mémoire. La méthode choisie est une décomposition de domaine spatiale avec un algorithme de Jacobi Parallèle par Bloc Multigroupe. Cet algorithme entraine un retard dans l'inversion de l'opérateur de transport mais aussi dans l'inversion de l'opérateur de scattering multigroupe. Par contre il permet d'épargner un grand nombre de communications et de synchronisations lors d'un calcul parallèle. L'algorithme est certainement moins performant que l'algorithme de résolution classique. De ce fait des méthodes d'accélération doivent être mises en place. Finalement, il s'avère qu'il est très aisé à programmer, ne nécessitant que peu de modifications du solveur transport séquentiel. Dans la section suivante nous présentons le formalisme de l'algorithme PMBJ.

2.4 Formalisme de la Méthode de Décomposition de Domaine par Bloc Multigroupe combinée à l'algorithme de Jacobi

Dans cette section, nous reprenons la présentation de l'algorithme PMBJ introduit dans la section précédente de manière à préciser les subtilités de sa réalisation.

2.4.1 Description du problème à résoudre et prise en compte des conditions aux limites

Nous cherchons à résoudre l'équation du transport pour un problème posé sur un domaine D présenté Figure 2.5 :

$$\begin{cases} (L-H)\psi(x) = Q(x), & x \in X, \\ \psi(x) = \psi^{-}(x), & x \in \Gamma^{-}. \end{cases} \tag{2.19}$$

$\psi(x)$ est le flux à l'intérieur du domaine et $\psi^{\pm}(x)$ est le flux à sa surface défini pour $x \in \Gamma^{\pm}$. $(L-H) \equiv \Omega \cdot \nabla + \Sigma(x) - \int dx' \Sigma_s(x' \to x)$ est l'opérateur de transport multigroupe et Q est la source de fission normalisée dans le domaine D:

$$Q(x) = \frac{F\psi(x)}{\lambda}, \qquad x \in X. \tag{2.20}$$

λ est la valeur propre du problème et F l'opérateur de fission. Je rappelle que l'espace des phases est défini comme suit :

$$\begin{aligned} X &\equiv (\boldsymbol{r} \in D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma^{\pm} &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_+(\boldsymbol{r}) \gtrless 0, E \in \mathbb{R}^+), \end{aligned} \tag{2.21}$$

où $\boldsymbol{r}$ est la position spatiale, $\boldsymbol{\Omega}$ la direction, et E est l'énergie. ∂D est la surface du domaine D. $\boldsymbol{n}_+(\boldsymbol{r})$ est la normale sortant du domaine D au point $\boldsymbol{r} \in \partial D$. Γ^+ et Γ^- sont respectivement les frontières sortantes et entrantes de l'espace des phases X. La Figure 2.5 représente le domaine D avec les notations utilisées pour la représentation des conditions aux limites.

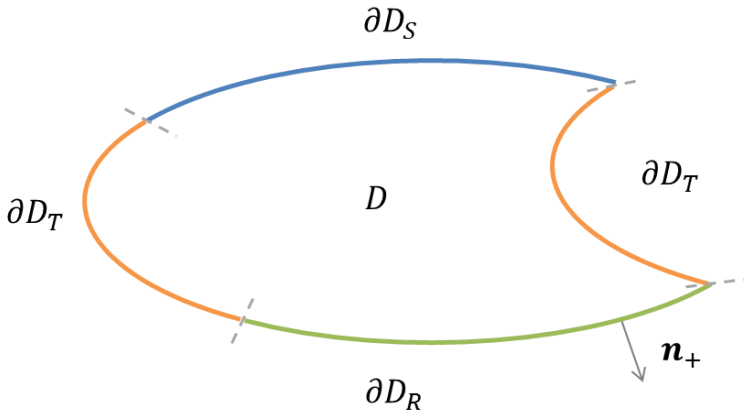


Figure 2.5. Exemple de Domaine et prise en compte des conditions aux limites.

La surface du domaine est divisée en plusieurs parties correspondant aux différentes conditions aux limites : flux entrant imposé sur ∂D_S, réflexion sur ∂D_R et translation sur ∂D_T. Nous définissons le flux entrant dans le domaine selon les conditions aux limites considérées :

$$\psi^{-}(x)=\begin{cases}\psi_S(x)=S(x) & , \quad x\in\Gamma_S\equiv(\boldsymbol{r}\in\partial D_S\,,\boldsymbol{\Omega}\in S^2\colon\boldsymbol{\Omega}\cdot\boldsymbol{n}^{+}<0,E\in\mathbb{R}^{+}),\\ \psi_R(x)=\psi^{+}(R^{-1}x), & x\in\Gamma_R\equiv(\boldsymbol{r}\in\partial D_R,\boldsymbol{\Omega}\in S^2\colon\boldsymbol{\Omega}\cdot\boldsymbol{n}^{+}<0,E\in\mathbb{R}^{+}),\\ \psi_T(x)=\psi^{+}(T^{-1}x), & x\in\Gamma_T\equiv(\boldsymbol{r}\in\partial D_T,\boldsymbol{\Omega}\in S^2\colon\boldsymbol{\Omega}\cdot\boldsymbol{n}^{+}<0,E\in\mathbb{R}^{+}).\end{cases} \tag{2.22}$$

La première ligne est une condition aux limites de source externe imposée notée $S(x)$. Pour un calcul à valeur propre, c'est la condition aux limites de vide $S(x)=0$. Les 2ème et 3ème lignes sont les conditions aux limites homogènes de réflexion et de translation. R et T sont respectivement les opérateurs de réflexion et translation définis par :

$$\begin{array}{lll} R:X\to X\,, & x(\boldsymbol{r},\boldsymbol{\Omega},E)\longmapsto x'(\boldsymbol{r},\boldsymbol{\Omega}',E), & \boldsymbol{\Omega}'=\boldsymbol{\Omega}-2|\boldsymbol{\Omega}\cdot\boldsymbol{n}(\boldsymbol{r})|\boldsymbol{n}(\boldsymbol{r}),\\ T:X\to X\,, & x(\boldsymbol{r},\boldsymbol{\Omega},E)\longmapsto x'(\boldsymbol{r}',\boldsymbol{\Omega},E), & \boldsymbol{r}'=\boldsymbol{r}-\boldsymbol{r_T}, \end{array} \tag{2.23}$$

où $\boldsymbol{r_T}$ est le vecteur de translation.

2.4.2 Décomposition de Domaine Spatiale par Bloc Multigroupe

Le domaine D est divisé en A sous domaines $D_\alpha\,,\alpha=1,\dots,A$, tels que $D=\bigcup_{\alpha=1,A}D_\alpha$. En appliquant la Décomposition de Domaine (DD) spatiale, l'espace des phases X est décomposé en sous espaces définis par :

$$\begin{array}{lll} X_\alpha & \equiv & (\boldsymbol{r}\in D_\alpha,\boldsymbol{\Omega}\in S^2,E\in\mathbb{R}^{+}),\\ \Gamma_\alpha & \equiv & (\boldsymbol{r}\in\partial D_\alpha,\boldsymbol{\Omega}\in S^2,E\in\mathbb{R}^{+}),\\ \Gamma_\alpha^{\pm} & \equiv & (\boldsymbol{r}\in\partial D_\alpha,\boldsymbol{\Omega}\in S^2\colon\boldsymbol{\Omega}\cdot\boldsymbol{n}_\alpha^{+}\gtrless 0,E\in\mathbb{R}^{+}). \end{array} \tag{2.24}$$

∂D_α est la surface du domaine D_α, $\boldsymbol{n}_\alpha^{+}(\boldsymbol{r})$ est la normale sortant de D_α au point $\boldsymbol{r}\in\partial D_\alpha$. Γ_α^{+} et Γ_α^{-} sont respectivement les surfaces sortantes et entrantes de D.

Le flux $\psi(x)$ dans les sous domaines est décomposé de la manière suivante :

$$\psi(x)=\sum_{\alpha=1}^{A}\chi_\alpha(\boldsymbol{r})\psi_\alpha(x) \quad , \quad x\in X, \tag{2.25}$$

où $\chi_\alpha(\boldsymbol{r})$ est la fonction caractéristique de D_α (à ne pas confondre avec le spectre de fission) :

$$\chi_\alpha(\boldsymbol{r})=\begin{cases}1, & \boldsymbol{r}\in D_\alpha,\\ 0, & sinon.\end{cases} \tag{2.26}$$

La continuité du flux angulaire est imposée à l'interface entre les sous domaines. Le flux angulaire sur les frontières entrantes et sortantes est noté $\psi_\alpha^{\pm}(x)$ et défini pour $x\in\Gamma_\alpha^{\pm}$. Le flux entrant est obtenu de la manière suivante en considérant les conditions aux limites étudiées au bord de D :

$$\psi_\alpha^{-}(x)\;=\begin{cases}\psi_\beta^{+}(x), & \forall\,\beta\neq\alpha:\ x\in\Gamma_\alpha^{-}\cap\Gamma_\beta^{+},\\ S(x), & x\in\Gamma_\alpha^{-}\cap\Gamma_S,\\ \psi_\alpha^{+}(R^{-1}x), & x\in\Gamma_\alpha^{-}\cap\Gamma_R,\\ \psi_\beta^{+}(T^{-1}x), & \forall\,\beta:\ (T^{-1}x)\in\Gamma_\beta^{+}\cap\Gamma_T.\end{cases} \tag{2.27}$$

A l'interface, le flux angulaire sortant d'un sous domaine est prolongé dans le sous domaine adjacent (1ère ligne). Une autre « interface » entre sous domaines peut être définie lorsqu'une condition aux limites de translation est présente au bord du domaine global et relie deux interfaces différentes (4ème ligne). Dans les autres situations, les conditions aux limites du problème sont décrites de la même manière que pour un domaine isolé, à savoir les conditions aux limites de source externe imposée et de réflexion (lignes 2 et 3).

L'équation du transport présentée en (2.19) est décomposée en $\alpha = 1, \dots, A$ problèmes multigroupes à source :

$$\begin{cases} (L-H)_\alpha \psi_\alpha(x) = Q_\alpha(x), & x \in X_\alpha, \\ \psi_\alpha(x) = \psi_\alpha^-(x), & x \in \Gamma_\alpha^-, \end{cases} \tag{2.28}$$

où $\psi_\alpha^-(x)$ est le flux entrant dans le sous domaine, $(L-H)_\alpha$ est l'opérateur de transport multigroupe et Q_α est la source de fission normalisée dans le sous domaine D_α:

$$Q_\alpha(x) = \frac{F_\alpha \psi_\alpha(x)}{\lambda}, \qquad x \in X_\alpha. \tag{2.29}$$

λ est la valeur propre du problème global tandis que F_α est l'opérateur de fission du sous domaine D_α.

2.4.3 Résolution par l'algorithme de Jacobi Parallèle par Bloc Multigroupes

Nous avons dans un premier temps implémenté l'algorithme de Jacobi Parallèle par Bloc Multigroupes (PMBJ) comme expliqué à la section 2.2.3. Avec ce type d'algorithme le flux multigroupe utilisé comme condition aux limites des sous domaines est calculé à l'itération externe précédente. Cette méthode à l'avantage de rendre indépendant les calculs multigroupes des sous domaines, permettant alors de paralléliser leur calcul. L'ordre dans lequel les sous domaines sont calculés n'a pas d'importance, contrairement aux algorithmes de Gauss-Seidel. L'objectif étant d'accélérer la résolution en utilisant le calcul parallèle, nous avons favorisé l'algorithme permettant de minimiser les communications et les synchronisations, sans contrainte sur l'ordre de balayage des sous domaines.

Avant de démarrer l'algorithme itératif, une étape d'initialisation est réalisée. Les flux aux interfaces des sous domaines et les sources de fission sont initialisés à partir du calcul non convergés des sous domaines en milieu infini :

$$\begin{cases} (L-H)_\alpha \psi_\alpha(x) = Q_\alpha(x), & x \in X_\alpha, \\ \psi_\alpha^-(x) = \psi_\alpha^+(R^{-1}x), & x \in \Gamma_\alpha^-. \end{cases} \tag{2.30}$$

La source de fission est normalisée par la valeur propre calculée localement :

$$Q_\alpha = \frac{F_\alpha \psi_\alpha(x)}{\lambda_\alpha}. \tag{2.31}$$

Après cette étape d'initialisation, l'algorithme consiste ensuite à résoudre le problème par des itérations de puissance sur la source de fission et sur les flux aux interfaces entre les sous domaines. Ce sont les itérations externes ayant pour indice e. L'implémentation a été réalisée selon l'algorithme PMBJ décrit par l'Algorithme 2.3 ci-après. Les 4 étapes suivantes constituent une itération externe.

1. **Etablissement des conditions aux limites de chaque sous domaine.**

Le flux entrant $\psi_\alpha^-(x)$ est initialisé pour chaque sous domaine $D_\alpha,\ \alpha \in A$. Comme décrit dans l'équation (2.27), le flux entrant est obtenu soit par le flux sortant des sous domaines adjacents soit par les conditions aux limites du problème global.

Pour les conditions aux limites homogènes d'un sous domaine, le flux à la surface est mis à jour à l'intérieur du balayage spatial et angulaire transport lors du calcul du sous domaine. C'est le cas pour les réflexions et les translations reliant deux bords d'un même sous domaine :

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\alpha^{+(e+1)}(R^{-1}x), & x \in \Gamma_\alpha^- \cap \Gamma_R, \\ \psi_\alpha^{+(e+1)}(T^{-1}x), & (T^{-1}x) \in \Gamma_\alpha^+ \cap \Gamma_T. \end{cases} \tag{2.32}$$

Lorsque les conditions aux limites du sous domaine sont non-homogènes et que le contact est fait avec d'autres sous domaines, le flux est obtenu par l'itération externe précédente :

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha :\ x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha :\ (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases} \tag{2.33}$$

Lorsqu'une source externe $S(x)$ est imposée à une condition limite on a :

$$\psi_\alpha^{-(e+1)}(x) = S(x)\,, \quad x \in \Gamma_\alpha^- \cap \Gamma_S\,, \qquad \forall\, e \geq 0. \tag{2.34}$$

Pour un problème à valeur propre $S(x) = 0$.

2. **Résolution du problème de transport à l'intérieur des sous domaines.**

Pour chaque sous domaine, le problème multigroupe à source fixe (source de fission et flux entrant) est ainsi résolu :

$$\begin{cases} (L-H)_\alpha \psi_\alpha{}^{(e+1)}(x) = Q_\alpha{}^{(e)}(x), & x \in X_\alpha, \\ \psi_\alpha{}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases} \tag{2.35}$$

L'opérateur de transport multigroupe est localement inversé. Un balayage multigroupe complet ainsi que les itérations thermiques sont effectués pour chaque sous domaine. Dans chaque groupe, les itérations internes sur la source de self-scattering sont réalisées localement, sans mise à jour du flux aux interfaces entre les sous domaines.

3. Mise à jour de la source de fission et de la valeur propre globale.

La valeur propre est calculée au niveau du domaine global :

$$\lambda^{(e+1)} = \lambda^{(e)} \frac{\sum_{\alpha \in A} \langle \boldsymbol{F}_\alpha{}^{(e)}, \boldsymbol{F}_\alpha{}^{(e+1)} \rangle}{\sum_{\alpha \in A} \langle \boldsymbol{F}_\alpha{}^{(e)}, \boldsymbol{F}_\alpha{}^{(e)} \rangle}, \tag{2.36}$$

Avec $\boldsymbol{F}_\alpha$, le vecteur des intégrales de fission par isotope (indice is), calculé par sous domaine :

$$\boldsymbol{F}_\alpha(\boldsymbol{r}) = \left\{ F_{is}(\boldsymbol{r}) = \sum_{g}^{N_g} \left(\nu\Sigma_f\right)_{is}^{g}(\boldsymbol{r}) \phi^g(\boldsymbol{r}) \ , \qquad \boldsymbol{r} \in D_\alpha \right\}. \tag{2.37}$$

Enfin, la source de fission est normalisée par la valeur propre dans chaque sous domaine :

$$Q_\alpha{}^{(e+1)}(x) = \frac{(F\psi)_\alpha{}^{(e+1)}(x)}{\lambda^{(e+1)}}, \tag{2.38}$$

avec :

$$(F\psi)_\alpha{}^{(e+1)}(x) = \sum_{is}^{N_f} \frac{\chi_{is}(E)}{4\pi} F_{is}^{(e+1)}(\boldsymbol{r}) \ , \qquad \boldsymbol{r} \in D_\alpha. \tag{2.39}$$

4. Tests de convergence.

Les tests portent sur l'erreur des sources de fission (err_f), de la valeur propre (err_λ) et des courants aux interfaces (err_J) pour tous les sous domaines :

$$\begin{cases} err_f = \max\limits_{\forall \boldsymbol{r}, \forall is} \left| 1 - \dfrac{F_{is}{}^{(e+1)}(\boldsymbol{r})}{F_{is}{}^{(e)}(\boldsymbol{r})} \right| < \ \varepsilon_f, \\ err_\lambda = \left| 1 - \dfrac{\lambda^{(e+1)}}{\lambda^{(e)}} \right| < \ \varepsilon_\lambda \ , \\ err_J = \max\limits_{\forall \alpha \in A \ , \forall x \in \Gamma_{\alpha-}} \left| 1 - \dfrac{\int d\Omega \ |\boldsymbol{\Omega}.\boldsymbol{n}_\alpha^+| \ \psi_\alpha{}^{(e+1)}(x)}{\int d\Omega \ |\boldsymbol{\Omega}.\boldsymbol{n}_\alpha^+| \ \psi_\alpha{}^{(e)}(x)} \right| < \ \varepsilon_J, \end{cases} \tag{2.40}$$

où $\varepsilon_f, \varepsilon_\lambda$ et ε_J sont respectivement les tolérances sur la distribution de la source de fission, sur la réactivité et sur les courants d'interface. Les itérations se terminent lorsque les critères de l'équation (2.40) sont vérifiés simultanément.

Ces étapes sont résumées dans Algorithme 2.3.

Algorithme 2.3. Algorithme de Jacobi Parallèle par Bloc Multigroupe.

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha \; \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux aux interfaces entre sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha : \; x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha : \; (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+1)}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Calcul de l'intégrale de fission

Pour chaque $\alpha \in A$, ***faire***

$$\boldsymbol{F}_\alpha^{(e+1)}(\boldsymbol{r}) = \left\{ F_{is}(\boldsymbol{r}) = \sum_g^{N_g} \left(\nu\Sigma_f\right)_{is}^g(\boldsymbol{r}) \phi_g^{(e+1)}(\boldsymbol{r}) \;\; , \qquad \boldsymbol{r} \in D_\alpha \right\}.$$

fin

Mise à jour de la valeur propre globale

$$\lambda^{(e+1)} = \lambda^{(e)} \frac{\sum_{\alpha \in A} \langle {\boldsymbol{F}_\alpha}^{(e)}, {\boldsymbol{F}_\alpha}^{(e+1)} \rangle}{\sum_{\alpha \in A} \langle {\boldsymbol{F}_\alpha}^{(e)}, {\boldsymbol{F}_\alpha}^{(e)} \rangle}.$$

Normalisation de la source de fission

Pour chaque $\alpha \in A$, ***faire***

$${Q_\alpha}^{(e+1)}(x) = \frac{(F\psi)_\alpha^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

fin

2.4.4 L'algorithme PMBJ avec le solveur IDT

La méthode de décomposition de domaine développée a été implémentée avec le solveur IDT décrit au paragraphe 1.5.

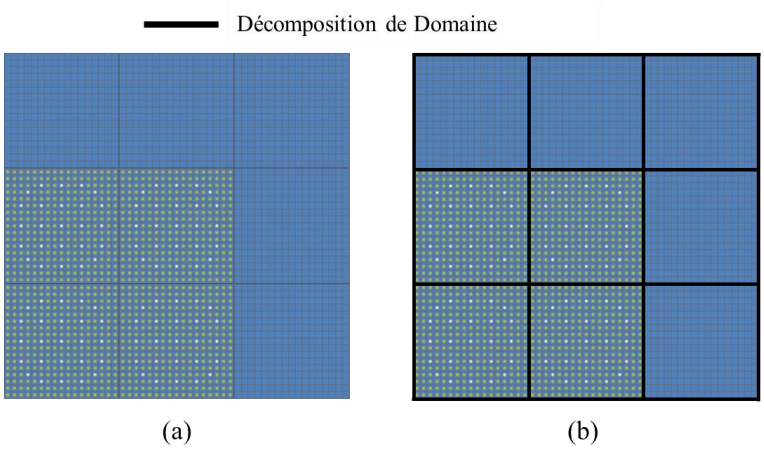


Figure 2.6. Exemple de Décomposition de domaine : le domaine global (a) est décomposé en 9 sous domaines (b) suivant un maillage cartésien.

Elle permet de diviser le domaine de calcul global en sous domaines définis par un maillage cartésien global comme le montre l'exemple de la Figure 2.6. La discrétisation spatiale doit être la même de part et d'autre de l'interface entre deux sous domaines. La discrétisation angulaire et énergétique est elle aussi la même pour l'ensemble des sous domaines. Dans la section 4.2 l'architecture de la réalisation sera détaillée.

2.5 Comparaison de la convergence des algorithmes PMBJ et Direct sur un colorset d'assemblages de REL

L'objectif de ce test est d'évaluer l'ordre de convergence de l'algorithme PMBJ dans le cadre d'un calcul de type REP et de le comparer à l'ordre de convergence de l'algorithme classique décrit à la section 1.4.4 (désigné par le terme *Direct* par opposition à la DD). L'algorithme de la PMBJ souffre d'un retard dans la transmission du flux à l'interface entre les sous domaines. Ce test vise à évaluer l'ampleur et l'effet de ce retard sur la convergence dans le cadre d'un calcul de Réacteur à Eau Pressurisé (REP). Ce test est important puisqu'il permet de prévoir quelle sera le potentiel de scalabilité de la méthode.

Pour cela nous proposons le calcul d'un colorset d'assemblages REP en 2D présenté en Annexe A. Le motif de base est composé de 9 assemblages UOx avec une barre insérée dans l'assemblage central. Ce motif est homogénéisé à l'échelle de la cellule (géométrie contenant un crayon, sa gaine et le modérateur). Le domaine de calcul est étendu à un colorset 6 x 6 par duplication du motif de base pour le présent test soit 10404 régions. Les calculs sont réalisés avec des sections efficaces macroscopiques à 281 groupes jusqu'à l'ordre P1 pour l'anisotropie. Nous utilisons un développement spatial linéaire du flux et 40 directions pour la quadrature angulaire. Les itérations de

puissance ne sont accélérées dans aucun calcul alors que les itérations internes sont accélérées par le CMFD dont nous parlerons par la suite (§3.3.3.d). Un sous domaine est associé à chaque assemblage pour le calcul en DDM, soit un total de 36 sous domaine.

Les critères de convergence sont fixés à 10^{-10} pour la valeur propre, la source de fission, les courants d'interface et le flux. Ces critères sont inférieurs à la précision numérique, réels en simple précision, dans le but de ne jamais atteindre la convergence. Ainsi nous pouvons comparer le comportement asymptotique de la convergence des algorithmes PMBJ et Direct.

La Figure 2.7 présente la convergence de la valeur propre pour les calculs en DDM et Direct avec un zoom autour de la convergence. Cette figure montre que le calcul en DDM converge vers la même valeur que le calcul direct. Celle-ci est comprise entre 1.23498 et 1.23496. Les oscillations à convergence sont dues au fait que les erreurs sont calculées à partir de grandeurs stockées en simple précision.

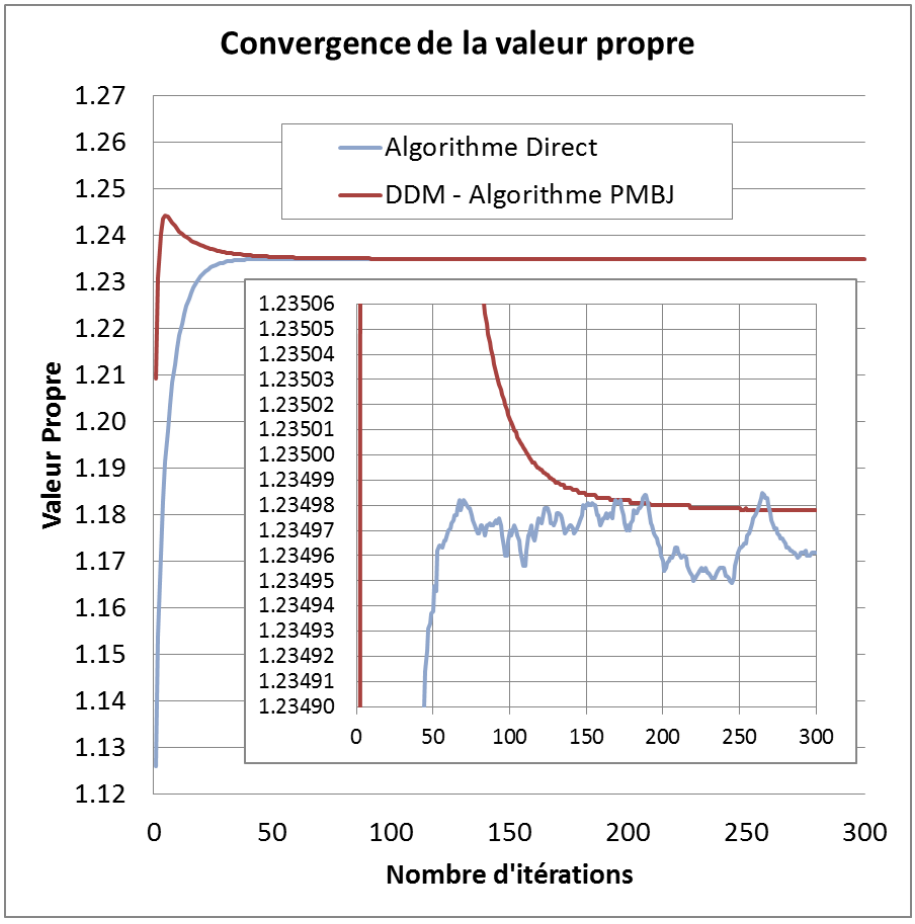


Figure 2.7. Convergence de la valeur propre avec et sans DDM.

Pour l'algorithme PMBJ on observe une oscillation de la valeur propre dans les premières itérations : elle atteint un maximum avant de décroitre pour converger, ce qui n'est pas le cas pour l'algorithme Direct. Ces oscillations sont une conséquence du retard dans la transmission des informations à travers les sous domaines. Le dernier

sous domaine, situé au nord-est du motif, ne peut prendre en compte l'influence du premier sous domaine, situé au sud-ouest, qu'à la 11^{me} itération. En effet, le flux calculé à la première itération externe dans le premier sous domaine doit être transmis à travers les 9 sous domaines qui le séparent du dernier. Avec l'algorithme PMBJ, il faut 9 itérations, les flux d'interface n'étant mis à jour qu'à la fin de l'itération externe. En généralisant sur un maillage cartésien en 3D, le nombre d'itérations nécessaire pour transmettre l'information à travers le domaine de calcul est $N_x + N_y + N_z - 2$, N_x, N_y, et N_z étant respectivement le nombre de sous domaines dans les directions x, y et z. L'augmentation du nombre de sous domaines peut ainsi augmenter l'effet d'oscillation observable dans les premières itérations. Cependant dans le cadre d'un calcul de REL, les neutrons subissent de nombreuses collisions et interagissent à quelques centimètres de leur lieu de création (de l'ordre du cm pour les neutrons thermiques et de l'ordre de 10 cm pour les neutrons rapides). Le milieu est optiquement épais et il en résulte que les sources de scattering sont importantes par rapport au transport des neutrons à travers la géométrie.

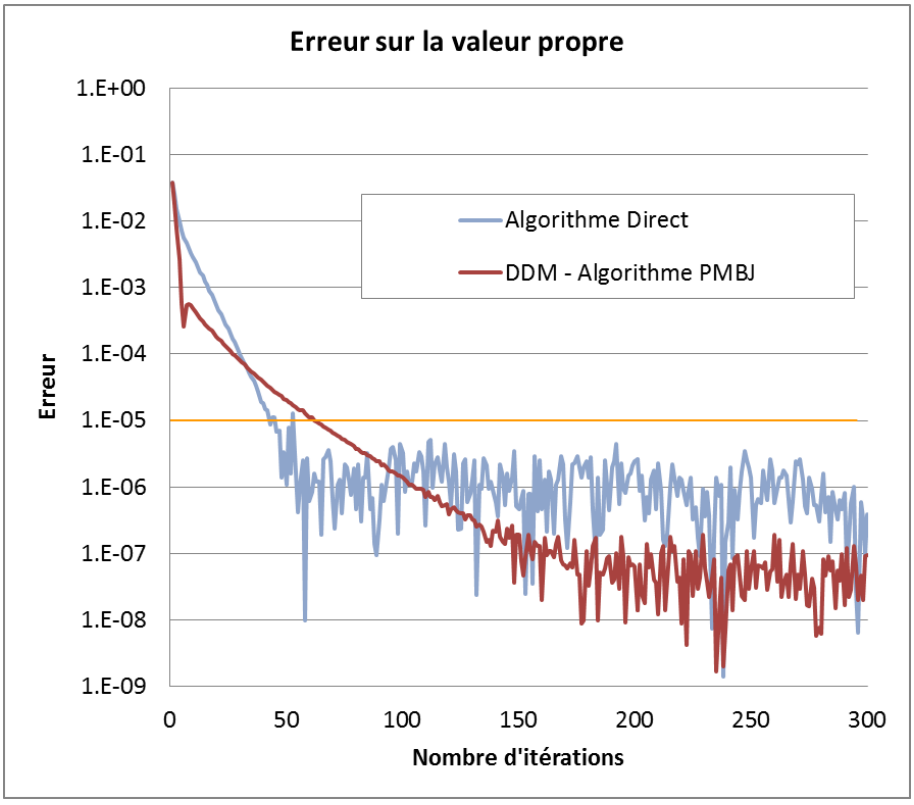


Figure 2.8. Convergence de l'erreur sur la valeur propre.

Les Figure 2.8 représente la variation de l'erreur sur la valeur propre et la Figure 2.9 celle de l'erreur sur la source de fission pour chaque itération externe. On observe que l'algorithme PMBJ du calcul DDM converge moins vite que le calcul Direct. Les critères de convergence classiques du solveur IDT sont indiqués par la ligne orange : 10^{-5} pour la valeur propre et 10^{-4} pour la source de fission. En appliquant ces critères, l'algorithme PMBJ aurait convergé en 93 itérations externes alors que le calcul direct

aurait convergé en 43 itérations externes. Cette différence du nombre d'itérations externes est alors pénalisante. En effet, le nombre de balayages transport, partie la plus coûteuse en temps de calcul, est directement lié au nombre d'itérations externes.

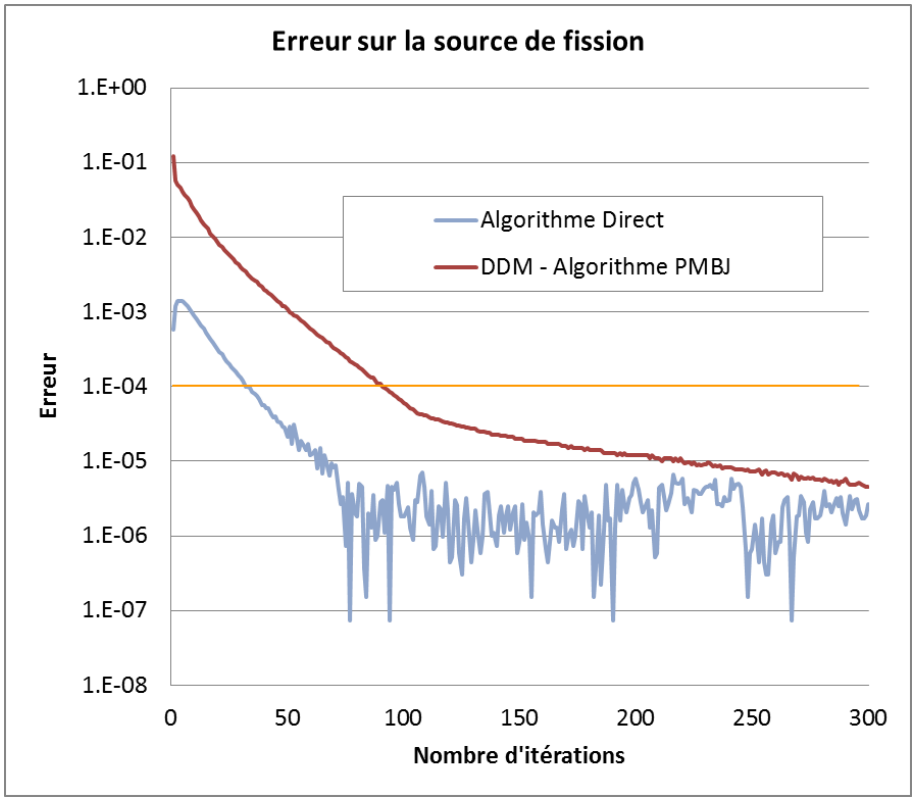


Figure 2.9. Convergence de l'erreur sur la source de fission.

Afin d'appuyer ces observations, nous avons mesuré le rayon spectral des deux méthodes. Pour cela nous nous appuyons sur le formalisme présenté dans [38]. Le rayon spectral est estimé par la norme suivante :

$$\rho = \frac{\left|I_{fis}^{(e)} - I_{fis}^{(e-1)}\right|}{\left|I_{fis}^{(e-1)} - I_{fis}^{(e-2)}\right|}, \qquad I_{fis}^{(e)} = \sum_{\forall r}\sum_{is\in r}\sum_{g}^{N_g}\left(\nu\Sigma_f\right)_{is}^{g}\phi_{r,g}^{(e)}(\boldsymbol{r}),$$

où $I_{fis}^{(e)}$ est l'intégrale de fission à l'itération externe e. Il est de 0.946 pour l'algorithme PMBJ et de 0.818 pour le Direct. Ce test a permis d'illustrer le fait que la DDM avec l'algorithme de PMBJ souffre d'un taux de convergence plus faible que le calcul Direct. Le risque étant alors d'entrainer une *fausse convergence* avec les tolérances classique d'IDT.

Nous avons réalisé ce test pour 3 tailles du colorset : 3 x 3 assemblages, 6 x 6 assemblages (cas étudié jusque-là) et 9 x 9 assemblages par réplication du motif 3 x 3. Le nombre d'itérations nécessaires pour converger peut être estimé de la manière suivante :

$$N_e \sim \frac{\ln(\varepsilon)}{\ln(\rho)},$$

où ε est la tolérance sur l'erreur entre deux itérés. Le nombre d'itérations externes estimé par cette méthode, N_e, et le rayon spectral sont présentés pour chaque tailles de colorset dans le Tableau 2.1 en prenant $\varepsilon = 10^{-5}$ pour les algorithmes Direct et PMBJ :

Tableau 2.1. Rayon spectral des algorithmes Direct et PMBJ pour différentes tailles de colorset et estimation du nombre d'itérations externes pour converger l'intégrale de fission.

Algorithme / Taille du motif	Direct		PMBJ	
	ρ	N_e	ρ	N_e
3 x 3	0.818	58	0.941	190
6 x 6	0.818	58	0.946	208
9 x 9	0.819	58	0.948	216

Le rayon spectral de l'algorithme Direct ne varie pas avec la taille du problème. En effet, la convergence de ce processus itératif dépend du rapport $c = \Sigma_{s0}/\Sigma_t$ qui ne varie pas en fonction de la taille du colorset. L'estimation du rayon spectral donne alors 58 itérations pour que l'écart entre deux itérés soit inférieur à la tolérance. Dans le cas de l'algorithme PMBJ, ceci n'est plus vrai. La DD entraine un retard dans la transmission des informations qui dépend du nombre de sous domaines. Le nombre d'itérations pour que l'écart entre deux itérés soit inférieur à la tolérance augmente alors en fonction du nombre de sous domaines, passant de 189 pour le colorset 3 x 3 à 216 pour le colorset 9 x 9.

L'erreur par rapport à la solution exacte I_{fis} lorsque l'écart entre deux itérés $I_{fis}^{(e-1)}$ et $I_{fis}^{(e)}$ est inférieur à la tolérance peut être estimée par la formule suivante :

$$\left| I_{fis} - I_{fis}^{(e)} \right| \leq \frac{\rho\varepsilon}{1-\rho}.$$

Dans le cas de l'algorithme Direct, cette erreur est de l'ordre de 4×10^{-5} alors que pour l'algorithme PMBJ, cette erreur est comprise entre 1.6×10^{-4} et 1.8×10^{-4}. L'erreur pour le dernier itéré de l'algorithme PMBJ est alors plus d'un ordre de grandeur supérieur à la tolérance. Ce phénomène est communément appelé *fausse convergence*. Lorsque nous tenterons de traiter un problème avec plus de sous domaines, le rayon spectral sera encore plus proche de l'unité, rendant très probable le fait d'être confronté au phénomène de fausse convergence.

2.6 Conclusion

Après une revue des moyens de calculs disponibles pour la simulation, nous avons présenté des méthodes modernes de parallélisation de la résolution de l'équation du

transport. Elles s'appuient généralement sur des décompositions de domaine spatiales et angulaires appliquées au problème transport monocinétique.

Dans le cadre de cette thèse, notre objectif est d'accélérer la résolution de l'équation du transport pour des calculs d'étude ou d'ingénierie visant à obtenir des résultats de « référence » rapides pour des problèmes de grande taille. L'accès au HPC n'étant pas encore généralisé, le nombre de processeurs que nous considérons est compris entre une dizaine (ordinateurs de bureau) et jusqu'à l'ordre du millier (calculateurs disponibles dans les laboratoires d'études). Notre approche a comme principal objectif de paralléliser le calcul tout en optimisant l'utilisation de la mémoire et en limitant les copies redondantes de données. L'objectif de portabilité du code engendre le fait que nous ne pouvons pas optimiser l'implémentation sur une architecture de calcul particulière. De ce fait l'overhead dû au parallélisme doit être minimisé.

Partant de ce constat, nous proposons un nouvel algorithme basé sur la décomposition de domaine spatiale. Cet algorithme a pour ambition de minimiser le nombre de synchronisations ainsi que le nombre de communications entre les sous domaines. Pour cela, nous appliquons la décomposition de domaine au problème multigroupe. Dans chaque sous domaine le balayage multigroupe complet est réalisé avant de communiquer les flux d'interface et de mettre à jour la source de fission. Cet algorithme est un algorithme de Jacobi Parallèle par Bloc Multigroupe (PMBJ). Une seule communication est alors requise par itération externe : après le balayage multigroupe dans chaque sous domaine, le flux d'interface multigroupe est communiqué.

L'objectif que nous visons est d'atteindre une scalabilité idéale, c'est-à-dire obtenir des propriétés de convergence et des temps de calculs similaires entre l'algorithme PMBJ et l'algorithme Direct, que nous considérons comme référence. Comme nous l'avons fait remarquer a priori, mais aussi par une illustration donnée par calcul d'un colorset d'assemblage de type REL, l'algorithme PMBJ a de moins bonnes propriétés de convergences que l'algorithme Direct. L'augmentation du nombre d'itérations nécessaire pour converger ne permet pas d'atteindre l'objectif de scalabilité idéale. De plus la dépendance du rayon spectral par rapport au nombre de sous domaine est très préjudiciable puisque nous visons à traiter des problèmes de grandes tailles. Finalement des phénomènes de fausse convergence peuvent apparaitre avec l'algorithme PMBJ.

Néanmoins, dans la section suivante, nous présenterons les méthodes qui nous ont permis d'améliorer les propriétés de convergence de l'algorithme PMBJ. Nous verrons ainsi qu'il est possible d'accélérer l'algorithme de telle sorte qu'il soit aussi performant que l'algorithme Direct.

Chapitre 3. Accélérations de l'algorithme PMBJ

3.1 Introduction

Nous avons vu dans le chapitre précédent que le taux de convergence de l'algorithme PMBJ est plus faible que celui de l'algorithme Direct. De ce fait, l'espoir d'obtenir une scalabilité idéale est vain avec cet algorithme dû à l'augmentation du nombre d'itérations nécessaires pour atteindre la convergence. L'algorithme PMBJ entraine un retard dans la transmission de l'information à l'interface entre les sous domaines. L'objectif des méthodes présentées dans cette section est de modifier l'algorithme de manière à améliorer la transmission de l'information entre les sous domaines.

Nous chercherons ainsi à améliorer le taux de convergence par plusieurs approches. Tout d'abord dans la section 3.2, nous remettrons en question l'algorithme PMBJ. Nous proposerons des algorithmes de type Gauss-Seidel Parallèle par Bloc Multigroupe (PMBG-S, acronyme de l'anglais Parallel Multigroup-Block Gauss-Seidel) permettant de profiter des flux d'interface les plus récents avant le calcul de chaque sous domaine. Ensuite, nous présenterons une méthode d'accélération non-linéaire par la diffusion, le Coarse Mesh Finite Difference (CMFD). Cette méthode sera présentée à la section 3.3 et son application au DDM sera détaillée. Enfin, à la section 3.4 nous modifierons la méthode de calcul des sous domaines. Localement, plusieurs itérations sur la source de fission seront réalisées avant de recalculer la source de fission et la valeur propre globale.

3.2 Résolution par des algorithmes de type Gauss Seidel par Bloc Multigroupe

L'Algorithme 2.2 proposé jusqu'à présent est un algorithme de Jacobi : le flux d'interface entre les deux sous domaines est le flux calculé à l'itération externe précédente. L'ordre de résolution des sous domaines n'a pas d'importance puisque qu'ils sont tous indépendants, permettant d'accéder à une parallélisation maximale. Dans cette section nous mettons de côté l'optimisation de la parallélisation pour mettre en place des algorithmes plus performants, tout en gardant les blocs multigroupes dans les sous domaines.

Nous proposons alors d'améliorer la convergence de la DDM par des algorithmes PMBG-S sur les flux d'interfaces comme décrit dans l'Algorithme 3.1 ci-dessous.

Algorithme 3.1. Algorithme de Schwarz multiplicatif appliqué à l'équation du transport pour le cas à 2 sous domaines.

$Jusqu'$à $convergence, répéter$:

$$résoudre : \begin{cases} A_1\psi_1^{(e+1)} = Q_1 + \tilde{B}_{1\Gamma}\psi_{\Gamma 1,-}^{(e)}, & dans\ D_1, \\ \psi_{\Gamma 1,+}^{(e+1)} = \widetilde{D}_{\Gamma 1}\psi_1^{(e+1)} + \tilde{C}_{\Gamma,1}\psi_{\Gamma 1,-}^{(e)}, & sur\ \Gamma. \end{cases}$$

$$nouveau\ flux\ entrant : \quad \psi_{\Gamma 2,-}^{(e+1)} = \psi_{\Gamma 1,+}^{(e+1)}, \ sur\ \Gamma,$$

$$résoudre : \begin{cases} A_2\psi_2^{(e+1)} = Q_2 + \tilde{B}_{2\Gamma}\psi_{\Gamma 2,-}^{(e+1)}, & dans\ D_2, \\ \psi_{\Gamma 2,+}^{(e+1)} = \widetilde{D}_{\Gamma 2}\psi_2^{(e+1)} + \tilde{C}_{\Gamma,2}\psi_{\Gamma 2,-}^{(e+1)}, & sur\ \Gamma. \end{cases}$$

$$nouveau\ flux\ entrant : \quad \psi_{\Gamma 1,-}^{(e+1)} = \psi_{\Gamma 2,+}^{(e+1)}, \ sur\ \Gamma.$$

Cet algorithme permet de profiter du dernier flux de surface calculé pour le transmettre au sous domaine voisin. En généralisant à un nombre quelconque de sous domaines, il s'agit d'assigner à chaque sous domaine un numéro d'ordre. Selon la manière de distribuer ces numéros d'ordre aux sous domaines, la solution est propagée de manière différente dans le domaine de calcul.

L'inconvénient est que la parallélisation ne pourra plus être totale. Pour définir le niveau de parallélisme d'un algorithme nous introduisons la notion de *cycle*. D'une manière générale, les sous domaines sont parallélisable par paquet. Le nombre cycles est défini par le nombre de ces paquets. Plus le nombre de cycles est petit, plus l'algorithme est parallélisable efficacement. Avec l'algorithme PMBJ, tous les sous domaines peuvent être résolus en même temps, c'est-à-dire en un seul cycle. Le niveau de parallélisme est alors optimal. Les algorithmes de type PMBG-S sont contraints par l'ordre de résolution des sous domaines. Cette contrainte implique de ne plus pouvoir résoudre tous les sous domaines à la fois. Ils seront alors parallélisés en plusieurs cycles.

IMPLEMENTATION DES ALGORITHMES PMBG-S

L'Algorithme 2.3 est remplacé par l'Algorithme 3.2 présenté ci-après. Les modifications sont indiquées en rouge. Le changement intervient dans l'échange des flux d'interface et la boucle de résolution des sous domaines. Ces derniers sont désormais résolus dans un ordre défini à l'avance. Cet ordre conditionne la convergence de l'algorithme puisqu'il définit la manière dont la solution est propagée dans les sous domaines.

Concrètement, à chaque sous domaine est associé un numéro d'ordre α. Le calcul des sous domaines est ordonné, allant de 1 à A. Ceci est indiqué par le remplacement de « *Pour chaque* α ∈ A » par « *Pour* α = 1, … , A » dans l'Algorithme 3.2. La mise à jour des flux d'interfaces est réalisée juste avant le calcul de chaque sous domaine :

- Le flux provenant de sous domaines de numéro d'ordre inférieur est le flux calculé à l'itération courante.
- Le flux provenant de sous domaines de numéros d'ordre supérieurs est celui de l'itération précédente.

Dans l'Algorithme 3.2, cette distinction est faite par l'introduction de l'indice d'itération e'. Pour le sous domaine α on a alors :

$$\forall\, \beta \neq \alpha :\ e'_\alpha(\beta) = \begin{cases} e+1, & \beta < \alpha\,, \\ e, & \beta > \alpha\,, \end{cases}$$

où β est le numéro d'ordre d'un sous domaine voisin. L'algorithme n'est pas modifié par ailleurs.

Algorithme 3.2. Algorithme de Gauss-Seidel par Bloc Multigroupe.

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha\ \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux d'interface et calcul transport des sous domaines

Pour $\alpha = 1, \dots, A$, ***faire***

$$\forall\, \beta \neq \alpha :\ e'(\beta) = \begin{cases} e+1, & \beta < \alpha\,, \\ e, & \beta > \alpha\,. \end{cases}$$

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e'(\beta))}(x), & \forall\, \beta \neq \alpha :\ x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e'(\beta))}(T^{-1}x), & \forall\, \beta \neq \alpha :\ (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

$$\begin{cases} (L-H)_\alpha \psi_\alpha{}^{(e+1)}(x) = Q_\alpha{}^{(e)}(x), & x \in X_\alpha, \\ \psi_\alpha{}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Calcul de l'intégrale de fission

Pour chaque $\alpha \in A$, ***faire***

$$\boldsymbol{F}_\alpha^{(e+1)}(\boldsymbol{r}) = \left\{ F_{is}(\boldsymbol{r}) = \sum_g^{N_g} \left(\nu\Sigma_f\right)_{is}^g(\boldsymbol{r}) \phi_g^{(e+1)}(\boldsymbol{r})\ ,\qquad \boldsymbol{r} \in D_\alpha \right\}.$$

fin

Mise à jour de la valeur propre globale

$$\lambda^{(e+1)} = \lambda^{(e)} \frac{\sum_{\alpha\in A} \langle \boldsymbol{F}_\alpha{}^{(e)}, \boldsymbol{F}_\alpha{}^{(e+1)} \rangle}{\sum_{\alpha\in A} \langle \boldsymbol{F}_\alpha{}^{(e)}, \boldsymbol{F}_\alpha{}^{(e)} \rangle}.$$

Normalisation de la source de fission

Pour chaque $\alpha \in A$, ***faire***

$$Q_\alpha{}^{(e+1)}(x) = \frac{(F\psi)_\alpha^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

fin

Dans cette section nous verrons plusieurs manières d'ordonner le calcul des sous domaines dans le but d'accélérer la convergence de la DDM par bloc multigroupe. Pour cela nous nous appuierons sur l'exemple d'un domaine en 2D divisé en 36 sous domaines tel que présenté Figure 3.1.

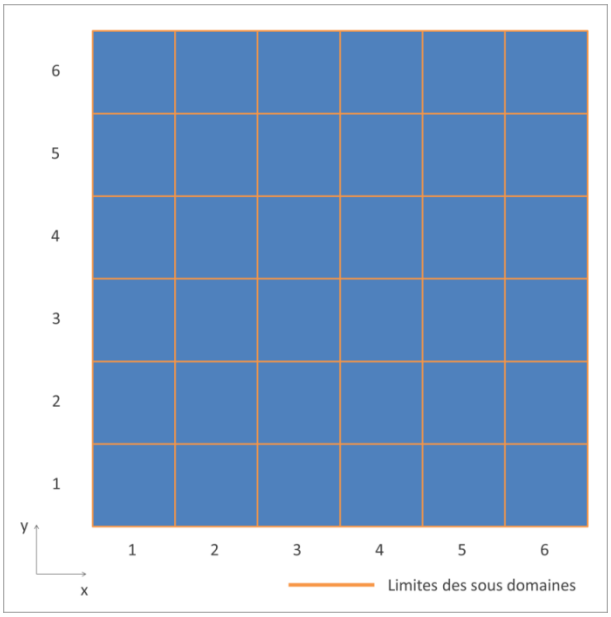


Figure 3.1. Exemple de découpage d'un domaine 2D en 36 sous domaines.

Les sous domaines sont repérés selon le double indice $\{i;j\}$ correspondant aux numéros de la maille selon les directions x et y de la géométrie. On a $1 \leq i \leq N_x$ et $1 \leq j \leq N_y$, N_x et N_y étant respectivement le nombre de sous domaines le long de l'axe x et de l'axe y.

3.2.1 Résolution selon la numérotation cartésienne ou par fronts de propagation (KBA)

Le premier algorithme que nous présentons est un algorithme de Gauss-Seidel dans sa version la plus intuitive. Le numéro d'ordre α de chaque sous domaine de cette méthode est défini par :

$$\alpha = i + (j-1) \times N_x,$$

qui correspond à la numérotation cartésienne. Cet ordre est présenté Figure 3.2 ci-après.

La contrainte majeure de cette méthode concerne la parallélisation du calcul. La résolution des sous domaines est complètement séquentielle et la parallélisation n'est plus possible.

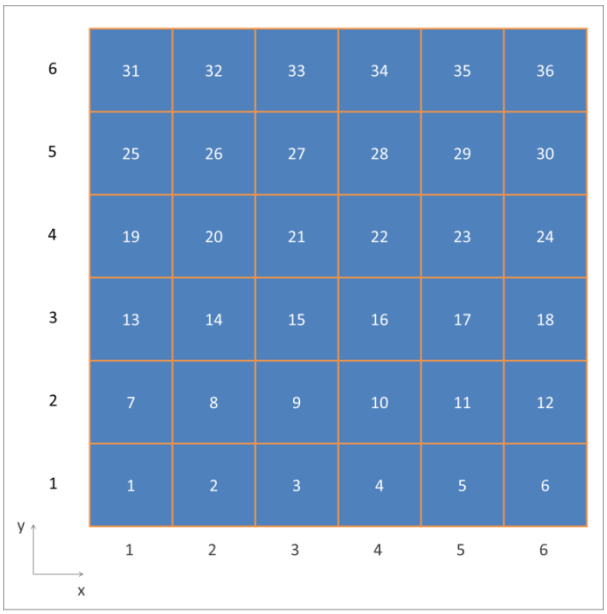


Figure 3.2. Ordre de résolution selon la numérotation cartésienne ou KBA séquentiel.

Cependant nous pouvons noter que le sous domaine dont le double indice est $\{i; j\}$ a pour condition de flux entrant :

- le flux calculé lors de l'itération courante pour ses interfaces avec les sous domaines $\{i-1; j\}$ et $\{i; j-1\}$,
- le flux entrant calculé lors de l'itération précédente pour ses interfaces avec les sous domaines $\{i+1; j\}$ et $\{i; j+1\}$.

Ainsi, en respectant cette propriété, il est possible de propager la solution par front : lorsque le sous domaine $\{1; 1\}$ a été calculé, le $\{2; 1\}$ et le $\{1; 2\}$ peuvent l'être en parallèle, et ainsi de suite. C'est-à-dire selon les indices $i + j$ croissants.

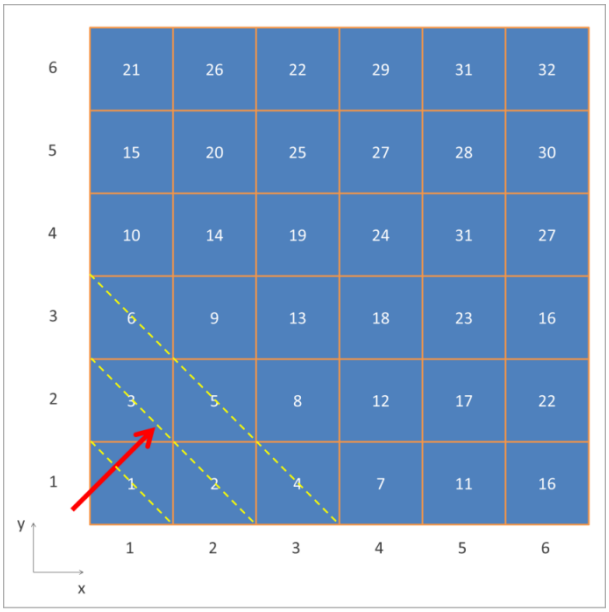


Figure 3.3. Ordre de résolution selon un front de propagation (KBA).

Cet algorithme de balayage n'est autre que l'algorithme de balayage KBA [28]. Le nombre de cycles pour balayer l'intégralité des sous domaines est de $N_x + N_y - 1$ par rapport au $N_x \times N_y$ tâches à effectuer. Dans le cas d'un algorithme par bloc multigroupe, la propagation de la solution de cette manière ne permet pas d'inverser directement l'opérateur de transport, contrairement aux méthodes de décomposition de domaine appliquées au balayage transport dans les itérations internes. De ce fait, on peut se permettre de commencer l'algorithme KBA par tous les coins du domaine de calcul simultanément (Figure 3.4). De cette manière, on a toujours deux interfaces avec les nouveaux flux et deux interfaces avec les anciens flux. Le niveau de parallélisme de cet algorithme KBA par fronts multiples est alors augmenté : le nombre de cycles est $\frac{N_x+N_y}{2} - 1$ soit de l'ordre de la moitié de l'algorithme KBA standard.

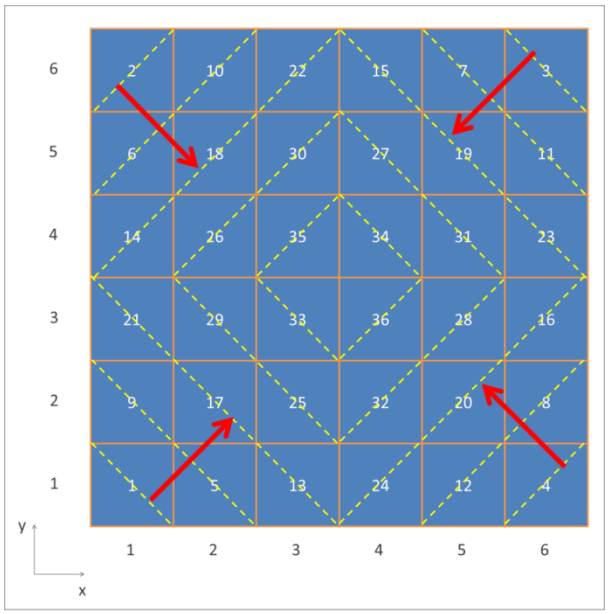


Figure 3.4. Ordre de résolution suivant plusieurs fronts de propagation.

Dans l'implémentation réalisée, les fronts de propagation sont arrêtés au centre du domaine de manière à ne calculer qu'une seule fois chaque sous domaine. Il eut été habile de propager les fronts de propagation à travers le domaine entier. En effet, au paragraphe 2.5, nous avons fait remarquer qu'il fallait $N_x + N_y - 1$ itérations externes pour propager la solution à travers le domaine. En prolongeant les fronts de propagation à travers le domaine entier sur les 4 directions, une seule itération externe est requise au prix de calculer 4 fois chaque sous domaine. Nous n'avons pas testé cette stratégie.

3.2.2 Schéma « Red-Black »

Dans la section précédente, nous avons vu qu'il n'était pas aisé de paralléliser efficacement un algorithme de Gauss-Seidel. Dans le but de pallier à ce défaut, nous proposons un schéma Red-Black (RB). Pour cela nous divisons les sous domaines en deux parties A_R et A_B de telle sorte que :

$$D = \left[\cup_{\alpha \in A_R} D_\alpha\right] \cup \left[\cup_{\alpha \in A_B} D_\alpha\right], \tag{3.1}$$

et que chaque élément $D_\alpha, \alpha \in A_R$ ne soit adjacent d'aucun élément $D_\alpha, \alpha \in A_B$. La Figure 3.5 illustre une telle partition : les cases rouges représentent les sous domaines appartenant à A_R et les cases noires ceux appartenant à A_B.

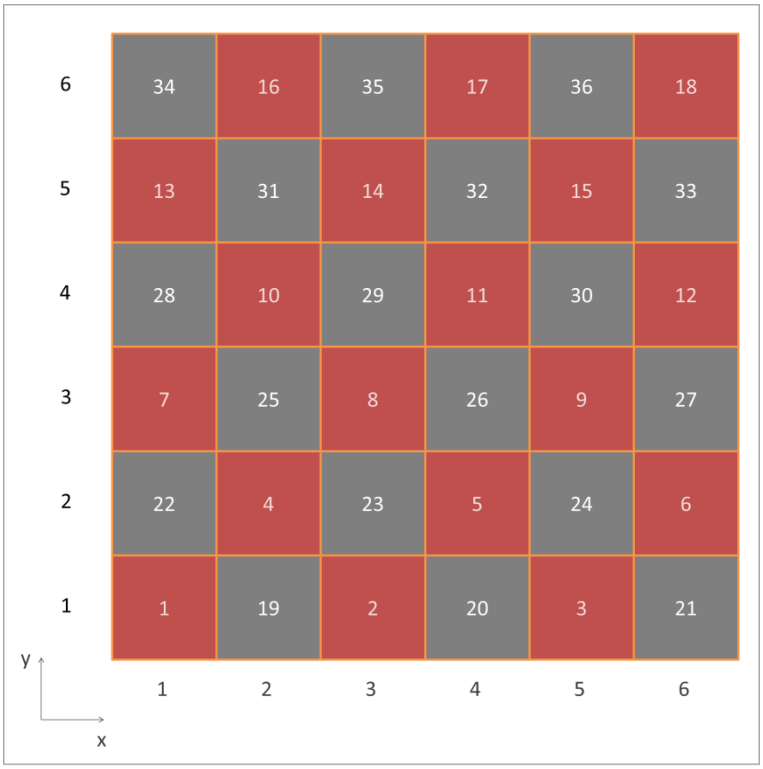


Figure 3.5. Balayage par le schéma « Red and Black ».

Cet algorithme permet de réduire le nombre de cycles à 2. L'avantage de cette méthode est que le nombre de cycles est indépendant du nombre de sous domaines. L'Algorithme 3.3 présente la résolution du DDM suivant le schéma RB.

Les sous domaines appartenant au groupe A_R sont résolus en premier. Les flux aux interfaces sont alors issus de l'itération externe précédente. Dans un second temps les sous domaines du groupe A_B sont résolus, profitant des flux d'interfaces de leurs voisins calculés au cours de l'itération externe courante. L'algorithme Red-Black est de type Jacobi pour la partition A_R et de type Gauss-Seidel pour la partition A_B.

Algorithme 3.3. Algorithme de résolution par le schéma « Red-Black ».

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha \; \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux d'interface et calcul transport des sous domaines « red »

Pour $\alpha \in A_R$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall \beta \neq \alpha : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall \beta \neq \alpha : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+1)}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Mise à jour des flux d'interface et calcul transport des sous domaines « black »

Pour $\alpha \in A_B$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e+1)}(x), & \forall \beta \neq \alpha : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e+1)}(T^{-1}x), & \forall \beta \neq \alpha : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+1)}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Calcul de l'intégrale de fission

Pour chaque $\alpha \in A$, ***faire***

$$\boldsymbol{F}_\alpha^{(e+1)}(\boldsymbol{r}) = \left\{ F_{is}(\boldsymbol{r}) = \sum_g^{N_g} (\nu\Sigma_f)_{is}^g(\boldsymbol{r}) \phi_g^{(e+1)}(\boldsymbol{r}) \;\; , \qquad \boldsymbol{r} \in D_\alpha \right\}.$$

fin

Mise à jour de la valeur propre globale

$$\lambda^{(e+1)} = \lambda^{(e)} \frac{\sum_{\alpha \in A} \langle {\boldsymbol{F}_\alpha}^{(e)}, {\boldsymbol{F}_\alpha}^{(e+1)} \rangle}{\sum_{\alpha \in A} \langle {\boldsymbol{F}_\alpha}^{(e)}, {\boldsymbol{F}_\alpha}^{(e)} \rangle}.$$

Normalisation de la source de fission

Pour chaque $\alpha \in A$, ***faire***

$${Q_\alpha}^{(e+1)}(x) = \frac{(F\psi)_\alpha^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

fin

3.2.3 Comparaison des algorithmes de type Gauss-Seidel par bloc multigroupe avec l'algorithme Direct sur un colorset d'assemblages de REP

L'objet de ce test est d'évaluer l'ordre de convergence des différents algorithmes PMBG-S dans le cadre d'un calcul de type REP par rapport à l'ordre de convergence de l'algorithme Direct et de l'algorithme PMBJ. Nous rappelons que l'on cherche à obtenir une scalabilité idéale avec l'algorithme parallélisé. Ceci n'est réalisable qu'à la condition d'avoir un rayon spectral similaire à l'algorithme Direct.

Nous reprenons ainsi le test présenté à la section 2.5 dans lequel nous avons comparé l'algorithme PMBJ avec l'algorithme Direct : un colorset en 2D composé de 36 sous domaines de la taille d'un assemblage de type REP, soit 10404 régions homogènes. Les calculs sont réalisés avec des sections efficaces macroscopiques à 281 groupes jusqu'à l'ordre P1 pour l'anisotropie. Les critères de convergence sont toujours fixés à 10^{-10} pour la valeur propre, la source de fission, les courants d'interface et le flux, dans le but de comparer le comportement asymptotique de la convergence des différents algorithmes.

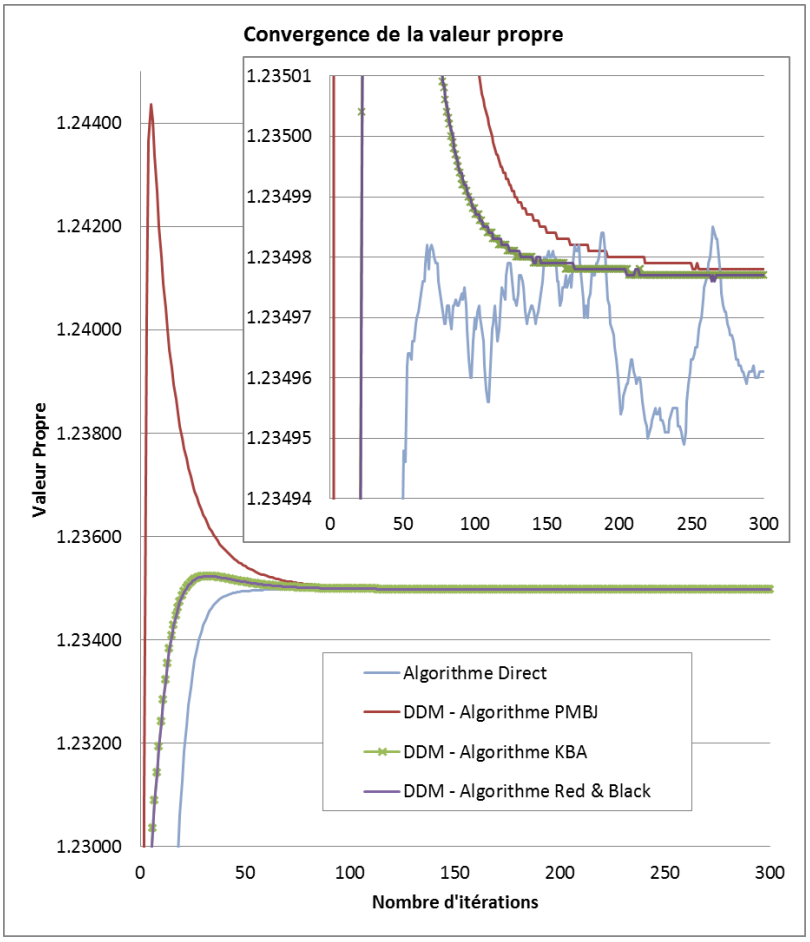


Figure 3.6. Convergence de la valeur propre pour les différents algorithmes de balayage des sous domaines.

La Figure 3.6 présente la convergence de la valeur propre pour deux des algorithmes de type PMBG-S (KBA et Red-Black) présentés ci-dessus, en plus des algorithmes

Directs et PMBJ déjà étudiés à la section 2.5. Un zoom est réalisé autour de la valeur de convergence. Cette figure montre que les algorithmes convergent vers la même valeur propre. Pour les algorithmes KBA et Red-Black, les courbes sont confondues. Par ailleurs, l'amplitude de l'oscillation observée dans les premières itérations pour l'algorithme PMBJ est nettement atténuée, mais toujours présente, pour les algorithmes PMBG-S.

La Figure 3.7 et la Figure 3.8 représentent respectivement l'erreur sur la valeur propre et la source de fission pour chaque itération externe. L'erreur sur la valeur propre décroit nettement plus rapidement pour les algorithmes PMBG-S alors que l'erreur sur la source de fission est très similaire pour les algorithmes PMBG-S et PMBJ.

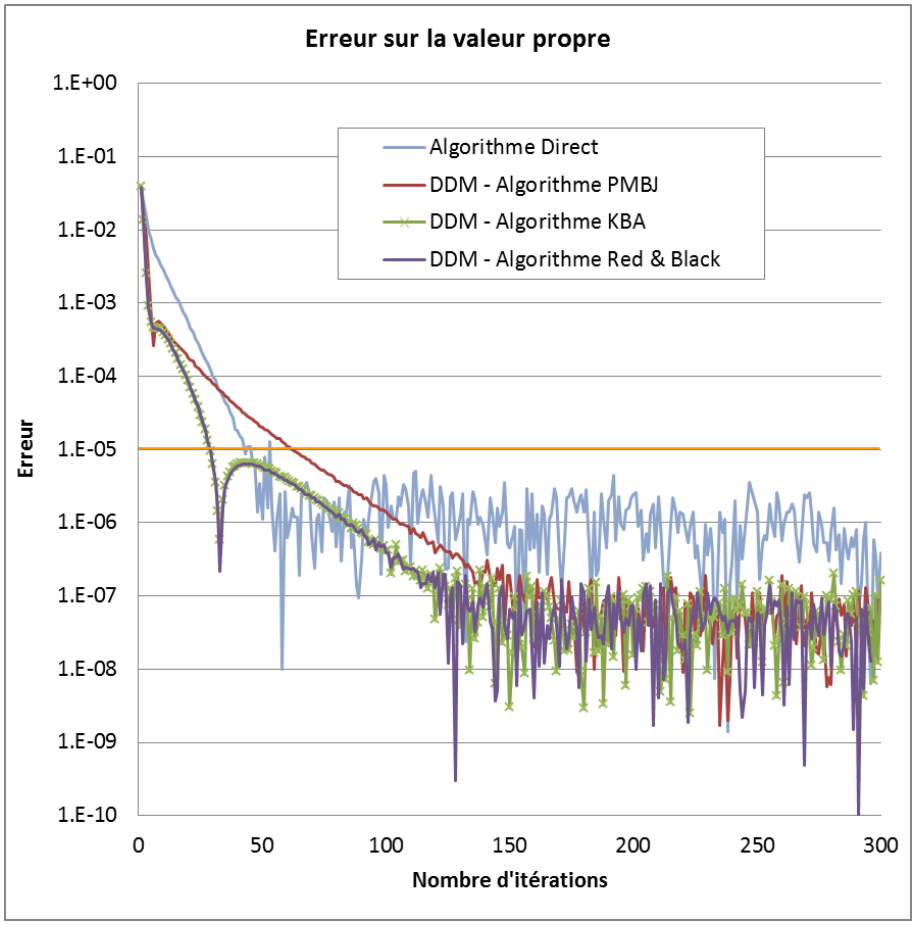


Figure 3.7. Convergence de l'erreur sur la valeur propre pour les différents algorithmes de balayage des sous domaines.

Les critères de convergence classiques pour le solveur IDT sont de 10^{-5} pour la valeur propre et 10^{-4} pour la source de fission. En appliquant ces critères, les algorithmes PMG-S auraient convergés en 81 itérations alors que l'algorithme PMBJ aurait eu besoin de 93 itérations. Les algorithmes PMBG-S ont permis d'accélérer la convergence, cependant le nombre d'itérations nécessaires est nettement supérieur à celui de l'algorithme Direct (43 itérations).

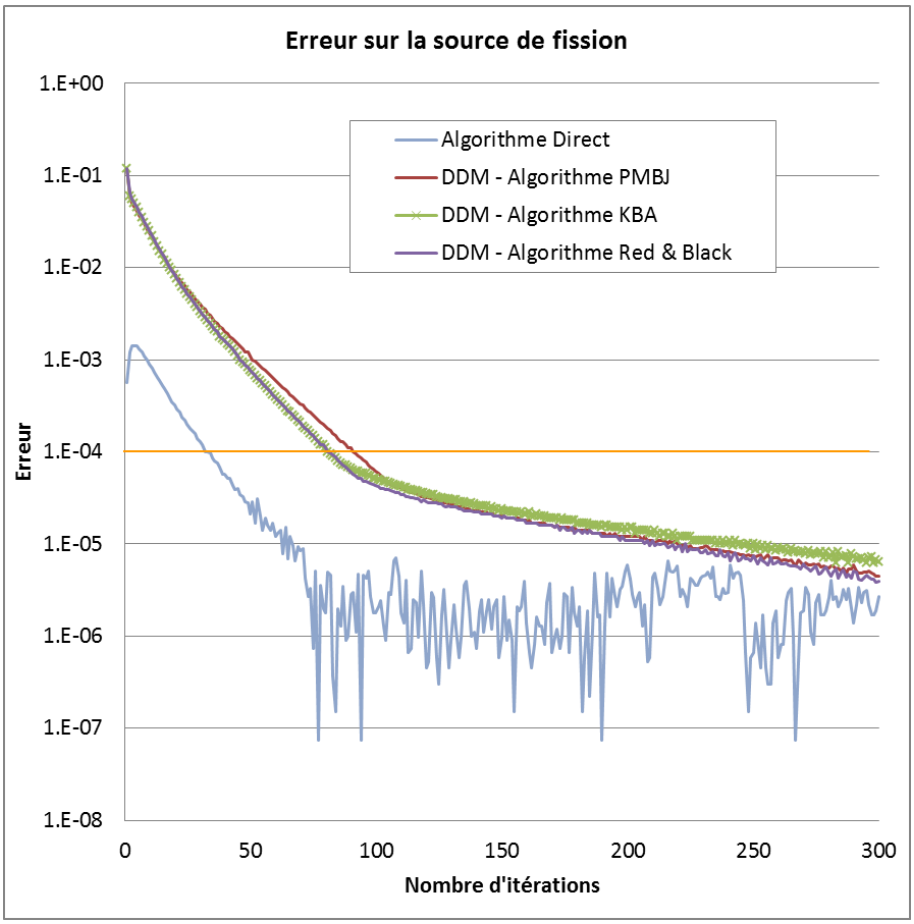


Figure 3.8. Convergence de l'erreur sur la source de fission pour les différents algorithmes de balayage des sous domaines.

Les critères de convergence classiques pour le solveur IDT sont de 10^{-5} pour la valeur propre et 10^{-4} pour la source de fission. En appliquant ces critères, les algorithmes PMG-S auraient convergés en 81 itérations alors que l'algorithme PMBJ aurait eu besoin de 93 itérations. Les algorithmes PMBG-S ont permis d'accélérer la convergence, cependant le nombre d'itérations nécessaires est nettement supérieur à celui de l'algorithme Direct (43 itérations). Comme le montre le Tableau 3.1, le rayon spectral n'est que peu amélioré.

Tableau 3.1. Rayon spectral des algorithmes Direct, PMBJ et PMBG-S.

Algorithme	Direct	PMBJ	KBA	Red-Black
Rayon spectral	0.818	0.946	0.941	0.941

Le rayon spectral est presque identique pour tous les algorithmes par bloc multigroupe présentés ici. Ce rayon est supérieur à celui de l'algorithme Direct.

Nous avons réalisé ce test pour 3 tailles du colorset : 3 x 3 assemblages, 6 x 6 assemblages (cas étudié jusque-là) et 9 x 9 assemblages par réplication du motif 3 x 3 de la même manière qu'au paragraphe 2.5. Le nombre d'itérations externes estimé, N_e, et le rayon spectral, ρ, sont présentés pour chaque taille de colorset et pour les

algorithmes Direct, PMBJ et PMBG-S dans le Tableau 3.2 en prenant comme tolérance $\varepsilon = 10^{-5}$:

Tableau 3.2. Rayon spectral des algorithmes Direct, PMBJ et PMBG-S pour différentes tailles de colorset et estimation du nombre d'itérations externes pour converger l'intégrale de fission.

	Rayon spectral							
Algorithme / Taille du motif	Direct		PMBJ		KBA		Red-Black	
	ρ	N_e	ρ	N_e	ρ	N_e	ρ	N_e
3 x 3	0.818	58	0.941	190	0.939	183	0.938	180
6 x 6	0.818	58	0.946	208	0.941	190	0.941	190
9 x 9	0.819	58	0.948	216	0.943	197	0.943	197

Le formalisme utilisé ici pour estimer le rayon spectral et le nombre d'itérations est présenté page 58 (section 2.5). Les résultats du paragraphe 2.5 ont été reportés dans ce tableau. Tout d'abord, on peut noter que les algorithmes KBA et Red-black ont des rayons spectraux très similaires. En effet, hormis le motif le plus petit, nous n'avons pas mesuré de différence. Par ailleurs, les algorithmes PMBG-S (KBA et Red-Black) n'améliorent que très peu le rayon spectral par rapport à l'algorithme PMBJ. Le rayon spectral de l'algorithme Direct est alors toujours nettement inférieur aux algorithmes s'appuyant sur la DDM. Il ne nous est pas possible d'atteindre une parallélisation idéale en utilisant uniquement ces méthodes. Nous n'avons donc pas approfondi notre recherche d'algorithmes de type PMBG-S et préféré travailler sur une autre méthode d'accélération. Celle-ci est présentée dans la section suivante.

3.3 Accélération par le CMFD

Le Coarse Mesh Finite Difference (CMFD) a été originellement développé comme technique d'accélération pour les méthodes de diffusion nodales utilisées pour les calculs de cœurs [39]. Néanmoins, le concept fondamental s'applique aussi bien aux méthodes de transport. Le choix de cette méthode d'accélération a été fait pour plusieurs raisons. Tout d'abord le CMFD a été utilisé avec succès pour accélérer les calculs transport [3, 34, 40]. Elle permet non seulement d'accélérer les calculs à sources mais aussi les problèmes à valeur propre. Par ailleurs, dans le cadre de la DDM, cette méthode permet d'accélérer les sources mais aussi les flux d'interfaces entre sous domaines. Un autre avantage de cette méthode d'accélération est qu'elle est peu coûteuse, les coefficients pouvant être calculés à la volée. De plus, sous certaines conditions, la consommation de mémoire est réduite. Le maillage adaptatif que nous avons développé (§3.3.3.a) permet de trouver une solution optimale pour l'utilisateur. Finalement, cette méthode peut être aisément utilisée pour un autre solveur qu'IDT.

3.3.1 Principe de l'accélération non linéaire par la diffusion

Nous cherchons à accélérer la résolution de l'équation du transport pour un problème stationnaire posé sur un domaine D :

$$\begin{cases} (L-H)\psi(x) = \mathrm{Q}(x), & x \in X, \\ \psi(x) = \psi^{-}(x), & x \in \Gamma^{-}. \end{cases} \tag{3.2}$$

Les notations sont les mêmes que celles de la section 2.4 : $\psi(x)$ est le flux à l'intérieur du domaine et $\psi^{\pm}(x)$ est le flux à sa surface définie pour $x \in \Gamma^{\pm}$. $(L-H)$ est l'opérateur de transport multigroupe et $\mathrm{Q}(x)$ est la source de fission normalisée. L'espace des phases est défini par :

$$\begin{aligned} X &\equiv (\boldsymbol{r} \in D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2, E \in \mathbb{R}^+), \\ \Gamma^{\pm} &\equiv (\boldsymbol{r} \in \partial D, \boldsymbol{\Omega} \in S^2 : \boldsymbol{\Omega} \cdot \boldsymbol{n}_{+}(\boldsymbol{r}) \gtrless 0, E \in \mathbb{R}^+). \end{aligned} \tag{3.3}$$

La notation employée sera « analytique » pour alléger la lecture. En réalité on sous-entendra une discrétisation pour tout l'espace des phases, de sorte que chaque intégrale est remplaçable par une somme discrète.

L'opérateur de diffusion est notoirement un excellent pré-conditionneur de l'opérateur transport. A la différence des méthodes synthétiques linéaires, la construction de l'opérateur de diffusion CMFD est basée sur la conservation du bilan neutronique sur un espace des phases grossier. Ceci engendre la non-linéarité de l'opérateur à travers la pondération des sections efficaces par le flux et la définition de facteurs d'équivalence pour la conservation des courants.

Par soucis de simplicité, on définit l'espace des phases grossier par :

$$Y \equiv (\cup y \ : \cap y = \emptyset) \, ,$$

où la variable y est en réalité une intégrale sur la sphère unité, sur une maille grossière de taille V_R et sur un macro intervalle d'énergie ΔE_G. On a alors :

$$y \equiv \int_y dx = \int_{V_R} d\boldsymbol{r} \int_{4\pi} d\boldsymbol{\Omega} \int_{\Delta E_G} dE \ , \tag{3.4}$$

tel que, pour n'importe quelle fonction f, on a : $f(y) \equiv \int_y dx\, f(x)$.

En notant avec l'indice $(e+1/2)$ toutes quantités liées au balayage transport et avec $(e+1)$ les quantités liées à l'opérateur grossier et au flux accéléré, chaque itération sera composée de 4 étapes :

1 - **Balayage transport multigroupe :**

$$(L-H)\psi^{(e+1/2)}(x) = Q^{(e)}(x)\,, \qquad x \in X. \tag{3.5}$$

2 - **Homogénéisation-condensation et équivalence :**

Après l'intégration du courant et du flux sur l'espace grossier :

$$\begin{aligned} \boldsymbol{J}^{(e+1/2)}(y) &= \int_y dx\, \boldsymbol{\Omega}\, \psi^{(e+1/2)}(x)\ , \\ \phi^{(e+1/2)}(y) &= \int_y dx\, \psi^{(e+1/2)}(x) \quad , \end{aligned} \tag{3.6}$$

les constantes de l'opérateur synthétique sont calculées de la manière suivante :

$$\begin{aligned} {\Sigma_t}^{(e+1/2)}(y) &= \int_y dx\, \Sigma_t(x) \frac{\psi^{(e+1/2)}(x)}{\phi^{(e+1/2)}(y)}\ , \\ {\Sigma_s}^{(e+1/2)}(y' \to y) &= \int_y dx \int_{y'} dx'\, \Sigma_s(x' \to x) \frac{\psi^{(e+1/2)}(x')}{\phi^{(e+1/2)}(y')}\ , \\ \left(\chi \upsilon \Sigma_f\right)^{(e+1/2)}(y' \to y) &= \int_y dx \sum_i \chi_i(x) \int_{y'} dx'\, (\upsilon\Sigma_f)_i(x') \frac{\psi^{(e+1/2)}(x')}{\phi^{(e+1/2)}(y')}\ , \end{aligned} \tag{3.7}$$

où l'indice i est l'indice des isotopes fissiles. Notons qu'il est nécessaire de conserver la matrice de fission entière pour prendre en compte un spectre de fission dépendant des isotopes. Les paramètres d'équivalence et les coefficients de diffusion sont donnés par :

$$\widetilde{\boldsymbol{D}}^{(e+1/2)}(y) = \left[\frac{\boldsymbol{J}(y) + D\boldsymbol{\nabla}\phi(y)}{\phi(y)}\right]^{(e+1/2)} \ , \tag{3.8}$$

$$D^{(e+1/2)} = \left[\frac{1}{3(\,\Sigma_t\, - \Sigma_{s1})}\right]^{(e+1/2)} . \tag{3.9}$$

3- **Solution de l'opérateur synthétique :**

En considérant les mêmes conditions aux limites que celles du problème transport, on résout l'équation pour le flux grossier :

$$\boldsymbol{\nabla} \cdot \boldsymbol{J}^{(e+1)}(y) + (\Sigma_t - H)^{(e+1/2)}\phi^{(e+1)}(y) = \frac{1}{\lambda^{(e+1)}} F^{(e+1/2)}\phi^{(e+1)}(y)\,. \qquad (3.10)$$

où le courant est donné par :

$$\boldsymbol{J}^{(e+1)}(y) = -D^{(e+1/2)}\boldsymbol{\nabla}\phi^{(e+1)}(y) + \widetilde{\boldsymbol{D}}^{(e+1/2)}\phi^{(e+1)}(y)\,, \qquad (3.11)$$

et les opérateurs de scattering H et de fission F sont définis par :

$$\begin{cases} \left(H^{(e+1/2)}\phi\right)(y) &= \int_Y dy'\, \Sigma_s{}^{(e+1/2)}(y' \to y)\, \phi(y'), \\ \left(F^{(e+1/2)}\phi\right)(y) &= \int_Y dy' \left(\chi \upsilon \Sigma_f\right)^{(e+1/2)}(y' \to y)\phi(y')\,, \end{cases} \qquad (3.12)$$

4- **Accélération du flux :**

Le flux angulaire accéléré est obtenu par :

$$\psi^{(e+1)}(x) = \psi^{(e+1/2)}(x)\frac{\phi^{(e+1)}(y)}{\phi^{(e+1/2)}(y)} \qquad \forall\, x \in y\,. \qquad (3.13)$$

A noter que la nouvelle source de fission qui alimente l'itération transport suivante est normalisée par la valeur propre $\lambda^{(e+1)}$ calculée par l'opérateur synthétique.

L'algorithme de résolution est résumé dans l'Algorithme 3.4 ci-dessous. Les itérations sont arrêtées lorsque les critères de convergence sur la valeur propre et la source de fission sont vérifiés simultanément.

Algorithme 3.4. Algorithme classique accéléré par le CMFD.

Itérations externes (e)

Tant que λ *et* $(F\psi)(x)$ *ne sont pas convergés,* ***faire***

1- ***Balayage transport multigroupe*** *:*

$$(L - H)\psi^{(e+1/2)}(x) = \frac{(F\psi)^{(e)}(x)}{\lambda^{(e)}}, \quad x \in X\,.$$

2- ***Homogénéisation-condensation et équivalence***

Flux et courants :

$$\boldsymbol{J}^{(e+1/2)}(y) = \int_y dx\, \boldsymbol{\Omega}\, \psi^{(e+1/2)}(x)\,,$$

$$\phi^{(e+1/2)}(y) = \int_y dx\, \psi^{(e+1/2)}(x)\,,$$

Sections efficaces :

$$\Sigma_t{}^{(e+1/2)}(y) = \frac{\int_y dx\, \Sigma_t(x)\psi^{(e+1/2)}(x)}{\phi^{(e+1/2)}(y)},$$

$$\Sigma_s{}^{(e+1/2)}(y' \to y) = \frac{\int_y dx \int_{y'} dx'\, \Sigma_s(x' \to x)\psi^{(e+1/2)}(x')}{\phi^{(e+1/2)}(y')},$$

$$\left(\chi \upsilon \Sigma_f\right)^{(e+1/2)}(y' \to y) = \frac{\int_y dx \sum_i \chi_i(x) \int_{y'} dx'\, (\upsilon\Sigma_f)_i \psi^{(e+1/2)}(x')}{\phi^{(e+1/2)}(y')}.$$

Paramètres d'équivalence et coefficients de diffusion :

$$\tilde{\boldsymbol{D}}^{\left(e+\frac{1}{2}\right)}(y) = \left[\frac{\boldsymbol{J}(y) + D\boldsymbol{\nabla}\phi(y)}{\phi(y)}\right]^{\left(e+\frac{1}{2}\right)}, \qquad D^{\left(e+\frac{1}{2}\right)} = \left[\frac{1}{3(\Sigma_t - \Sigma_{s1})}\right]^{\left(e+\frac{1}{2}\right)}.$$

*3- **Solution de l'opérateur synthétique** :*

$$\begin{cases} \boldsymbol{\nabla} \cdot \boldsymbol{J}^{(e+1)}(y) + (\Sigma_t - H)^{(e+1/2)}\phi^{(e+1)}(y) = \frac{1}{\lambda^{(e+1)}} F^{(e+1/2)}\phi^{(e+1)}(y), \\ \boldsymbol{J}^{(e+1)}(y) = -D^{(e+1/2)}\boldsymbol{\nabla}\phi^{(e+1)}(y) + \tilde{\boldsymbol{D}}^{(e+1/2)}\phi^{(e+1)}(y). \end{cases}$$

*4- **Accélération du flux** :*

$$\psi^{(e+1)}(x) = \psi^{(e+1/2)}(x)\frac{\phi^{(e+1)}(y)}{\phi^{(e+1/2)}(y)} \qquad \forall\, x \in y.$$

fin

L'opérateur de diffusion défini par les équations (3.10) et (3.11) est équivalent à l'opérateur de transport (3.2) homogénéisé sur l'espace grossier uniquement lorsqu'il est construit à partir de la solution $\psi(x)$ du problème (3.2). Ainsi lorsque le flux du problème de transport est convergé, le CMFD donne le même flux moyen que le transport et l'accélération ne modifie pas la solution. De plus le CMFD donne la même valeur propre et source de fission.

L'opérateur CMFD permet de réduire plus rapidement les erreurs sur le mode fondamental de la solution transport, le mode fondamental étant le plus lent à converger. Cela est d'autant plus vrai quand la solution $\psi^{(e+1/2)}(x)$ est proche de la convergence. C'est d'ailleurs pour cela que l'opérateur CMFD est un excellent outil pour accélérer les calculs d'évolution des assemblages et pour la réalisation de calculs de sensibilité à moindre coût.

La résolution de l'équation de la diffusion est réalisée en substituant l'expression du courant (3.11) dans l'équation de bilan (3.10). De cette manière on obtient l'équation sur le flux :

$$\left[\boldsymbol{\nabla}\cdot\left(-D\boldsymbol{\nabla}+\widetilde{\boldsymbol{D}}\right)+(\Sigma_t-H)\right]^{(e+1/2)}\phi^{(e+1)}(y)=\frac{1}{\lambda^{(e+1)}}F^{(e+1/2)}\phi^{(e+1)}(y), \quad (3.14)$$

où l'opérateur $\boldsymbol{\nabla}\cdot\left(-D\boldsymbol{\nabla}+\widetilde{\boldsymbol{D}}\right)$ est discrétisé par la méthode des différences finies.

Dans le solveur IDT, l'équation (3.14) est résolue par des itérations de puissance. La partie non diagonale, notée $H_{\backslash D}\phi$, de l'opérateur de scattering $H\phi$ est séparée de la partie diagonale, $H_D\phi$. Cette séparation engendre des itérations internes sur la partie diagonales et des itérations thermiques sur la partie non diagonale. L'opérateur $K=\left[\boldsymbol{\nabla}\cdot\left(-D\boldsymbol{\nabla}+\widetilde{\boldsymbol{D}}\right)+\Sigma_t-H_D\right]$, qui caractérise la solution à l'intérieur d'un macro groupe, est inversé par une méthode de Krylov basée sur le BiCGStab multi-grille avec un pré-conditionnement ILU(0) [41]. Enfin, l'opérateur CMFD multigroupe $[1-K^{-1}H_{\backslash D}]$ est inversé directement pour les groupes rapides tandis qu'il est inversé itérativement par l'algorithme de Gauss-Seidel pour les groupes thermiques.

3.3.2 Présentation du CMFD dans le solveur IDT

Dans cette section, nous présenterons le CMFD appliqué à l'algorithme Direct dans le solveur IDT. Ce CMFD a servi de base de travail son application à l'algorithme PMBJ de la DDM.

3.3.2.a Calcul des paramètres homogénéisés

Nous venons de voir que le CMFD est construit en établissant l'opérateur de diffusion sur un maillage grossier. A partir de l'opérateur de transport fin défini sur un maillage discrétisé en N_r mailles spatiales et N_g groupes, on définit le maillage cartésien conforme grossier par $N_R \leq N_r$ mailles spatiales grossières et $N_G \leq N_g$ macro groupes d'énergie. Les notations utilisées dans la suite sont représentées sur la Figure 3.9 : les indices en lettres minuscules font référence à la discrétisation fine et les indices en lettres majuscules font référence au maillage grossier.

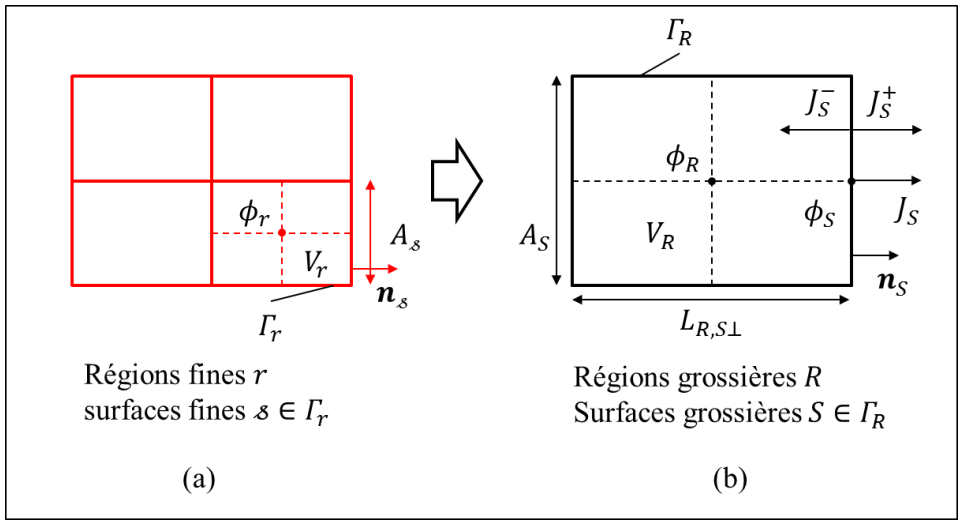


Figure 3.9. Représentation du maillage CMFD (b) par rapport au maillage transport (a) et notations utilisées.

La Figure 3.9(a) présente 4 régions du maillage fin qui sont homogénéisées en une maille grossière sur la Figure 3.9(b).

Les quantités liées au maillage fin sont les suivantes :

- V_r = volume de la maille fine r.
- A_s = aire de la surface s.
- Γ_r = surface de la maille fine r.
- $\boldsymbol{n}_s$ = normale sortante à la surface s.
- ϕ_r^g = flux scalaire moyen dans la maille fine r dans le groupe g.
- ϕ_s^g = flux scalaire moyen sur la surface s dans le groupe g.
- J_s^g = courant moyen sur la surface s dans le groupe g.

Et les quantités liées au maillage grossier sont définies par :

- $V_R = \sum_{r \in R} V_r$ est le volume de la maille grossière R.
- $A_S = \sum_{s \in S} A_s$ est l'aire de la surface S.
- Γ_R = surface de la maille grossière R.
- $\boldsymbol{n}_S$ = normale sortante à la surface S.
- $L_{R,S}^{\perp}$ = longueur de la maille grossière R dans la direction transverse S.
- ϕ_R^G = flux scalaire moyen dans la maille grossière R dans le groupe G.
- ϕ_S^G = flux scalaire moyen sur la surface S dans le groupe G.
- J_S^G = courant moyen sur la surface S dans le groupe G.

Afin d'alléger les notations, les indices $(e + 1/2)$ et $(e + 1)$ son omis dans la suite. Les quantités transport sont celles obtenues avant l'accélération. Nous préciserons ensuite lorsque la distinction devra être faite. Le flux scalaire intégré sur la maille fine r, le flux scalaire intégré sur la surface s ainsi que la composante normale du courant intégré sur la surface s sont donnés par :

$$\phi_r^g = \int_{V_r} d\boldsymbol{r} \int_{4\pi} \boldsymbol{d\Omega}\, \psi_r^g(\boldsymbol{\Omega}, \boldsymbol{r}), \tag{3.15}$$

$$\phi_s^g = \int_{A_s} d\boldsymbol{r}_s \int_{4\pi} d\boldsymbol{\Omega}\, \psi_s^g(\boldsymbol{\Omega}, \boldsymbol{r}_s), \tag{3.16}$$

$$J_s^g = \int_{A_s} d\boldsymbol{r}_s \int_{4\pi} d\boldsymbol{\Omega}\, \psi_s^g(\boldsymbol{\Omega}, \boldsymbol{r}_s)\, \boldsymbol{\Omega} \cdot \boldsymbol{n}_s\,, \tag{3.17}$$

$$\forall r, \qquad \forall s, \qquad \forall g,$$

où ψ_r^g et ψ_s^g sont respectivement le flux angulaire dans la région r et le flux angulaire sur la surface s après le balayage transport multigroupe. On peut noter que l'intégrale angulaire est calculée numériquement par la quadrature angulaire associée aux ordonnées discrètes. L'intégrale spatiale est analytique : avec référence à la représentation polynomiale du flux dans IDT (cf. équation (1.14)), les équations (3.15) à (3.17) engendrent seulement les moments zéro.

Pour exprimer le gradient du courant dans la direction $\boldsymbol{n}_s$ par la formulation en différences finies, seule la composante normale du courant à la surface s est nécessaire, d'où l'expression (3.17).

Le flux scalaire moyen ϕ_R^G, le flux scalaire moyen ϕ_S^G sur la surface S et le courant moyen J_S^G sur la surface S sont définis de la manière suivante :

$$\phi_R^G = \frac{1}{V_R}\sum_{r\in R}\sum_{g\in G}\phi_r^g, \tag{3.18}$$

$$\phi_S^G = \frac{1}{A_S}\sum_{s\in S}\sum_{g\in G}\phi_s^g, \tag{3.19}$$

$$J_S^G = \frac{1}{A_S}\sum_{s\in S}\sum_{g\in G}J_s^g, \tag{3.20}$$

$$\forall R, \qquad \forall S, \qquad \forall G,$$

La section totale, la matrice de scattering et la matrice de fission sont respectivement obtenues de la manière suivante :

$$\begin{aligned}
\Sigma_{t,R}^G &= \frac{1}{V_R\phi_R^{G'}}\sum_{r\in R}\sum_{g\in G}\Sigma_{t,r}^g\phi_r^g \ , \\
\Sigma_{s,R}^{G'\to G} &= \frac{1}{V_R\phi_R^{G'}}\sum_{r\in R}\sum_{g\in G}\sum_{g'\in G'}\Sigma_{s,r}^{g'\to g}\phi_r^{g'} \ , \\
\left(\chi\nu\Sigma_f\right)_R^{G'\to G} &= \frac{1}{V_R\phi_R^{G'}}\sum_{r\in R}\sum_{i\in r}\sum_{g\in G}\chi_i^g\sum_{g'\in G'}\left(\nu\Sigma_f\right)_i^{g'}\phi_r^{g'} \ ,
\end{aligned} \tag{3.21}$$

$$\forall R, \qquad \forall G,$$

où i est l'indice des isotopes fissiles contenus dans les régions r. Ces quantités correspondent à la formulation discrète des équations (3.6) et (3.7).

L'équation (3.11) discrétisée par la méthode des différences finies prend la forme suivante :

$$J_S^G = -d_{R,S}^G(\phi_S^G - \phi_R^G) + \tilde{d}_{R,S}^G f(\phi_S^G, \phi_R^G), \qquad \forall R, \forall S \in \Gamma_R, \forall G, \tag{3.22}$$

où :

$$d_{R,S}^G = \frac{D_{R,S}^G}{L_{R,S}^{\perp}/2} \ , \qquad D_{R,S}^G = \frac{1}{3\left(\Sigma_{t,R}^G - \Sigma_{s1,R}^{G\to G}\right)}, \tag{3.23}$$

$$\tilde{d}_{R,S}^G = \frac{J_S^G + d_{R,S}^G(\phi_S^G - \phi_R^G)}{f(\phi_s^G, \phi_R^G)}. \tag{3.24}$$

La loi de fermeture $f(\phi_S^G, \phi_R^G)$ peut être définie de plusieurs manières. Moon, Cho, Noh et Hong [42] proposent la somme du flux volumique et du flux surfacique (formulation MCNH) :

$$f(\phi_S^G, \phi_R^G) \quad = \quad \phi_S^G + \phi_R^G \,. \tag{3.25}$$

Aragones et Ahnert [43] utilisent uniquement le flux surfacique (formulation AA) :

$$f(\phi_S^G, \phi_R^G) \quad = \quad \phi_S^G. \tag{3.26}$$

De manière similaire, le flux volumique seul peut être utilisé (formulation AFC) :

$$f(\phi_S^G, \phi_R^G) \quad = \quad \phi_R^G \,. \tag{3.27}$$

Des détails sur ces méthodes peuvent être trouvés dans [44] et [45]. Toutes ces formulations sont disponibles dans le solveur IDT. La formulation MCNH (3.25) est celle utilisée dans la suite de ce travail.

3.3.2.b Formulation de l'opérateur du CMFD

Pour établir les matrices permettant la résolution du système, il faut exprimer le courant en fonction des flux scalaires volumiques.

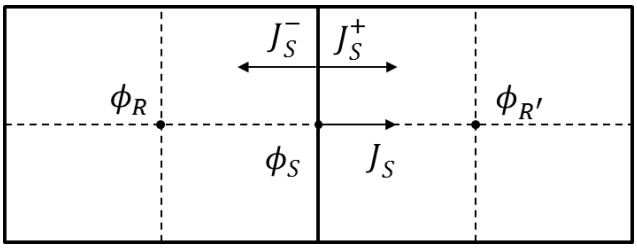


Figure 3.10. Interface entre deux mailles grossières et notations utilisées.

En utilisant la continuité du flux et du courant à l'interface *S* entre deux mailles *R* et R' (cf. Figure 3.10) :

$$J_S^G = \begin{cases} -d_{R,S}^G(\phi_S^G - \phi_R^G) + \tilde{d}_{R,S}^G f(\phi_S^G, \phi_R^G), \\ -d_{R',S}^G\left(\phi_{R'}^G - \phi_S^G\right) + \tilde{d}_{R',S}^G f\left(\phi_S^G, \phi_{R'}^G\right), \end{cases}$$

et après avoir éliminé le flux d'interface, on obtient des équations de la forme :

$$J_S^G = a_S^G \, \phi_R^G - b_S^G \phi_{R'}^G, \qquad \forall S, G. \tag{3.28}$$

Les coefficients a_S^G et b_S^G sont calculés à partir de l'expression du courant corrigé (3.22) et des lois de fermeture du CMFD énoncées ci-dessus (équations (3.25) à (3.27)). Avec la relation de fermeture MCNH (3.25), on a :

$$a_S^G = \frac{\left(d_{R,S}^G + \tilde{d}_{R,S}^G\right)}{\left(d_{R,S}^G - \tilde{d}_{R,S}^G\right) + \left(d_{R',S}^G - \tilde{d}_{R',S}^G\right)}, \tag{3.29}$$

et :

$$b_S^G = \frac{\left(d_{R',S}^G + \tilde{d}_{R',S}^G\right)}{\left(d_{R,S}^G - \tilde{d}_{R,S}^G\right) + \left(d_{R',S}^G - \tilde{d}_{R',S}^G\right)}. \tag{3.30}$$

La forme discrète de l'équation (3.14) est alors :

$$\begin{aligned}\sum_{S\in\Gamma_R} \frac{A_S}{V_R}\left(a_S^G\ \phi_R^G - b_S^G \phi_{R'(S)}^G\right) + \left(\Sigma_{t,R}^G - \Sigma_{s,R}^{G\to G}\right)\phi_R^G \\ = \sum_{\forall G'\neq G} \Sigma_{s,R}^{G'\to G}\phi_R^{G'} + \frac{1}{\lambda}\sum_{\forall G'} (\chi\nu\Sigma)_{f,R}^{G'\to G}\phi_R^{G'}\ ,\qquad \forall R, G.\end{aligned} \tag{3.31}$$

$R'(S)$ est la région séparée de R par l'interface S. On définit alors le problème à valeur propre suivant :

$$\left(K - H_{\backslash D}\right)\phi = \frac{1}{\lambda} F\phi\ . \tag{3.32}$$

avec :

$$\begin{cases} (K\phi)_R^G &= \left(\sum_{S\in\Gamma_R} \frac{A_S}{V_R} a_S^G\ + \Sigma_{t,R}^G - \Sigma_{s,R}^{G\to G}\right)\phi_R^G - \sum_{S\in\Gamma_R} \frac{A_S}{V_R} b_S^G \phi_{R'(S)}^G\,, \\ \left(H_{\backslash D}\phi\right)_R^G &= \sum_{\forall G'\neq G} \Sigma_{s,R}^{G'\to G}\phi_R^{G'}\,, \\ (F\phi)_R^G &= \sum_{\forall G'\in N_G} (\chi\nu\Sigma)_{f,R}^{G'\to G}\phi_R^{G'}\ . \end{cases} \tag{3.33}$$

La matrice *K* des coefficients est penta-diagonale en 2D et hepta-diagonale 3D en faisant abstraction des conditions aux limites.

3.3.2.c Traitement des conditions aux limites dans IDT

Dans le solveur IDT, les conditions aux limites sont traitées de deux manières différentes. La première est une solution générique et la seconde est le cas particulier de la réflexion.

SOLUTION GENERIQUE POUR TRAITER LES CONDITION AUX LIMITES

Pour traiter des conditions aux limites arbitraires, nous définissons une relation d'albédo à la surface qui remplace la relation de fermeture (3.22) :

$$J_S^G = \beta_S^G \phi_R^G\,,\qquad \beta_S^G = \frac{J_S^G}{\phi_R^G},\qquad \forall S \in \Gamma_R \cap \Gamma, \forall G. \tag{3.34}$$

L'albédo β_S^G est calculé à partir des grandeurs transport homogénéisées sur le maillage grossier.

De cette manière, l'équivalence transport-diffusion est assurée à convergence. L'opérateur K est alors modifié pour les régions dont au moins une surface est en contact avec le bord du domaine :

$$(K\phi)_R^G = \left(\sum_{S \in \Gamma_R \cap \Gamma} \frac{A_S}{V_R} \beta_S^G + \sum_{S \in \Gamma_R \setminus \Gamma} \frac{A_S}{V_R} a_S^G + \Sigma_{t,R}^G - \Sigma_{s,R}^{G \to G} \right) \phi_R^G - \sum_{S \in \Gamma_R \setminus \Gamma} \frac{A_S}{V_R} b_S^G \phi_{R'(S)}^G , \qquad \forall R, G. \tag{3.35}$$

La notation $\Gamma_R \setminus \Gamma$ signifie Γ_R privé de son intersection avec Γ.

Cette solution permet de conserver la structure multi-diagonale de l'opérateur K.

CONDITION AUX LIMITES DE REFLEXION

La condition de réflexion sur une surface implique un albédo β_S^G nul à la surface, ce qui n'est pas nécessairement vérifié pendant les itérations. L'équation (3.35) est alors modifiée de la manière suivante :

$$(K\phi)_R^G = \left(\sum_{S \in \Gamma_R \cap \Gamma \setminus \Gamma^{refl}} \frac{A_S}{V_R} \beta_S^G + \sum_{S \in \Gamma_R \setminus \Gamma} \frac{A_S}{V_R} a_S^G + \Sigma_{t,R}^G - \Sigma_{s,R}^{G \to G} \right) \phi_R^G - \sum_{S \in \Gamma_R \setminus \Gamma} \frac{A_S}{V_R} b_S^G \phi_{R'(S)}^G , \qquad \forall R, G, \tag{3.36}$$

où Γ^{refl} fait référence aux surfaces ayant une condition de réflexion. La condition de réflexion est ainsi traitée de manière exacte.

3.3.2.d Accélération du flux et de la valeur propre dans IDT

Le flux $\phi^{(e+1)}$ solution du système (3.32) est utilisé pour accélérer les moments spatiaux et angulaires du flux transport par la relation suivante :

$$\varphi_{r,n,l,m}^{g\ (e+1)} = \varphi_{r,n,l,m}^{g\ (e+1/2)} \times \frac{\phi_R^{G\ (e+1)}}{\phi_R^{G\ (e+1/2)}}, \tag{3.37}$$

$$\forall g \in G, \qquad \forall\, r \in R, \qquad \forall\, n \in N_V, \qquad \forall\, l \leq L, \qquad m = -l, \dots, l\,,$$

Où les moments du flux $\varphi_{r,n,l,m}^g$ sont donnés par l'équation (1.15). N_V est le nombre de moments spatiaux et L est l'ordre d'anisotropie du calcul. Dans le solveur IDT, tous les ordres du développement spatial et angulaire du flux sont accélérés.

La source de fission de l'opérateur de transport est ensuite calculée à partir du flux accéléré et normalisée par la valeur propre $\lambda^{(e+1)}$ obtenue par la solution du CMFD (3.32).

3.3.2.e Accélération des itérations internes

L'équation (3.32) est la formulation du CMFD pour un problème multigroupe. Cette équation peut être établie pour un seul groupe, noté g, en prenant les sources externes au groupe fixes :

$$(K\phi)^g = Q^g \,, \tag{3.38}$$

où Q^g est la somme de la source de scattering et de la source de fission dans le groupe :

$$Q^g = \left(H_{\setminus D}\phi\right)^g + \frac{1}{\lambda}(F\phi)^g. \tag{3.39}$$

La solution de l'équation (3.38) permet d'accélérer le flux dans les itérations internes en corrigeant les moments du flux comme présenté ci-dessous :

$$\varphi_{r,k,m}^{g\ (e+1)} = \varphi_{r,k,m}^{g\ (e+1/2)} \times \frac{\phi_R^{g(e+1)}}{\phi_R^{g(e+1/2)}}, \qquad \forall\, r \in R, \quad \forall\, k \in n_k, \quad \forall\, m \in n_m. \tag{3.40}$$

Notons que dans le solveur IDT, les conditions aux limites homogènes sont elles aussi accélérées. Pour cela, la formulation donnée par l'équation (3.46) est appliquée au flux angulaire à la frontière du domaine pour les directions entrantes et sortantes.

3.3.2.f Stabilité du CMFD

Bien que largement utilisée pour accélérer le transport, il a été depuis longtemps montré que la stabilité du CMFD n'est pas garantie et que l'accélération n'est pas inconditionnellement convergente lorsque l'épaisseur optique d'un maille est supérieure à 1 libre parcours moyen [46, 47, 48, 49, 50]. La plupart de ces études réalisent une analyse de Fourier en linéarisant le CMFD et montrent que le rayon spectral peut être supérieur à l'unité.

f.1. Revue des méthodes de stabilisation

Dernièrement, une analyse de la stabilité par une analyse de Fourier pour des problèmes à valeur propre a été publiée [51]. Cette étude préconise de réduire la taille des mailles grossières et/ou augmenter le nombre d'itérations internes par itération externe pour stabiliser le CMFD et montre que les résultats expérimentaux concordent très bien avec l'analyse de Fourier. D'autres publications présentent les différentes techniques qui peuvent être mises en œuvre pour stabiliser le CMFD. L. Li [52] propose une relaxation du coefficient correctif $\tilde{d}_{R,S}^G$ entre deux itérations successives :

$$\tilde{d}_{R,S}^{G\ (e+1/2)} = \eta \tilde{d}_{R,S}^{G\ (e+1/2)} + (1-\eta)\tilde{d}_{R,S}^{G\ (e-1/2)}. \tag{3.41}$$

Dans cette étude, le paramètre de relaxation η est choisi empiriquement. En complément de ce facteur stabilisant, deux autres méthodes sont proposées. La

première consiste à réaliser plusieurs itérations thermiques pour stabiliser la source de scattering multigroupe. Dans la seconde méthode, le facteur d'accélération (3.37) est modifié. La méthode s'appuie sur une homogénéisation à l'échelle de la cellule (géométrie contenant un crayon, sa gaine et le modérateur), chaque cellule étant discrétisée en secteurs. L'auteur propose la correction suivante :

$$\psi_r^{g(e+1)}(\boldsymbol{\Omega}) = \psi_r^{g(e+1/2)}(\boldsymbol{\Omega}) \times \left(\frac{2}{3} \frac{\phi_R^{G(e+1)}}{\phi_R^{G(e+1/2)}} + \frac{1}{3} \frac{\phi_{R'}^{G(e+1)}}{\phi_{R'}^{G(e+1/2)}} \right). \tag{3.42}$$

Dans la maille R, pour les régions r dont le secteur est en contact avec la maille R', le flux est corrigé par une somme pondérée des facteurs d'accélération des mailles R et R'.

M. Jarrett [53] conseille lui aussi une relaxation du coefficient correctif $\tilde{d}_{R,S}^G$ entre deux itérations successives et de réaliser plusieurs itération internes avant d'accélérer. Il propose aussi d'accroître le coefficient de diffusion par un terme additif défini arbitrairement avec pour conséquence de stabiliser le CMFD mais aussi d'augmenter son rayon spectral. Il suggère finalement de raffiner le maillage du CMFD pour améliorer le rayon spectral de la méthode.

f.2. Stabilisation dans le solveur IDT

En ce qui concerne le CMFD implémenté dans le solveur IDT, plusieurs techniques de stabilisation ont été implémentées. Dans [50], E. Masiello réalise une étude de la stabilité du CMFD en 1D pour deux régions en milieu infini pour analyser l'effet de l'homogénéisation et d'un paramètre d'interface. Il préconise un maillage grossier défini de manière à minimiser les différences d'épaisseur optiques entre les régions et suggère que l'homogénéisation peut être utilisé comme moyen de stabilisation pour le CMFD. Dans [3], il explicite la manière dont le coefficient de diffusion $D_{R,S}$ dans un groupe est modifié de manière à améliorer l'approximation par la diffusion:

$$D_{R,S} = e \times \frac{1}{3\left(\Sigma_{t,R} - \Sigma_{s1,R}\right)},$$

avec:

$$e = max\left(\max_{\forall S} (e_S \times f_S) ; 1 \right).$$

Ici, e_S est le facteur d'Eddington à la surface S inspiré de la quasi-diffusion [54] :

$$e_S = \frac{\theta_S}{\phi_S},$$

avec :

$$\theta_S = \frac{1}{A_S} \sum_{s \in S} \int_{A_s} d\boldsymbol{r}_s \int_{4\pi} d\boldsymbol{\Omega}\, \psi_s^{(e+1/2)}(\boldsymbol{\Omega}, \boldsymbol{r}_s)\, |\boldsymbol{\Omega} \cdot \boldsymbol{n}_s|^2.$$

De manière à préserver la formulation simple en différences finies, les composantes parallèles à la surface S sont négligées ce qui permet de transformer le tenseur d'Eddington en simple facteur.

f_S est un facteur garantissant la stabilité du CMFD. Il dépend du rapport $c = \Sigma_{s,R}/\Sigma_{t,R}$ et de l'épaisseur optique des régions R et R' voisines à l'interface S. Ce paramètre de stabilité permet de faire en sorte que l'accélération soit inconditionnellement convergente avec, dans la pire situation, le rayon spectral de l'opérateur de transport non accéléré [55].

Pour assurer la stabilité de l'algorithme par rapport à la précision numérique des flottants, d'autres paramètres ont été implémentées. Tout d'abord, les critères de convergence du CMFD sont de 10^{-2} fois ceux de l'opérateur de transport pour l'accélération des itérations de puissance. Pour l'accélération des itérations internes, la convergence est vérifiée par une norme L^2 du flux dans le CMFD alors que la convergence est vérifiée par une norme L^∞ dans les itérations transport. De manière à forcer la convergence dans les itérations internes du CMFD, qui utilise des réels double précision, la tolérance sur le flux est fixée à la précision numérique (10^{-12}). Si les itérations internes du CMFD n'ont pas convergé, un critère d'acceptation a été ajouté : l'erreur doit être inférieure à 10^{-7}. L'objectif est d'éviter les instabilités lors de la correction des flux transport.

3.3.3 Implémentation du CMFD dans la DDM

La décomposition de domaine associée à l'accélération par le CMFD a déjà été proposée par E. Masiello, B. Martin et J-M Do dans [56] où l'opérateur de transport est discrétisé par la méthode des caractéristiques courtes pour des mailles cartésiennes homogènes et sous domaines intégralement homogénéisés à l'échelle de l'assemblage. Ce travail est à l'origine de la méthode proposée dans cette thèse. Dans [57] B.W. Kelley propose une DDM appliquée à l'équation monogroupe du transport accélérée par le CMFD. Il présente une analyse de Fourier de la méthode pour un calcul 1D avec des conditions aux limites de réflexion et de vide. Cette analyse, dont la cohérence avec les résultats numérique est illustrée, apporte des éléments confirmant la stabilité du schéma et l'efficacité du CMFD pour accélérer une DDM. Cette méthode a été utilisée dans [34] où le MOC est couplé au CMFD pour des calculs HPC.

Dans cette section, nous verrons la manière dont le CMFD est appliqué à l'accélération de la DDM avec toutes les stratégies de balayage des sous domaines.

3.3.3.a Maillages énergétique et spatial du CMFD pour la DDM

Dans le contexte de l'accélération des itérations externes, le CMFD est appliqué à l'échelle du domaine global. Le maillage spatial et énergétique du CMFD est défini à la fois par rapport au maillage transport fin mais aussi par rapport à la décomposition de domaine.

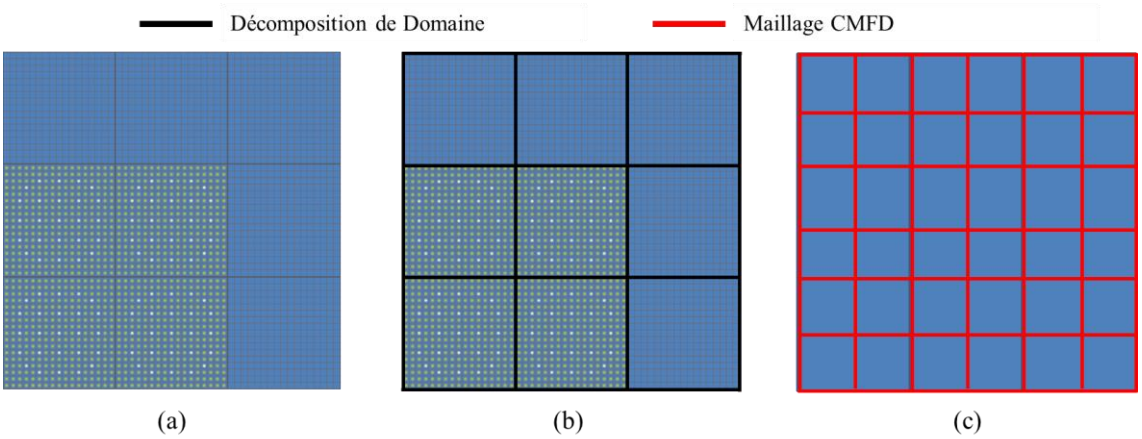


Figure 3.11. Exemple de décomposition de domaine et maillage du CMFD. Le domaine global (a) est décomposé en 9 sous domaines (b). Le maillage CMFD (c) inclue les divisions des sous domaines et est inclus dans le maillage fin.

La Figure 3.11 présente un exemple de décomposition de domaine associée à un maillage CMFD dans la DDM. Le maillage spatial du CMFD est défini tel que $N_\alpha \leq N_R \leq N_r$ où N_α est le nombre de sous domaines, N_R le nombre de mailles du CMFD et N_r le nombre de mailles fines. Une contrainte supplémentaire est qu'une maille grossière ne peut pas contenir plusieurs sous domaines. Dans notre implémentation, le maillage du CMFD est établi par mots clés via le jeu de données : le nombre de mailles grossières souhaité sur chaque axe (x, y, z) est défini. Le maillage grossier s'adapte ensuite automatiquement au maillage fin de manière à ce que le maillage soit le plus régulier possible, respectant les directives de l'utilisateur.

Le maillage énergétique du CMFD est également défini par mot clé via le jeu de données. Il est contraint par $N_G \leq N_g$ (N_G est le nombre de macro groupes et N_g le nombre de groupes fins). Le maillage énergétique fin est divisé en deux parties : les groupes rapides $N_{g_{fast}}$ et les groupes thermiques $N_{g_{th}}$. Le nombre de macro groupes rapides, $N_{G_{fast}}$, est défini de manière à respecter au mieux le rapport entre le nombre de groupes rapides et le nombre de groupes total N_g de la discrétisation fine. Les $N_{g_{fast}}$ groupes rapides fins sont répartis régulièrement dans les $N_{G_{fast}}$ macro groupes rapides. De même, les $N_{g_{th}}$ groupes thermiques fins sont répartis régulièrement dans les $N_{G_{th}}$ macro groupes thermiques.

Nous reprenons dans la suite de cette section les notations de la section 2.4.2 (Décomposition de Domaine Spatiale par Bloc Multigroupe).

3.3.3.b Calcul des coefficients du CMFD et équivalence

Le flux angulaire à l'interface entre les sous domaines doit être calculé pour évaluer les coefficients d'équivalence $\tilde{d}_{R,S}^{G}$ de l'équation de fermeture (3.22). Prenons l'exemple de 2 sous domaines adjacents α et β. L'intégration angulaire du flux transport est réalisée en intégrant les composantes sortantes des deux sous domaines séparément. On a ainsi le flux scalaire partiels intégrés et les courants partiels intégrés sur le maillage fin aux intersections des surfaces Γ_α et Γ_β des deux sous domaines :

$$
\begin{aligned}
\phi_{s,\alpha}^{+,g} &= \int_{A_s} d\boldsymbol{r}_s \int_{\boldsymbol{\Omega}\cdot\boldsymbol{n}_s>0} d\boldsymbol{\Omega}\, \psi_{s,\alpha}^{g}(\boldsymbol{\Omega},\boldsymbol{r}_s)\ , \\
\phi_{s,\beta}^{-,g} &= \int_{A_s} d\boldsymbol{r}_s \int_{\boldsymbol{\Omega}\cdot\boldsymbol{n}_s<0} d\boldsymbol{\Omega}\, \psi_{s,\beta}^{g}(\boldsymbol{\Omega},\boldsymbol{r}_s)\ , \\
J_{s,\alpha}^{+,g} &= \int_{A_s} d\boldsymbol{r}_s \int_{\boldsymbol{\Omega}\cdot\boldsymbol{n}_s>0} d\boldsymbol{\Omega}\, \psi_{s,\alpha}^{g}(\boldsymbol{\Omega},\boldsymbol{r}_s)\,|\boldsymbol{\Omega}\cdot\boldsymbol{n}_s|\ , \\
J_{s,\beta}^{-,g} &= \int_{A_s} d\boldsymbol{r}_s \int_{\boldsymbol{\Omega}\cdot\boldsymbol{n}_s<0} d\boldsymbol{\Omega}\, \psi_{s,\beta}^{g}(\boldsymbol{\Omega},\boldsymbol{r}_s)\,|\boldsymbol{\Omega}\cdot\boldsymbol{n}_s|\ , \\
&\forall s \in \Gamma_\alpha \cap \Gamma_\beta, \qquad \forall g,
\end{aligned}
\tag{3.43}
$$

$\boldsymbol{n}_s$ étant la normale sortante du sous domaine α. Pour les interfaces internes au sous domaines ($\forall s \in S, \forall S \in (D_\alpha \setminus \Gamma_\alpha)$, $\forall\, D_\alpha \in D$), on retrouve les équations pour le courant (3.20) et pour le flux à la surface (3.19) en prenant $\beta = \alpha$. Les équations (3.43) permettent d'obtenir ces grandeurs sur le maillage grossier global, ainsi que le courant net :

$$
\begin{aligned}
\phi_S^G &= \frac{1}{A_S}\sum_{g\in G}\sum_{s\in S} \phi_{s,\alpha}^{+,g} + \phi_{s,\beta}^{-,g}\ , \\
J_S^{+,G} &= \frac{1}{A_S}\sum_{g\in G}\sum_{s\in S} J_{s,\alpha}^{+,g}\ , \\
J_S^{-,G} &= \frac{1}{A_S}\sum_{g\in G}\sum_{s\in S} J_{s,\beta}^{-,g}\ , \\
J_S^G &= J_S^{+,G} - J_S^{-,G}\ , \\
&\forall\, S \in \Gamma_\alpha \cap \Gamma_\beta, \qquad \forall G\ .
\end{aligned}
\tag{3.44}
$$

La relation de fermeture utilisée pour l'expression du courant est la formulation MCNH (cf. équation (3.25)) :

$$
J_S^G = -d_{R,S}^G(\phi_S^G - \phi_R^G) + \tilde{d}_{R,S}^G(\phi_S^G + \phi_R^G), \qquad \forall R, \forall S \in \Gamma_R, \forall G \tag{3.45}
$$

La matrice des coefficients est ainsi définie au niveau du domaine global (cf. équations (3.28) à (3.32)). Le traitement des conditions aux limites est présenté dans la section 3.3.2.c et la technique de stabilisation est celle d'IDT (§3.3.2f.2).

3.3.3.c Accélération des flux et de la valeur propre

c.1. Accélération du flux volumique et de la valeur propre

L'accélération du flux volumique et de la valeur propre est réalisée comme présenté section 3.3.2.d. Tous les moments spatiaux et angulaires du flux volumique sont accélérés comme présenté à l'équation (3.37) où le flux transport est pris dans chaque

sous domaine. Je rappelle que la valeur propre $\lambda^{(e+1)}$ est utilisée comme valeur propre pour la normalisation de la source de fission de l'opérateur de transport.

c.2. Accélération des flux angulaires surfaciques implémentée dans l'algorithme PMBJ

Une particularité de la DDM est la transmission du flux angulaire à l'interface entre les sous domaines. Alors que dans l'utilisation classique du CMFD, seuls les flux volumiques, les conditions aux limites et la valeur propre sont accélérés, dans son application à la DDM les flux d'interface entre les sous domaines sont eux aussi accélérés. Les flux angulaires sortants du sous domaine α sont corrigés par le flux scalaire de la région adjacente :

$$P0v: \quad \psi_{s,b,\alpha}^{g\ (e+1)}(\boldsymbol{\Omega}) = \psi_{s,b,\alpha}^{g\ \left(e+\frac{1}{2}\right)}(\boldsymbol{\Omega}) \times \frac{\phi_R^{G\,(e+1)}}{\phi_R^{G\,(e+1/2)}}, \tag{3.46}$$

$$\forall \alpha, \quad \forall g \in G, \quad \forall\, s \in S, \quad \forall\, b \in n_b, \quad \forall\, S \in \Gamma_R \cap \Gamma_\alpha, \quad \forall \boldsymbol{\Omega} \cdot \boldsymbol{n}_S > 0$$

Les n_b moments spatiaux du flux angulaire surfacique sont accélérés. Le nom $P0v$ utilisé pour désigner cette méthode d'accélération des flux surfaciques vient du fait que l'on utilise le flux scalaire (d'où le $P0$) volumique (d'où le v).

D'autres méthodes existent pour corriger les flux angulaires surfaciques. Dans [58], les auteurs établissent plusieurs relations pour accélérer les flux angulaires surfaciques avec la CMFD :

$$P0: \quad \psi_{s,\alpha}^{g\ (e+1)}(\boldsymbol{\Omega}) = \psi_{s,\alpha}^{g\ (e+1/2)}(\boldsymbol{\Omega}) \times \frac{\phi_S^{G\,(e+1)}}{\phi_S^{G\,(e+1/2)}},$$

$$DP0: \quad \psi_{s,\alpha}^{g\ (e+1)}(\boldsymbol{\Omega}) = \psi_{s,\alpha}^{g\ (e+1/2)}(\boldsymbol{\Omega}) \times \frac{J_S^{+,G\,(e+1)}}{J_S^{+,G\,(e+1/2)}}, \qquad \boldsymbol{\Omega} \cdot \boldsymbol{n}_S > 0,$$

$$P1: \quad \psi_{s,\alpha}^{g\ (e+1)}(\boldsymbol{\Omega}) = \psi_{s,\alpha}^{g\ (e+1/2)}(\boldsymbol{\Omega}) \times \frac{\phi_S^{G\,(e+1)} + 3(\boldsymbol{\Omega} \cdot \boldsymbol{n}_S) J_S^{G\,(e+1)}}{\phi_S^{G\,(e+1/2)} + 3(\boldsymbol{\Omega} \cdot \boldsymbol{n}_S) J_S^{G\,(e+1/2)}},$$

$$\forall\, \alpha, \quad \forall g \in G, \quad \forall\, s \in S, \quad \forall\, S \in \Gamma_\alpha$$

Le courant et le flux scalaire issus du CMFD sont calculés à partir des relations de fermeture du CMFD (formulation MCNH). Les courants partiels de la correction $DP0$ sont calculés par la relation issue de la théorie de la diffusion :

$$J_S^{\pm,G} = \frac{\phi_S^G}{4} \pm \frac{J_S^G}{2}.$$

Dans ce même papier [58], les auteurs mettent en avant le fait que le choix du type de correction n'a pas d'influence particulière sur la convergence du problème et suggèrent l'utilisation de la correction $P0$. Dans [59], les auteurs appliquent la

correction $P1$ pour corriger les flux aux interfaces entre les sous domaines. Nous avons réalisé des comparaisons succinctes avec les méthodes $P0$ et $DP0$, sans obtenir de résultats qui encouragent leur utilisation. De manière à éviter de devoir évaluer les grandeurs surfaciques issues du CMFD, la correction par le flux volumique, $P0v$, est utilisée pour la DDM dans ce travail.

3.3.3.d Accélération des itérations internes pour l'algorithme PMBJ

Le CMFD est aussi utilisé pour accélérer la convergence des itérations internes dans les sous domaines. L'accélération des itérations internes et celle des itérations externes sont complètement indépendantes. Dans le solveur IDT, le maillage du CMFD est identique au maillage cartésien du transport pour l'accélération des itérations internes.

La particularité de l'algorithme PMBJ est que l'accélération est réalisée par sous domaine : le flux entrant aux surfaces est imposé pour le calcul multigroupe (cf. équation (2.33)) et n'est pas modifié dans les itérations internes. Cette condition aux limites est caractérisée par l'égalité suivante :

$$\psi_s^g(\boldsymbol{\Omega}) = q_s^g(\boldsymbol{\Omega}) , \qquad \forall\, \boldsymbol{\Omega}\cdot\boldsymbol{n}_s < 0 , \qquad \forall s \in \Gamma_\alpha^{intf}.$$

$\Gamma_\alpha^{intf} = \Gamma_\alpha \setminus (\Gamma_\alpha \cap \Gamma)$ fait référence aux surfaces du sous domaine α ayant un flux entrant imposé, c'est-à-dire les interfaces entre sous domaines (je rappelle que Γ_α est la surface du sous domaine α et Γ la surface du problème global). $q_s^g(\boldsymbol{\Omega})$ représente ainsi le flux entrant imposé. Le courant est exprimé en séparant le courant partiel entrant du courant partiel sortant du sous domaine α :

$$\begin{aligned} J_S^{+,g} &= \frac{1}{A_S}\sum_{s\in S} J_s^{+,g} \\ J_S^{-,g} &= \frac{1}{A_S}\sum_{s\in S}\int_{A_s} d\boldsymbol{s} \int_{\boldsymbol{\Omega}\cdot\boldsymbol{n}_s<0} d\boldsymbol{\Omega}\; q_s^g(\boldsymbol{\Omega})\,|\boldsymbol{\Omega}\cdot\boldsymbol{n}_s| \;= Q_S^g \end{aligned} \qquad (3.47)$$

$$\forall\, S \in \Gamma_\alpha^{intf} .$$

Le courant entrant, ici noté $J_S^{-,g}$, est alors considéré comme source surfacique notée Q_S^g dans la suite. L'expression du courant sortant en fonction du flux est donnée par la relation suivante :

$$J_S^{+,g} = d_S^{intf,g}\phi_R^g , \qquad d_S^{intf,g} = \frac{J_S^{+,g}}{\phi_R^g}, \qquad \forall S \in \Gamma_\alpha^{inft}. \qquad (3.48)$$

Le coefficient $d_S^{intf,g}$ est calculé à partir des grandeurs transport homogénéisées sur le maillage grossier. De cette manière, l'équivalence est assurée à convergence. Pour les sous domaines en contact avec la surface du domaine de calcul global, les formulations utilisées sont celles du §3.3.2.c (Traitement des conditions aux limites dans IDT). Sur les surfaces internes au sous domaine, la relation de fermeture utilisée est la formulation MCNH (3.25).

L'opérateur K est alors modifié de la manière suivante pour l'accélération des itérations internes :

$$(K\phi)_R^g = \left\{ \sum_{S\in\Gamma_R\cap\Gamma_\alpha^{intf}} \frac{A_S}{V_R} d_S^{intf,g} + \sum_{S\in\Gamma_R\cap\Gamma\setminus\Gamma^{refl}} \frac{A_S}{V_R}\beta_S^g + \sum_{S\in\Gamma_R\setminus\Gamma_\alpha} \frac{A_S}{V_R} a_S^g + \Sigma_{t,R}^g - \Sigma_{s,R}^{g\to g} \right\} \phi_R^g - \sum_{S\in\Gamma_R\setminus\Gamma_\alpha} \frac{A_S}{V_R} b_S^g \phi_{R'(S)}^g , \qquad \forall R \in D_\alpha \tag{3.49}$$

Et le système à inverser est alors :

$$(K\phi)_R^g = Q_R^g + Q_S^g , \qquad \forall R \in D_\alpha, \forall S \in \Gamma_R , \tag{3.50}$$

avec Q_S^g la source à la frontière du sous domaine définie par :

$$Q_S^g = \begin{cases} J_S^{-,g}, & S \in \Gamma_\alpha^{intf}, \\ 0, & sinon \quad , \end{cases} \tag{3.51}$$

et Q_R^g la source externe au groupe :

$$Q_R^g = \left(H_{\setminus D}\phi\right)_R^g + \frac{1}{\lambda}(F\phi)_R^g , \qquad \forall\, R\,. \tag{3.52}$$

Les moments spatiaux et angulaires du flux volumique sont accélérés comme présenté dans l'équation (3.37) et les moments spatiaux du flux angulaire surfacique le sont comme dans l'équation (3.46).

3.3.3.e Algorithme PMBJ accéléré par le CMFD

L'Algorithme 3.5 présente la manière dont le CMFD est inséré dans l'algorithme non accéléré. Les modifications apparaissent en rouge. Après le calcul transport de chaque sous domaine, les flux, les courants et les sections efficaces sont moyennés sur le maillage grossier. Cette partie est réalisée par sous domaine et est facilement parallélisable. Ensuite le calcul des coefficients d'équivalence et de la matrice de CMFD est réalisé à l'échelle du domaine global. Finalement, le flux est accéléré dans chaque sous domaine et la source de fission est normalisée par la valeur propre obtenue par le CMFD.

Algorithme 3.5. Algorithme PMBJ accéléré par le CMFD.

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha\ \forall\alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux aux interfaces entre sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\,\beta \neq \alpha : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\,\beta \neq \alpha : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+1/2)}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha^-}^{(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Homogénéisation-condensation sur le maillage grossier pour le CMFD

Pour chaque $\alpha \in A$, ***faire***

$$\left[\phi\,(y),\ J^+(y),\ J^-(y),\ \Sigma_t(y),\ \Sigma_s(y),\ (\chi\nu\Sigma_f)(y)\right]^{(e+1/2)}$$

fin

Calcul des coefficients de diffusion et d'équivalence et construction de la matrice du CMFD

Solution de l'opérateur du CMFD :

$$(K - H_{\setminus D})\phi^{(e+1)}(y) = F\phi^{(e+1)}(y)/\lambda^{(e+1)}$$

Pour chaque $\alpha \in A$, ***faire***

Accélération des flux dans les sous domaines et sur leurs surfaces

$$\psi_\alpha^{(e+1)}(x) = \psi_\alpha^{(e+1/2)}(x)\frac{\phi^{(e+1)}(y)}{\phi^{(e+1/2)}(y)} \qquad \forall\, x \in y \in X_\alpha\ .$$

Calcul de la nouvelle source de fission normalisée

$${Q_\alpha}^{(e+1)}(x) = \frac{{(F\psi)_\alpha}^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

fin

3.3.4 Tests de démonstration de l'efficacité du CMFD appliqué au DDM et éléments d'optimisation

3.3.4.a Efficacité de la DDM+CMFD : comparaison avec le calcul Direct sur un colorset 6 x 6

L'objectif de ce test est d'évaluer l'ordre de convergence de l'algorithme PMBJ accéléré par le CMFD dans le cadre d'un calcul d'assemblages de type REP. Nous cherchons ainsi à évaluer le potentiel de scalabilité de l'algorithme PMBJ par rapport à l'algorithme Direct. En effet, une scalabilité idéale ne pourra être obtenue que si l'ordre de convergence de l'algorithme PMBJ accéléré par le CMFD est similaire à celui de l'algorithme Direct.

Afin d'étudier l'accélération du CMFD couplé à la DDM, nous avons repris le test des sections 2.5 et 3.2.3 en étudiant les différents algorithmes de balayage des sous domaines ainsi que le calcul Direct. On rappelle que ce test est un colorset en 2D composé de 36 sous domaines de la taille d'un assemblage de type REP, soit 10404 régions. Les calculs sont réalisés avec des sections efficaces macroscopiques à 281 groupes jusqu'à l'ordre P1 pour l'anisotropie. Les critères de convergence sont toujours fixés à 10^{-10} pour la valeur propre, la source de fission, les courants d'interface et le flux, dans le but de comparer le comportement asymptotique de la convergence des algorithmes. Le maillage du CMFD est composé de 64 mailles par assemblage et 26 groupes en énergie. Ce maillage est le même pour tous les calculs en DDM et le calcul Direct.

De même, nous nous sommes intéressés à la convergence de la valeur propre (Figure 3.12), des erreurs sur la valeur propre (Figure 3.13) et la source de fission (Figure 3.14) pour les deux algorithmes de type PMBG-S en plus des algorithmes Directs et PMBJ.

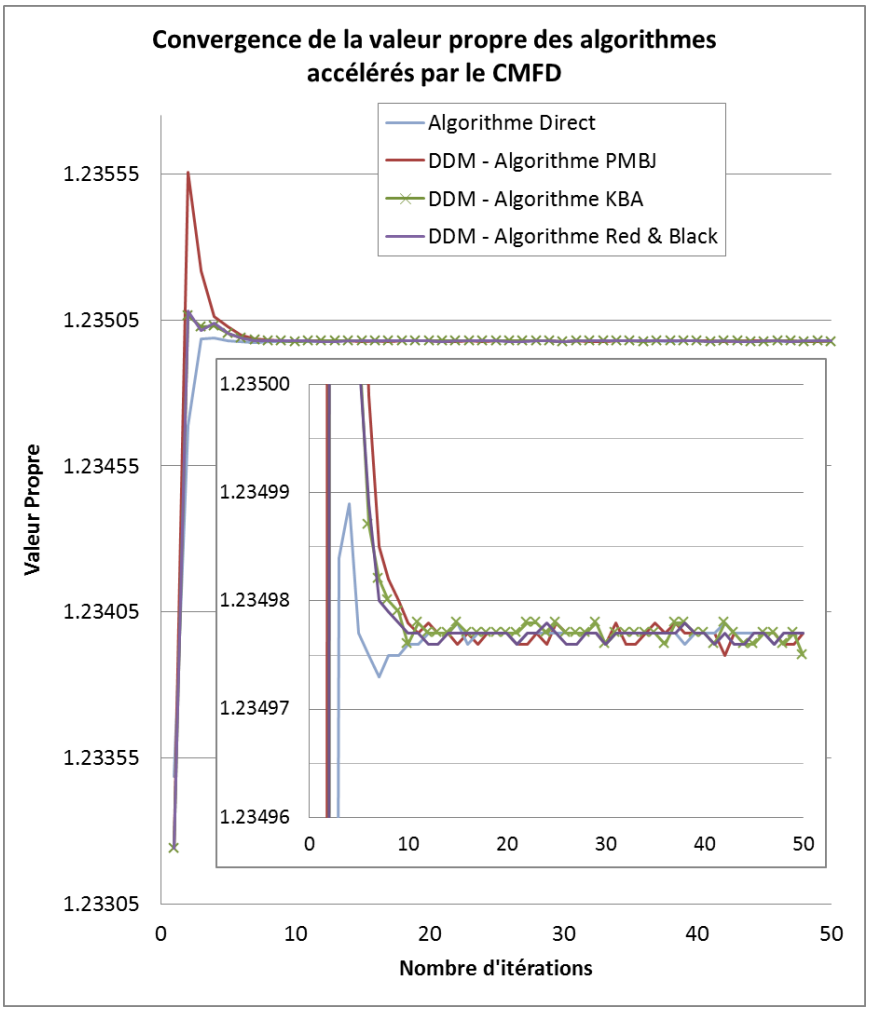


Figure 3.12. Convergence de la valeur propre pour les algorithmes accélérés par le CMFD.

L'oscillation dans la convergence (Figure 3.12) de la valeur propre est cette fois-ci présente quelle que soit la méthode, avec ou sans DDM. Cependant l'amplitude est bien plus faible que pour les calculs sans accélération par le CMFD (Figure 3.6).

La Figure 3.13 et la Figure 3.14 représentent respectivement la variation de l'erreur sur la valeur propre et la source de fission pour chaque itération externe. Avec les critères de convergence usuels utilisés pour le solveur IDT (10^{-5} pour la valeur propre et 10^{-4} pour la source de fission), les algorithmes avec DDM convergent en 9 itérations alors que le calcul direct en a besoin de 8. La différence du nombre d'itérations externes est alors réduite en comparaison aux algorithmes non accélérés (cf. page 71).

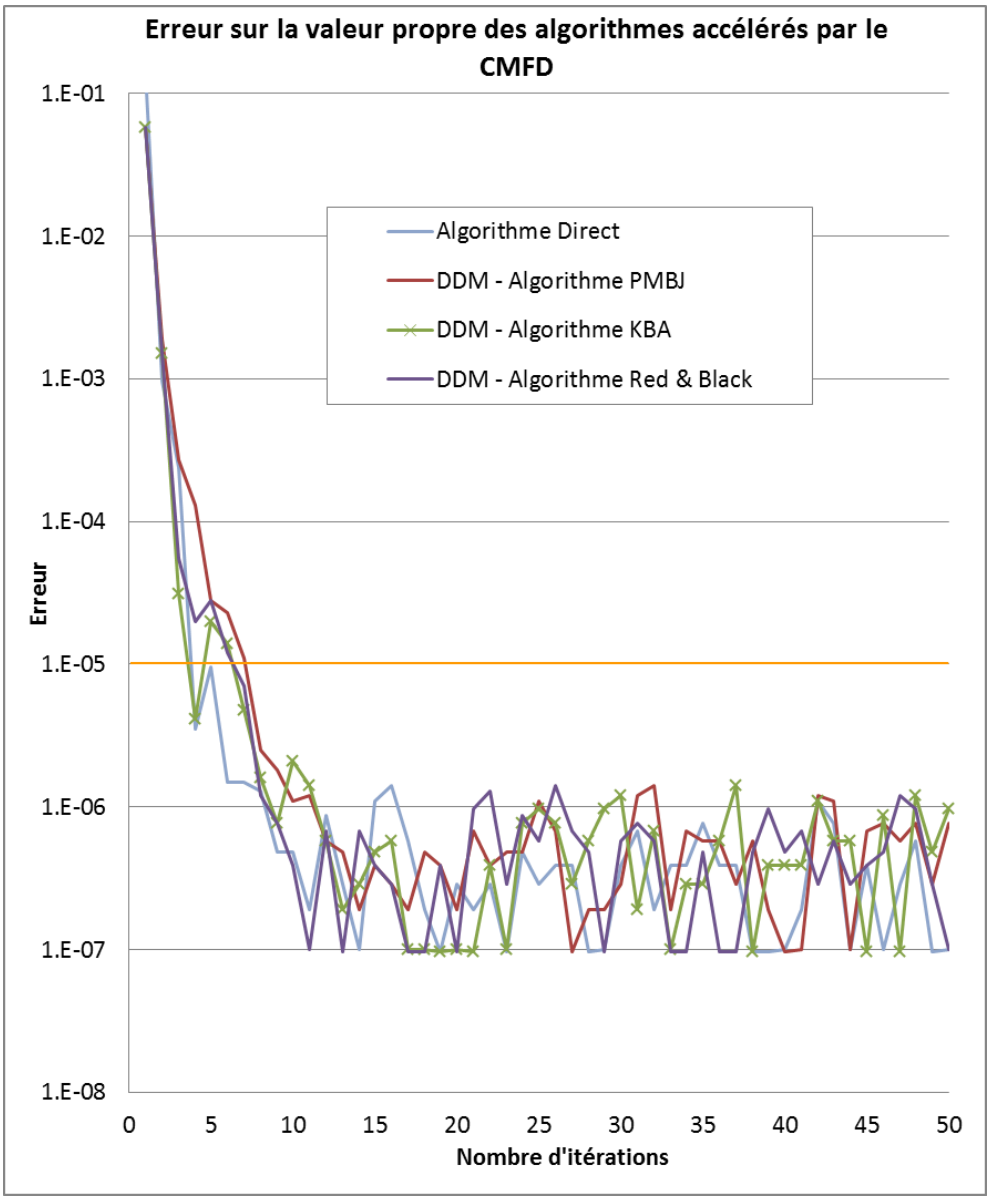


Figure 3.13. Convergence de l'erreur sur la valeur propre pour les différents algorithmes accélérés par le CMFD.

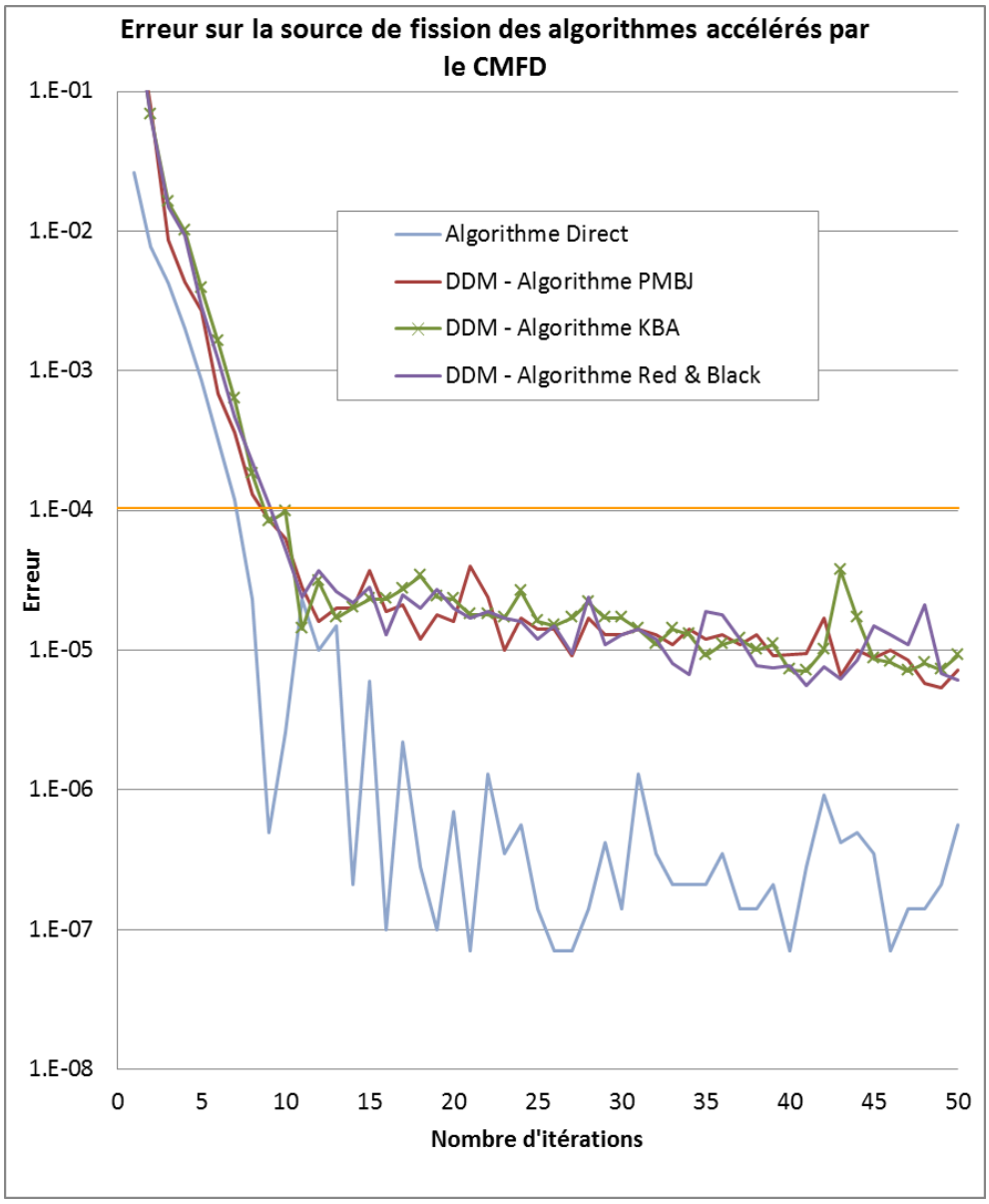


Figure 3.14. Convergence de l'erreur sur la source de fission pour les différents algorithmes accélérés par le CMFD.

Le Tableau 3.3 présente le rayon spectral mesuré pour les différents algorithmes de DDM, avec ou sans l'accélération par le CMFD.

Tableau 3.3. Rayon spectral des algorithmes Direct, PMBJ et PMBG-S, avec ou sans l'accélération par le CMFD.

	Rayon spectral			
Algorithme	Direct	PMBJ	KBA	Red-Black
Sans CMFD	0.818	0.946	0.941	0.941
Avec CMFD	0.385	0.400	0.394	0.396

Le rayon spectral est nettement amélioré dès que l'accélération par le CMFD est utilisée. De plus, l'écart de rayon spectral entre le calcul Direct et les calculs DDM est

très faible ce qui montre que le CMFD permet d'accélérer les communications entre les sous domaines. C'est le CMFD qui pilote la convergence du calcul.

Il est alors très intéressant de connaître le comportement de l'algorithme accéléré vis-à-vis de la taille du domaine de calcul. Pour cela, nous avons repris ce test pour 3 tailles du colorset : 3 x 3 assemblages, 6 x 6 assemblages (cas étudié jusque-là) et 9 x 9 assemblages par réplication du motif 3 x 3 de la même manière qu'aux paragraphes 2.5 et 3.2.3. Le nombre d'itérations externes estimé, N_e, et le rayon spectral, ρ, sont présentés pour chaque tailles de colorset et pour les algorithmes Direct, PMBJ et PMBG-S dans le Tableau 3.4 en prenant comme tolérance $\varepsilon = 10^{-5}$:

Tableau 3.4. Rayon spectral des algorithmes accélérés par le CMFD pour différentes tailles de colorset et estimation du nombre d'itérations externes pour converger l'intégrale de fission.

		Rayon spectral							
Algorithme / Taille du motif		Direct		PMBJ		KBA		Red-Black	
		ρ	N_e	ρ	N_e	ρ	N_e	ρ	N_e
Non accéléré	3 x 3	0.818	57	0.941	189	0.939	183	0.938	180
	6 x 6	0.818	57	0.946	207	0.941	189	0.941	189
	9 x 9	0.819	58	0.948	216	0.943	196	0.943	196
CMFD	3 x 3	0.379	12	0.401	13	0.395	13	0.392	13
	6 x 6	0.385	13	0.400	13	0.394	13	0.396	13
	9 x 9	0.393	13	0.406	13	0.398	13	0.397	13

Le formalisme utilisé ici pour estimer le rayon spectral et le nombre d'itérations est présenté page 58 (section 2.5). Les résultats des paragraphes 2.5 et 3.2.3 ont été reportés dans ce tableau. Ces résultats nous conduisent à formuler deux observations :

- Le rayon spectral des algorithmes accélérés par le CMFD est très nettement inférieur aux algorithmes non accélérée. De ce fait, la problématique de fausse convergence évoquée à la section 2.5 pour l'algorithme PMBJ non accéléré est résolue grâce à l'accélération.
- Le rayon spectral de l'algorithme PMBJ est supérieur aux rayons spectraux des algorithmes PMBG-S (KBA et Red-Black) ainsi que de celui de l'algorithme Direct. Cependant l'écart est réduit d'un ordre de grandeur.

Ces deux observations font que le nombre d'itérations externes estimé pour chaque algorithme accéléré est identique. De plus, le nombre d'itération est stable par rapport à l'augmentation de la taille du domaine de calcul pour l'algorithme PMBJ. D'après ces résultats, le CMFD semble être un très bon moyen d'accélérer notre algorithme.

Maintenant que le nombre d'itérations est stable quelle que soit l'algorithme, Direct ou PMBJ, il est possible d'envisager une accélération idéale en parallélisant le calcul. Dans la suite de ce travail, nous choisissons l'algorithme PMBJ puisqu'il permet d'accéder à une parallélisation maximale.

3.3.4.b Etude d'optimisation du maillage CMFD et évaluation de la mémoire requise

Comme nous venons de le voir, l'accélération par le CMFD à l'échelle du domaine global est une pierre angulaire de la méthode de décomposition de domaine présentée ici. Le nombre d'itérations externes est très fortement réduit dès lors que l'accélération est utilisée. Cependant il est important de veiller à ce que le coût de l'accélération ait le moins d'impact possible sur le temps de calcul. En effet, le temps de calcul d'une itération externe peut être exprimé de la manière suivante :

$$T_{ext} = T_{dom} + T_{glob}$$

où T_{dom} est le temps de calcul des tâches à effectuer dans les sous domaines et T_{glob} le temps de calcul des tâches à effectuer au niveau du calcul global. D'après l'Algorithme 3.5 (PMBJ accéléré par le CMFD), toutes les tâches sont réparties sur les sous domaines à l'exception de la construction de l'opérateur du CMFD et sa résolution. T_{dom} est ainsi parfaitement parallélisable, contrairement à T_{glob}. Le poids de l'accélération sera alors d'autant plus important que le domaine de calcul sera grand. C'est pourquoi il est essentiel d'optimiser la résolution du CMFD.

La performance du CMFD est analysée selon 3 axes détaillés dans la suite de cette section. Le premier est le coût en mémoire, le second est le coût en temps de calcul et le troisième est l'efficacité du CMFD à accélérer la convergence du problème (réduction du nombre d'itérations externes). Une particularité du CMFD développé pour la DD présentée ici est la possibilité de pouvoir modifier le maillage spatial et énergétique. La discrétisation du CMFD pouvant être choisie de plusieurs manières, c'est sur l'optimisation du maillage spatial et énergétique du CMFD que porte cette section. Nous pouvons noter que cette analyse peut aussi être appliquée à l'algorithme Direct.

COUT EN MEMOIRE

L'inventaire de la mémoire allouée pour le CMFD est décrit dans le Tableau 3.5 pour des maillages en 2 et 3 dimensions. N_G et N_R désignent toujours le nombre de macro groupes et de mailles grossières et N_X, N_Y, N_Z désignent le nombre de mailles respectivement suivant les directions X, Y et Z de l'espace.

Tableau 3.5. Dimension des tableaux FORTRAN alloués pour le CMFD en 2 et 3 dimensions.

Elément	Dimension en 2D	Dimension en 3D
Flux volumique	$N_G \times N_R$	$N_G \times N_R$
Flux surfacique	$N_G \times (2N_R + N_X + N_Y)$	$N_G \times (3N_R + N_Y N_Z + N_X N_Z + N_X N_Y)$
Courants	$N_G \times (2N_R + N_X + N_Y)$	$N_G \times (3N_R + N_Y N_Z + N_X N_Z + N_X N_Y)$
Section totale	$N_G \times N_R$	$N_G \times N_R$
Matrice de fission	${N_G}^2 \times N_R$	${N_G}^2 \times N_{\mathrm{R}}$
Matrice de transfert	${N_G}^2 \times N_R$	${N_G}^2 \times N_R$
Coefficients de la matrice	$5 \times N_G \times N_R$	$7 \times N_G \times N_R$

Les composantes de l'occupation mémoire sont les suivantes :

- Celle du flux volumique est proportionnelle au nombre de régions et au nombre de groupes.
- Celles des flux surfaciques et les courants sont proportionnelles au nombre de groupes et au nombre de surfaces. Ce dernier est du même ordre de grandeur que le nombre de régions.
- Celle de la section efficace totale est proportionnelle au nombre de groupes et de régions.
- Celles des matrices de fission et de transfert sont proportionnelles au carré du nombre de groupes et au nombre de régions.
- Celles des coefficients de la matrice pour la résolution spatiale du CMFD dépendent linéairement du nombre de groupes et de régions.

Afin d'illustrer la quantité de mémoire associée à un calcul CMFD, la Figure 3.15 présente l'occupation mémoire pour diverses discrétisations du maillage CMFD d'un assemblage de type REP en 2D :

- Le nombre de mailles énergétiques varie de 1 groupe (homogénéisation maximale) à 281 groupes (maillage couramment utilisé pour les calculs transports de référence).
- Le nombre de mailles spatiales varie de 1 maille spatiale (assemblage entièrement homogénéisé) à 289 mailles (assemblage homogénéisé crayon par crayon).

On peut observer l'effet des différents maillages possibles sur la mémoire en isolant certaines configurations :

- Pour le maillage le plus fin, l'occupation mémoire est de 186 Mo.
- Pour le couple $\{N_G = 281; N_R = 25\}$, l'occupation mémoire est de 16 Mo.
- Pour le couple $\{N_G = 26; N_R = 289\}$, l'occupation mémoire est de 2 Mo.

L'effet de la condensation énergétique est plus important que celui de l'homogénéisation spatiale. Ceci est dû au fait que toutes les composantes de l'occupation mémoire ont une taille d'ordre $o(N_R)$ par rapport au nombre de régions alors que les matrices de fission et de transfert ont une taille d'ordre $o({N_G}^2)$ par rapport au nombre de groupes.

Pour un couple $\{N_G = 26; N_R = 64\}$, l'occupation mémoire est de 0.4 Mo. A titre de comparaison, le stockage du flux transport pour un calcul d'assemblage en 2D est de l'ordre de 17 Mo (évaluation pour 3 moments spatiaux, 3 moments angulaires (anisotropie P_1) et 281 groupes d'énergie). L'opportunité de condenser/homogénéiser permet de rendre négligeable l'occupation mémoire du CMFD par rapport au calcul transport.

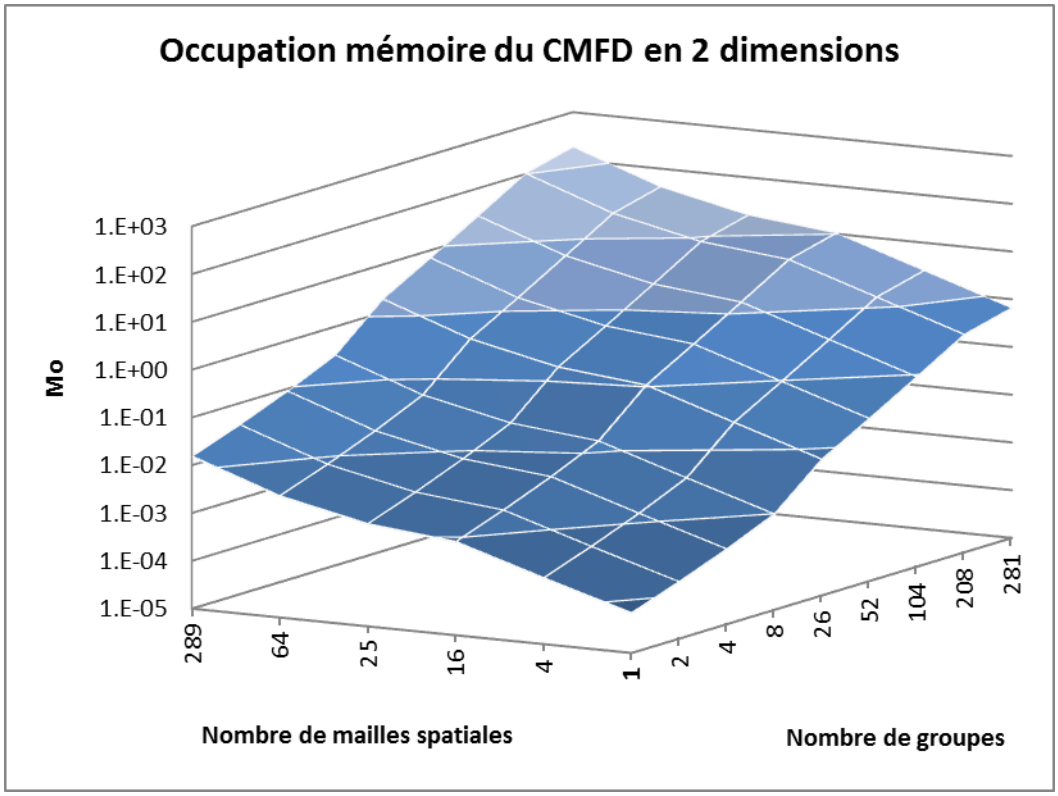


Figure 3.15: Occupation mémoire du CMFD (2D) en fonction du nombre de mailles spatiales et du nombre de groupes.

En 3D, la Figure 3.16 prend l'exemple d'un cube (nombre de mailles identique dans chaque direction). Concrètement, la quantité de mémoire est multipliée par le nombre de plan par rapport au calcul 2D, la majeure partie de la mémoire étant due aux matrices de transfert et de fission. Une discrétisation uniforme dans les 3 directions n'est pas nécessairement représentative du maillage que l'on pourrait appliquer pour un calcul de cœur en 3D. Cependant, supposant une discrétisation axiale avec environ 50 plans pour décrire la hauteur d'un assemblage, la taille du problème CMFD serait alors de l'ordre de 9 Go $(N_G = 289; N_R = 289 \times 50)$. La possibilité d'avoir un maillage (très) grossier est un avantage pour traiter des problèmes de taille importante.

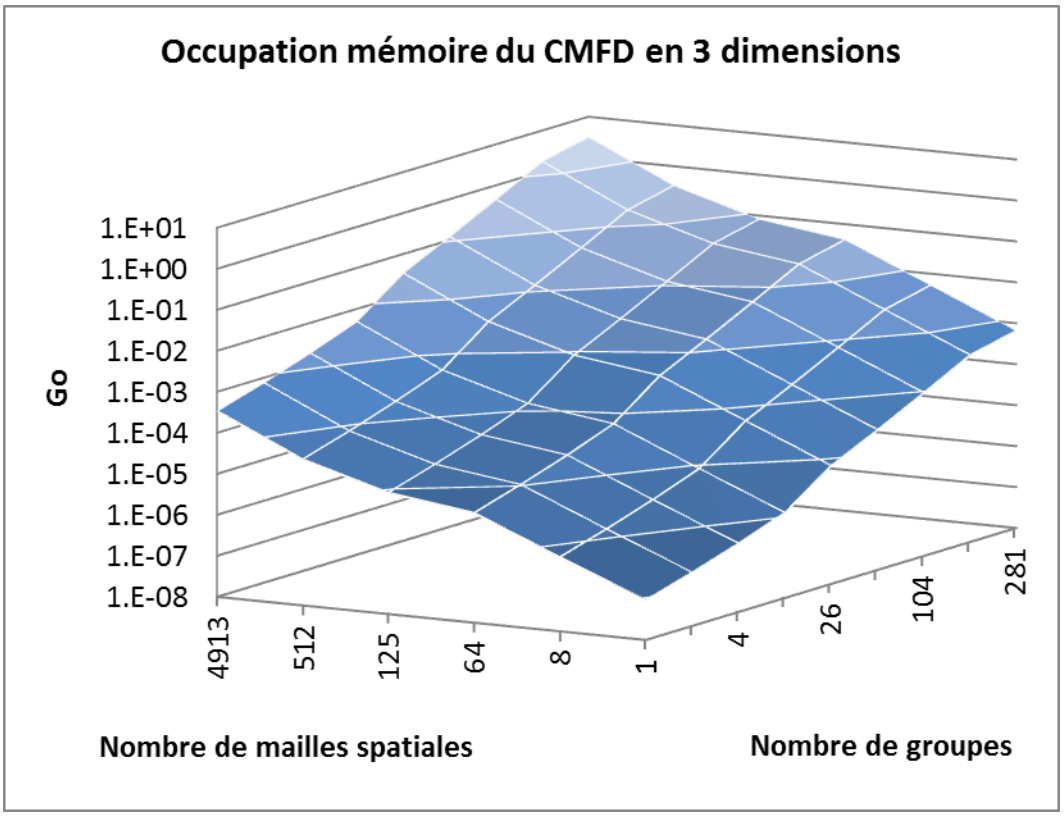


Figure 3.16. Occupation mémoire du CMFD (cube 3D) en fonction du nombre de mailles spatiales et du nombre de groupes.

Pour mettre en évidence les effets du maillage CMFD sur l'optimisation de l'accélération, nous avons repris le cas test décrit au paragraphe 2.5. Le motif est composé de 36 assemblages homogénéisés cellule par cellule. Les calculs sont réalisés avec des sections efficaces macroscopiques à 281 groupes jusqu'à l'ordre P1 pour l'anisotropie. Nous utilisons un développement spatial linéaire et 40 directions. Les critères de convergence sont fixés à 10^{-5} pour la valeur propre et le flux, 10^{-4} pour la source de fission et 10^{-4} pour les courants d'interface. Nous avons réalisé le calcul de ce motif avec différentes discrétisations du maillage CMFD en faisant varier le nombre de mailles spatiales par sous domaine de 1 (assemblage homogène) à 289 (crayon par crayon) et le nombre de groupes du CMFD de 1 à 281. La décomposition de domaine est appliquée à l'échelle de l'assemblage (un sous domaine par assemblage).

TEMPS DE CALCUL DU CMFD

Le nombre de mailles spatiales et énergétiques n'impacte pas seulement l'occupation mémoire du calcul. Le nombre d'inconnues du CMFD est lui aussi dépendant du maillage utilisé et contraint ainsi la résolution du CMFD (nombre d'itérations pour converger, nombre d'opérations par itération).

Lorsque le calcul des sous domaines est parallélisé, la résolution du CMFD constitue un goulot d'étranglement : toutes les tâches devront attendre que l'accélération soit terminée. La loi d'Amdahl donne le speedup maximal de l'accélération de l'algorithme parallélisé par :

$$S_{max} = \frac{1}{(1-s)}$$

Où s est dénote la fraction du temps séquentiel parallélisable. Considérant $(1-s)$ comme la fraction du temps dans le CMFD, c'est-à-dire non-parallélisable, il est essentiel de la limiter pour que l'algorithme soit scalable.

La fraction du temps dédié au CMFD est représenté Figure 3.17. Elle comprend les opérations d'homogénéisation, de calcul des matrices et de résolution. La part du CMFD dans le temps de calcul varie de 6% du temps total ($N_G \leq 26$; $(N_R/assemblage) = 1$) et jusqu'à 26 groupes d'énergie à 73% du temps total avec le maillage le plus fin ($N_G = 281$; $(N_R/assemblage) = 289$).

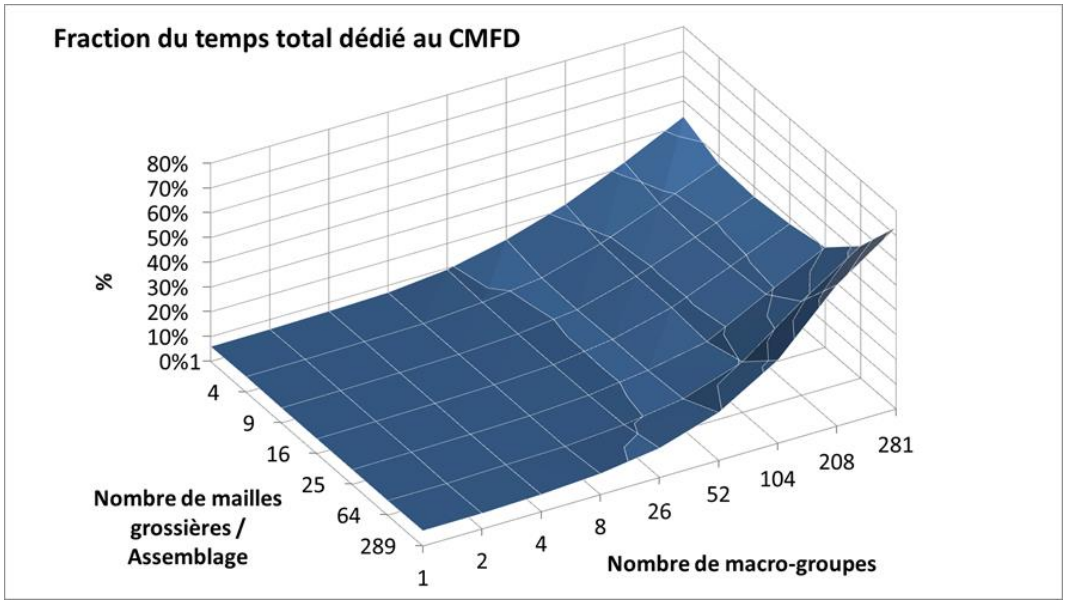


Figure 3.17: Etude du maillage CMFD : fraction du temps passé dans l'accélération par le CMFD en fonction du nombre de mailles spatiales et de groupes d'énergie.

La fraction dédiée au CMFD est réduite à 1.6% lorsque l'on ne comptabilise que le temps dédié à la résolution du système, pour le cas le plus favorable. Du point de vue de la parallélisation, le speedup maximum atteignable est donc de $S_{max} \approx 60$. Nous savons donc dès à présent que la parallélisation de la résolution du CMFD sera nécessaire pour étendre la scalabilité de la DDM que nous avons développée.

Le speedup maximal calculé ici ne rend compte que du potentiel de parallélisation. Cependant, l'efficacité du CMFD pour accélérer la convergence n'est pas mise en évidence. Nous étudions maintenant cet aspect.

EFFICACITE DE L'ACCELERATION

La Figure 3.18 présente le nombre d'itérations externes nécessaires pour atteindre la convergence dans chacun des cas étudiés. Nous séparons l'analyse de cette figure selon 2 axes :

- Le premier est la dépendance au nombre de mailles par sous domaine. Pour les calculs avec une seule région par sous domaine, le nombre moyen d'itérations est de 66 alors que pour le maillage le plus fin, il est de 12 (données obtenues en moyennant les résultats de toutes les configurations énergétiques). Le nombre d'itérations diminue fortement lorsque le maillage spatial est raffiné. L'accélération est plus efficace lorsque le maillage spatial du CMFD est le plus proche du maillage transport.
- Le second est la dépendance au nombre de groupes. Le nombre moyen d'itérations est de 30 pour les calculs à deux groupes et 25 avec 281 groupes (données obtenues en moyennant les résultats de toutes les configurations spatiales). Le nombre de groupes influe peu sur le nombre total d'itération.

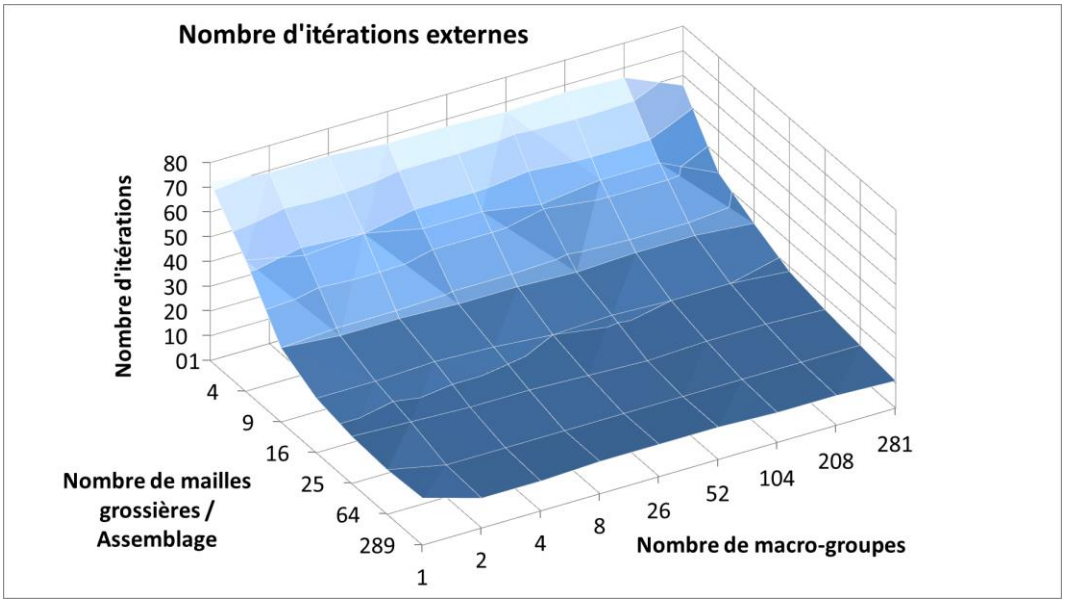


Figure 3.18: Etude du maillage CMFD : Nombre d'itérations externes en fonction du nombre de mailles spatiale et de groupes d'énergie.

Le nombre minimum d'itérations externes est obtenu pour le cas ayant la discrétisation la plus fine : 289 mailles par assemblage et 281 groupes. L'accélération par le CMFD est plus efficace lorsque sa discrétisation spatiale et énergétique coïncide avec le maillage transport. Lorsque le maillage spatial du CMFD est très grossier, l'accélération ne permet pas de prendre en compte les variations locales du flux. L'efficacité de l'accélération est alors amoindrie.

CONCLUSION

Jusque-là nous n'avons pas prise en compte le temps total du calcul. En effet, ce dernier est le paramètre important pour mesurer l'efficacité. La Figure 3.19 présente le temps du calcul pour les différents maillages de CMFD testés. Le temps maximal est

de plus de 10 000 secondes avec 1 région par assemblage et 281 groupes alors que le temps minimal est de 1 250 secondes avec 289 régions par assemblages et 8 groupes. Deux tendances générales sont observées :

- le temps de calcul diminue quand le nombre de mailles spatiales augmente,
- le temps de calcule augmente quand le nombre de groupes augmente.

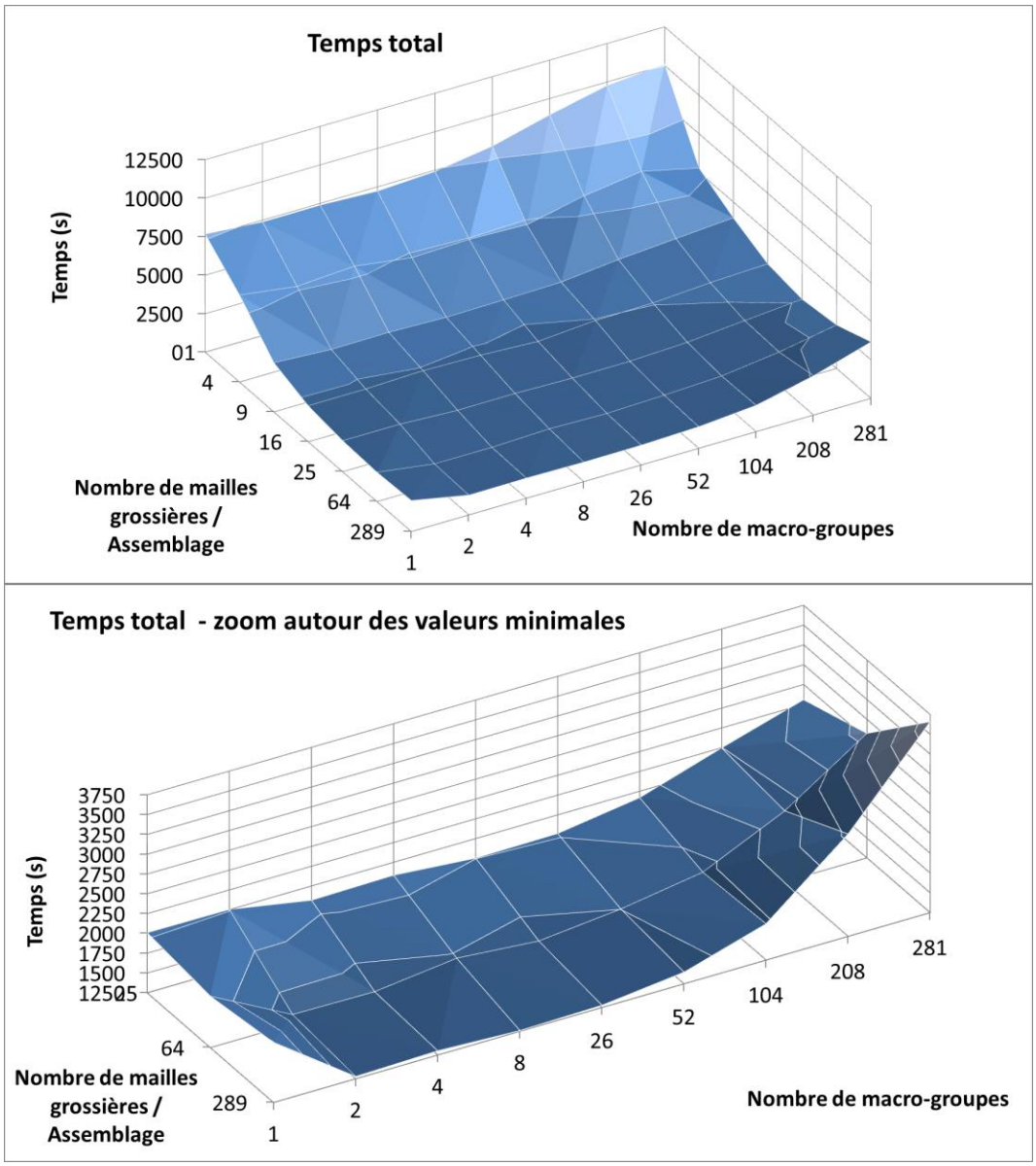


Figure 3.19: Etude du maillage CMFD : Temps de calcul en secondes en fonction du nombre de mailles spatiale et de groupes d'énergie.

Ce phénomène s'explique par deux raisons. Tout d'abord, nous avons vu que lorsque la discrétisation spatiale et la discrétisation énergétique du CMFD sont raffinées, le nombre d'itérations externes diminue, ce qui explique que le temps de calcul est réduit pour les maillages les plus fins (Figure 3.18). Cependant, le raffinement énergétique a un effet faible par rapport au raffinement spatial sur le nombre d'itérations. Il en découle qu'avec un nombre de groupes élevé, le temps passé dans l'accélération

augmente sans réduire notablement le nombre d'itérations externes. Ceci explique l'augmentation du temps de calcul avec le nombre de groupes.

De cette étude paramétrique du maillage du CMFD ressortent plusieurs enseignements. Tout d'abord, il apparait que quel que soit le maillage choisi, le calcul converge, même avec les maillages les plus grossiers. Le maillage optimal est un compromis entre coût en mémoire, coût en temps de calcul pour la résolution du CMFD. **Nous avons illustré le fait qu'une discrétisation énergétique fine est couteuse pour la mémoire et le temps de calcul alors qu'elle n'apporte pas de gain signifiant en termes d'accélération de la convergence. Au contraire, une discrétisation spatiale fine est essentielle : plus le maillage spatial du CMFD est proche de celui du transport, plus l'accélération est efficace.** Le nombre d'itérations est réduit d'un facteur 5 entre l'assemblage homogénéisé et le maillage crayon par crayon. **En résumé : peu de groupe avec beaucoup de régions permet d'accélérer efficacement.**

3.4 Résolution des sous domaines avec des Itérations Multigroupes Locales ou des Itérations de Puissance Locales

L'équation du problème monocinétique dans un sous domaine est un problème à source. Ces sources sont d'une part surfaciques, i.e. le flux angulaire entrant aux interfaces avec d'autres sous domaines, et d'autre part volumiques, composées de la source de scattering multigroupe et de la source de fission. La convergence de ces sources requiert plusieurs itérations multigroupes dues à l'up-scattering et à la source de fission qui génère des neutrons dans les groupes rapides à partir des groupes thermiques. L'objet de cette section est d'accélérer la convergence en itérant localement sur les sources de scattering et de fission. Nous présentons ici deux méthodes que nous avons expérimentées.

Pour rappel, dans l'Algorithme 2.3 présenté initialement, le nouveau flux est obtenu en inversant le problème multigroupe local pour lequel la source de fission est calculée à l'itération externe précédente :

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+1)}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha^-}^{(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

Un Simple Balayage Multigroupe (SBM) est réalisé avec éventuellement des itérations thermiques. Une autre possibilité consiste à itérer localement sur la source de fission sans mettre à jour les flux aux interfaces entre les sous domaines :

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(l+1)}(x) = {Q_\alpha}^{(l)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha^-}^{(e+1)}(x), & x \in \Gamma_\alpha^-, \end{cases}$$

où l est l'indice de l'itération locale sur la source de fission. Le nouveau flux est alors obtenu par :

$${\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha}^{(l=\infty)}(x).$$

Nous proposons ici deux algorithmes pour lesquels la source de fission ${Q_\alpha}^{(l)}(x)$ est calculée itérativement.

ITERATIONS MULTIGROUPES LOCALES (IML)

Dans le premier algorithme, la source de fission est calculée en normalisant par la valeur propre globale :

$${Q_\alpha}^{(l+1)}(x) = \frac{{(F\psi)_\alpha}^{(l+1)}(x)}{\lambda^{(e)}},$$

avec l'initialisation :

$${Q_\alpha}^{(l=0)}(x) = \frac{{(F\psi)_\alpha}^{(e)}(x)}{\lambda^{(e)}}.$$

Cette méthode est présentée dans l'Algorithme 3.6. A la fin de chaque Itération Multigroupe Locale (IML) la source de fission est mise à jour et normalisée par la valeur propre globale $\lambda^{(e)}$ calculée à l'itération externe précédente. Cet algorithme présente l'avantage de mieux converger la source multigroupe dans les sous domaines.

Algorithme 3.6. Calcul des sous domaines par Itérations Multigroupes Locales (IML).

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha \; \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux aux interfaces entre sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha : \; x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha : \; (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A$, ***faire***

Initialisation de la source de fission locale

$Q_\alpha{}^{(l=0)}(x) = Q_\alpha{}^{(e)}(x)$

Itération multigroupes locales

Tant que Q_α *n'est pas convergée,* ***faire***

$$\begin{cases} (L-H)_\alpha \psi_\alpha{}^{(l+1)}(x) = Q_\alpha{}^{(l)}(x), & x \in X_\alpha, \\ \psi_\alpha{}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

$$Q_\alpha{}^{(l+1)}(x) = \frac{(F\psi)_\alpha{}^{(l+1)}(x)}{\lambda^{(e)}}$$

fin

$\psi_\alpha{}^{(e+1)}(x) = \psi_\alpha{}^{(l=\infty)}(x).$

fin

Calcul de la nouvelle source de fission normalisée et de la valeur propre avec ou sans l'accélération par le CMFD : $\lambda^{(e+1)}, Q_\alpha{}^{(e+1)}(x).$

fin

ITERATIONS DE PUISSANCE LOCALES (IPL)

Pour le second algorithme, le problème à résoudre dans chaque sous domaine est posé comme un problème à « valeur propre ». La source de fission calculée localement est normalisée par une valeur propre locale :

$$\begin{cases} \lambda_\alpha^{(l+1)} = \lambda_\alpha^{(l)} \dfrac{\langle \boldsymbol{F}_\alpha{}^{(l)}, \boldsymbol{F}_\alpha \psi^{(l+1)} \rangle}{\langle \boldsymbol{F}_\alpha{}^{(l)}, \boldsymbol{F}_\alpha \psi^{(l)} \rangle}, \\ Q_\alpha{}^{(l+1)}(x) = \dfrac{(F\psi)_\alpha{}^{(l+1)}(x)}{\lambda_\alpha^{(l+1)}}, \end{cases}$$

avec l'initialisation :

$$\begin{cases} \lambda_\alpha^{(l=0)} = \lambda^{(\mathrm{e})}, \\ Q_\alpha{}^{(l=0)}(x) = Q_\alpha{}^{(e)}(x). \end{cases}$$

Ce problème n'est pas à proprement parler un problème à valeur propre puisque les conditions aux limites du problème ne sont pas homogènes. Cependant, à convergence, la valeur propre doit être la même en tout point du domaine de calcul. C'est-à-dire $\lambda_\alpha = \lambda$ lorsque le calcul est convergé. Le calcul local de la valeur propre permet alors d'éviter la synchronisation entre tous les calculs de sous domaines. Cet algorithme, avec ce que nous nommons des Itérations de Puissance Locales (IPL), est présenté par l'Algorithme 3.7.

Algorithme 3.7. Calcul des sous domaines par Itérations de Puissance Locales (IPL).

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha \; \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Mise à jour des flux aux interfaces entre sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha : \; x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha : \; (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A$, ***faire***

Initialisation de la source de fission locale

$$\begin{cases} \lambda_\alpha^{(l=0)} = \lambda^{(e)}, \\ Q_\alpha{}^{(l=0)}(x) = Q_\alpha{}^{(e)}(x). \end{cases}$$

Itération multigroupes locales

Tant que Q_α, λ_α *ne sont pas convergés,* ***faire***

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(l+1)}(x) = \dfrac{F_\alpha {\psi_\alpha}^{(l)}(x)}{\lambda_\alpha^{(l)}}, & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha^-}^{(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

$$\begin{cases} \lambda_\alpha^{(l+1)} = \lambda_\alpha^{(l)} \dfrac{\langle {\boldsymbol{F}_\alpha}^{(l)}, {\boldsymbol{F}_\alpha}^{(l+1)} \rangle}{\langle {\boldsymbol{F}_\alpha}^{(l)}, {\boldsymbol{F}_\alpha}^{(l)} \rangle}, \\ {Q_\alpha}^{(l+1)}(x) = \dfrac{{(F\psi)_\alpha}^{(l+1)}(x)}{\lambda_\alpha^{(l+1)}}, \end{cases}$$

fin

$${\psi_\alpha}^{(e+1)}(x) = {\psi_\alpha}^{(l=\infty)}(x).$$

fin

Calcul de la nouvelle source de fission normalisée et de la valeur propre avec ou sans l'accélération par le CMFD : ${\lambda}^{(e+1)}, {Q_\alpha}^{(e+1)}(x)$.

fin

Ces méthodes visent à favoriser la convergence des sources de scattering et de fission par rapport à la propagation de la solution entre les sous domaines (échanges des flux aux interfaces des sous domaines). En effet, le nombre d'itérations multigroupes total est plus important que le nombre d'échanges de flux entre les sous domaines.

Dans l'implémentation réalisée, le nombre d'itérations multigroupes locales peut être limité. Ce nombre est choisi par l'utilisateur. Lorsque le critère de convergence sur la valeur propre globale est atteint pour la première fois, le nombre d'itérations multigroupes locales est doublé de manière à favoriser d'avantage la convergence des sources de scattering et de fission par rapport aux échanges entre les sous domaines. Finalement, lorsque le critère de convergence sur la source de fission globale est atteint pour la première fois, le nombre d'itérations multigroupes locales est fixé à 1, de manière à favoriser les échanges entre les sous domaines.

3.4.1 Etude de l'influence du nombre d'itération multigroupes locales sur un colorset 3 x 3

Nous étudions ici l'impact du nombre d'itérations multigroupes locales pour les méthodes avec IML et IPL (respectivement Algorithme 3.6 et Algorithme 3.7) sur la convergence du DDM. Le cas test utilisé ici est le colorset d'assemblages en milieu infini de type REL décrit en Annexe A. Le motif est composé de 9 assemblages UOx (homogénéisation cellule par cellule) avec une barre d'absorbant B4C dans l'assemblage central soit 2601 régions. Les calculs sont réalisés avec des sections efficaces macroscopiques à 281 groupes jusqu'à l'ordre P1 pour l'anisotropie. Nous utilisons un développement spatial linéaire et 40 directions. Les critères de convergence sont fixés à 10^{-5} pour la valeur propre, 10^{-4} pour la source de fission, 10^{-5}

pour le flux et 10^{-4} pour les courants d'interface. Les calculs sont accélérés par le CMFD à l'échelle globale mais aussi pour les itérations internes dans les sous domaines. Le maillage du CMFD est composé de 64 régions par sous domaine et 26 groupes d'énergie. Un sous domaine est associé à chaque assemblage.

ETUDE DE L'INFLUENCE DU NOMBRE D'ITERATIONS LOCALES.

Nous avons étudié la convergence de l'erreur sur les courants d'interface et des sources de fission avec l'Algorithme 3.7 (Calcul des sous domaines par Itérations de Puissance Locales (IPL)). Ce calcul a été réalisé avec différentes limites du nombre d'itérations multigroupes : 1, 2, 5, 8, 10, 15 et 20. Le Tableau 3.6 présente le nombre d'itérations externes à convergence selon les différentes limites du nombre d'itérations multigroupes locales.

Tableau 3.6. Nombre d'itérations externes en fonction de la limite du nombre d'itérations de puissances locales (IPL).

Nombre limite d'itérations locales (IPL)	1	2	5	8	10	15	20
Nombre d'itérations externes atteint	11	9	8	9	9	9	9

Lorsque la limite est une seule itération, le nombre d'itérations externes est 11. Pour les autres cas, le problème converge en 8 ou 9 itérations. Quand le nombre d'itérations multigroupes locales est supérieur à 5, le nombre d'itérations externes reste constant.

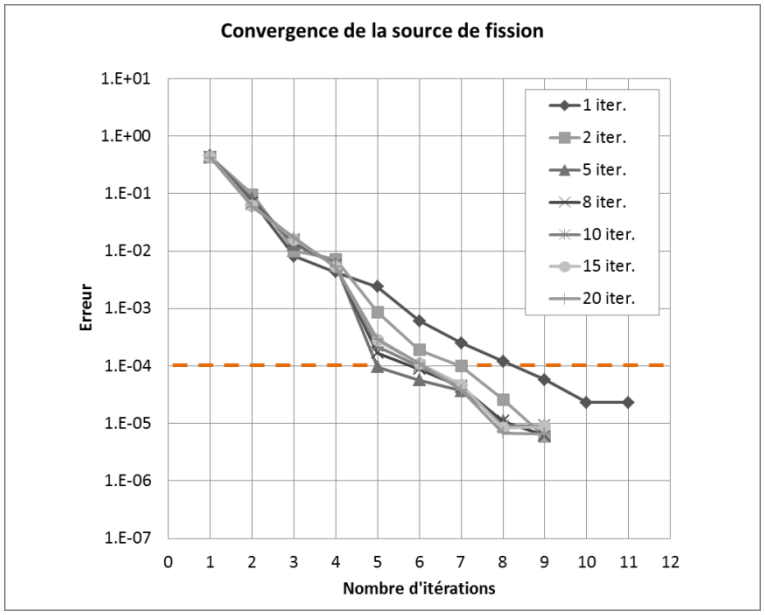


Figure 3.20. Variation de l'erreur sur la source de fission pour différentes limites du nombre d'itérations multigroupes locales.

La Figure 3.20 et la Figure 3.21 présentent respectivement la variation de l'erreur sur les sources de fission et celle sur les courants d'interface par rapport aux itérations externes. L'erreur décroît plus vite lorsque le nombre d'itérations locales augmente, en cohérence avec les résultats du Tableau 3.6. On observe aussi un phénomène de saturation : la vitesse de convergence est la même quelle que soit la limite au-delà de 5

itérations multigroupes locales. Ce phénomène est observable à la fois pour l'erreur sur la source de fission et l'erreur sur les courants d'interface.

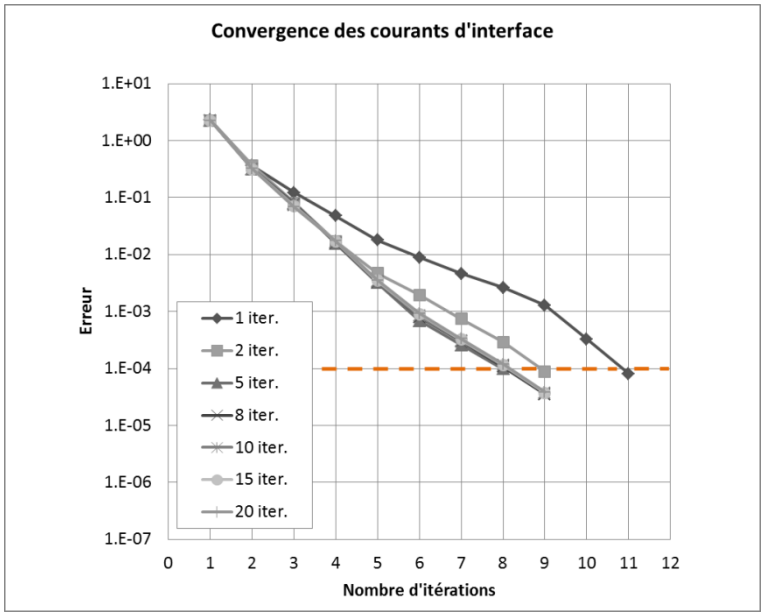


Figure 3.21. Variation de l'erreur sur les courants d'interface pour différentes limites du nombre d'itérations multigroupes locales.

Réaliser des itérations multigroupes locales permet donc d'accélérer la convergence de la source multigroupe mais aussi celle des courants d'interface : le flux angulaire sortant étant alors mieux calculé. Le phénomène de saturation s'explique par le fait que les flux d'interface (qui ne sont pas mis à jour) entrainent une erreur sur la source multigroupe. Cette erreur devient alors prédominante après plusieurs itérations multigroupes locales.

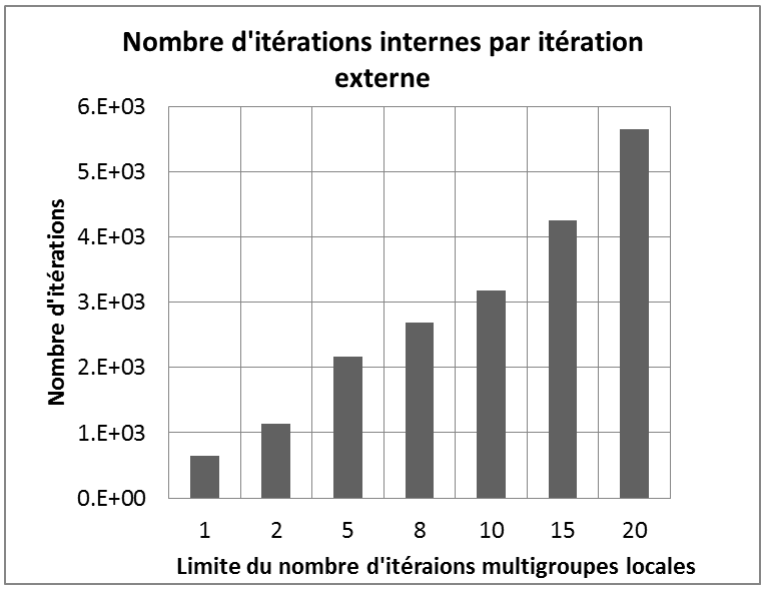


Figure 3.22. Nombre d'itérations internes par itération externe pour différentes limites d'itérations multigroupes locales.

Le temps de calcul est cependant le paramètre le plus important pour mesurer l'efficacité. Pour un calcul transport multigroupe, il est principalement consommé

dans les itérations internes du transport et dans le calcul des sources de scattering. La Figure 3.22 illustre le nombre total d'itérations internes en fonction du nombre d'itérations multigroupes locales.

Le nombre d'itérations internes croit avec le nombre d'itérations multigroupes : il passe de 640 pour le cas limité à 1 itération multigroupe, à plus de 5600 lorsque la limite est de 20 itérations, soit plus d'un facteur 8. Pour le cas limité à 2 itérations locales par externes, le nombre d'itérations internes ne croit que d'un facteur 1.8. Réaliser plusieurs itérations multigroupes locales par itérations externe peut s'avérer coûteux en termes de temps de calcul comme le montre la Figure 3.23.

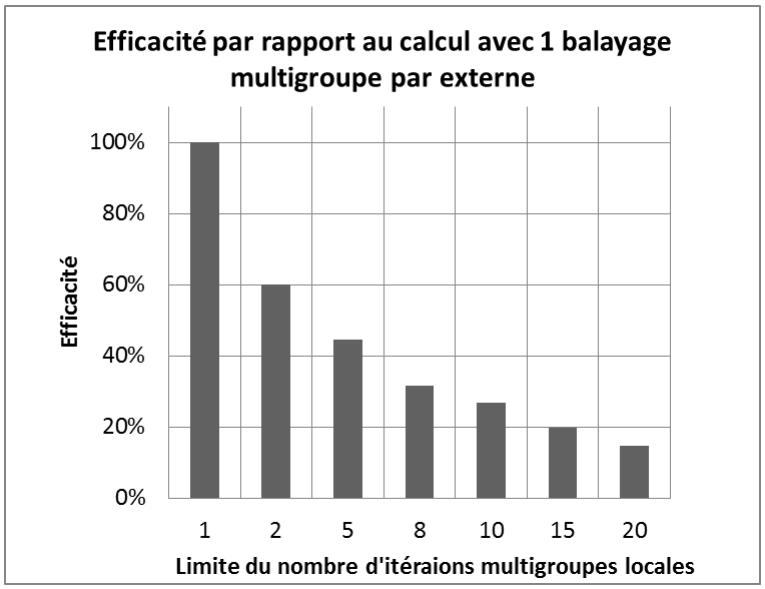


Figure 3.23. Efficacité des itérations multigroupe locales multiples sur le temps de calcul.

L'efficacité, définie comme le rapport $T(1\ iter.)/T(N\ iter.)$, décroit dès que plusieurs itérations multigroupes locales sont réalisées à cause du coût du balayage transport dans les itérations internes. Une solution envisagée est d'accélérer les itérations multigroupes locales par un CMFD multigroupe défini lui aussi localement. De cette manière la convergence des sources de fission est accélérée sans le coût du calcul transport. Nous n'avons pas eu l'opportunité de tester cette méthode d'accélération des itérations multigroupes locales.

Pour conclure, Le nombre d'itérations externes est réduit en réalisant plusieurs itérations multigroupes locales. Ce gain sature au-delà de quelques itérations multigroupes locales, l'erreur sur le flux d'interface devenant prédominant. L'Algorithme 3.7 bénéficie d'un meilleur ordre de convergence lorsque plusieurs itérations multigroupes locales sont réalisées. Cependant, dans l'implémentation actuelle, l'augmentation du nombre d'itérations multigroupes entraine l'augmentation du nombre de balayages transport. Le temps de calcul est alors dégradé. Cette étude a aussi été menée avec l'Algorithme 3.6 et les résultats sont similaires. L'implémentation actuelle ne permet pas de bénéficier de l'avantage donné par cette amélioration de l'algorithme en termes de temps de calcul.

INFLUENCE DU NOMBRE DE SOUS DOMAINES

Afin de pouvoir mettre en évidence la stabilité des méthodes par Simple Balayage Multigroupe (SBM), Itérations Multigroupes Locales (IML) et Itérations de Puissance Locales (IPL) vis-à-vis de la taille du domaine de calcul, nous avons dupliqué le motif utilisé dans le test précédent en plusieurs motifs 6 x 6 et 9 x 9. On peut noter que le colorset 9 x 9 est constitué d'un nombre d'assemblage supérieur à celui d'un quart de grand cœur de REL. Le nombre limite d'itérations multigroupes locales est fixé à deux. Les tolérances et le CMFD sont définis de la même manière que précédemment.

La Figure 3.24 présente le temps de calcul total avec les méthodes SBM, IPL et IML pour les 3 tailles de colorset étudiées. Le nombre d'itérations externes est aussi précisé pour chaque point de calcul.

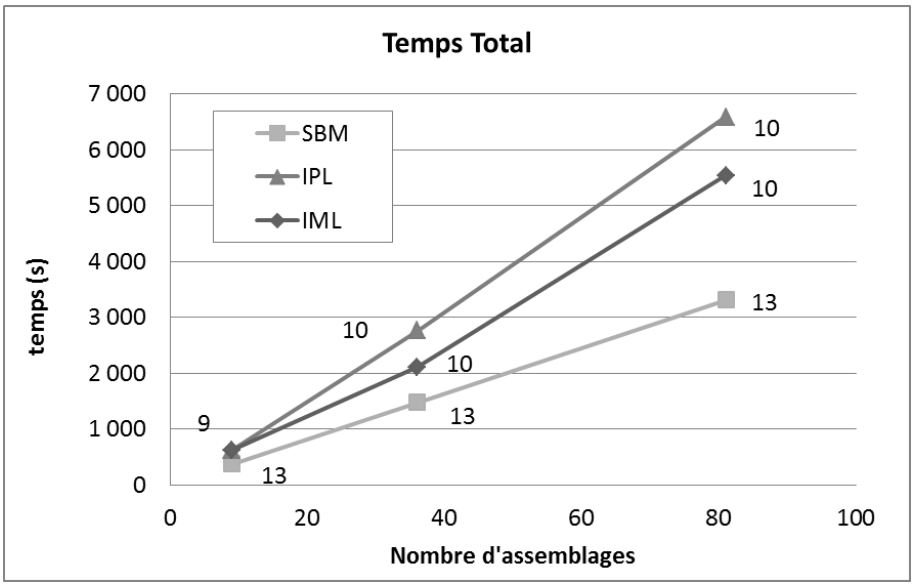


Figure 3.24. Influence de la taille du problème pour les méthodes SBM, IML et IPL.

Il apparait que quelle que soit la taille du problème à résoudre, la méthode SBM est toujours la plus rapide, et ce même si le nombre d'itérations externes est plus important (13 vs. 10). Ces observations sont similaires à celle du paragraphe précédent. Il est intéressant de noter que le nombre d'itérations n'augmente pas avec la taille du problème, ce qui n'aurait pas été le cas sans l'accélération par le CMFD.

En conclusion, les méthodes IPL et IML présentées ici permettent de réduire le nombre d'itérations externes par rapport à la méthode SBM initialement proposée. Cependant, dans l'implémentation actuelle, ces méthodes ne sont pas avantageuses en termes de temps de calcul. Une solution serait alors de définir un CMFD multigroupe local permettant d'accélérer la convergence des sources de fission sans le coût des itérations multigroupes du transport. Nous avons aussi illustré le fait que la taille du domaine de calcul n'a pas d'influence sur la convergence de ces trois méthodes.

3.5 Conclusion

Nous avons vu dans le chapitre précédent que le taux de convergence de l'algorithme PMBJ est plus faible que celui de l'algorithme Direct. De ce fait, l'espoir d'obtenir une scalabilité idéale est vain avec l'Algorithme 2.3 (algorithme PMBJ non accéléré) dû à l'augmentation du nombre d'itérations nécessaires pour atteindre la convergence. L'algorithme PMBJ entraine un retard dans la transmission de l'information à l'interface entre les sous domaines. L'objectif des méthodes présentées dans cette section est de modifier cet algorithme de manière à améliorer la transmission de l'information aux interfaces entre les sous domaines.

Nous avons présenté 3 méthodes visant à améliorer le rayon spectral de l'algorithme PMBJ :

- Tout d'abord, nous avons remis en question l'algorithme PMBJ dans la section 3.2 en proposant des algorithmes de type PMBG-S permettant de profiter des flux d'interface les plus récents avant le calcul de chaque sous domaine. Ces algorithmes ont permis de réduire le rayon spectral mais de manière trop peu significative pour permettre de se protéger des phénomènes de fausse convergence. De plus la dépendance du rayon spectral au nombre de sous domaines est toujours présente.
- Ensuite, nous avons proposé une méthode d'accélération non linéaire par la diffusion, le CMFD. Cette méthode accélère significativement la convergence de l'algorithme PMBJ, permettant d'obtenir un rayon spectral proche de celui de l'algorithme Direct, accéléré lui aussi par le CMFD. Dans ce cadre, nous avons vu que les algorithmes de type PMBG-S ne présentent pas d'avantage notable par rapport à l'algorithme PMBJ. La réduction significative du rayon spectral a permis de se protéger des phénomènes de fausse convergence. De plus, le rayon spectral des calculs accélérés par le CMFD est pratiquement insensible au nombre de sous domaines. Dans l'exemple d'application (un colorset d'assemblages), le nombre d'itérations estimé pour converger est le même quel que soit l'algorithme, pour toutes les tailles de domaines étudiées.
- Enfin, dans la section 3.4 nous avons modifié la méthode de calcul des sous domaines. Localement, plusieurs itérations sur la source de fission sont réalisées avant de recalculer la source de fission et la valeur propre globale. Ces méthodes permettent de réduire le nombre d'itérations externes pour converger. Cependant elles entrainent une augmentation du nombre de balayages transport multigroupes, pénalisant le temps de calcul final. De plus amples recherches sont à effectuer pour que ce type d'algorithme soit avantageux vis-à-vis de l'algorithme PMBJ avec un simple balayage multigroupe.

Ainsi, parmi les méthodes d'accélération proposées dans ce chapitre, nous retenons principalement l'algorithme PMBJ accéléré par le CMFD sans itérations multigroupes locales. C'est sur ce choix que porterons l'ensemble des travaux dans la suite de cette thèse. En effet, les sous domaines peuvent être calculés de manière complètement indépendante avec l'algorithme PMBJ, ce qui permet une parallélisation optimale. Le CMFD permet quant à lui d'obtenir un rayon

spectral similaire entre le calcul en DDM et le calcul Direct. Finalement, dans notre réalisation, n'effectuer qu'un seul balayage multigroupe par itération externe est le plus favorable.

Chapitre 4. Architecture de l'implémentation en maquette et méthodes de parallélisation utilisées

4.1 Introduction

Ce chapitre est composé de 3 parties :

- **La première sera la description de l'architecture de la réalisation**. On y présentera la manière dont on optimise la gestion de la mémoire et l'utilisation du solveur IDT, ce dernier étant utilisé pour réaliser les calculs transport ; nous verrons que nous n'avons pas eu besoin de modifier le solveur IDT pour implémenter l'algorithme PMBJ. Dans cette section, nous comparerons les performances en termes d'utilisation des processeurs entre l'algorithme Direct et l'algorithme PMBJ dans le cadre de l'application à un colorset d'assemblage REL. Cette section se terminera par une illustration de l'effet de la taille des sous domaines sur la convergence. Cette illustration permettra de démontrer la possibilité d'accélérer efficacement le calcul de grands motifs d'assemblages de REL.
- La seconde partie sera dédiée à la première voie de parallélisation que nous avons choisi : **le parallélisme à mémoire partagée avec la bibliothèque OpenMP**. Il permet de conserver les avantages de l'optimisation de la mémoire présentée dans la première partie. Ensuite nous aborderons les avantages de l'implémentation vis-à-vis du parallélisme à mémoire partagée : pas de conflits d'accès mémoire, parallélisme coarse grained facile à implémenter, coût des communications minimisé, équilibrage de charge dynamique et désynchronisation des tâches. Ces avantages seront illustrés par des tests de scalabilité (weak et strong scaling) sur une machine de bureau standard pour lesquels nous avons obtenu une scalabilité « idéale ».
- **Enfin, la dernière partie est dédiée à la distribution de la mémoire pour accéder au parallélisme à mémoire distribuée**. La motivation de cette implémentation est le manque de mémoire et de processeur dans les architectures à mémoire partagées classiques. Nous présenterons la méthode hybride MPI/OpenMP développée permettant de garder certains avantages du parallélisme à mémoire partagée. La solution consistera à allouer plusieurs sous domaines par processus MPI et à utiliser OpenMP pour paralléliser le calcul des sous domaines dans chaque processus. De cette manière la scalabilité idéale obtenue précédemment sera conservée. Nous verrons aussi que la topographie des processus MPI peut être quelconque, rendant complexe l'optimisation des communications des flux d'interface. L'algorithme de communication sera présenté. Bien que n'étant pas optimisé, nous verrons qu'il permet de supporter n'importe quelle topographie de processus MPI tout en réalisant un maximum de communication simultanément. Nous présenterons enfin des éléments permettant des améliorations.

4.2 Architecture de la réalisation

A la section 2.2.1, nous avons relevé que l'organisation des données influe fortement sur l'efficacité d'utilisation des processeurs (localité spatiale et temporelle des données) ainsi que sur la qualité d'une implémentation parallèle. Dans cette section, nous présentons l'architecture générale de la maquette dans laquelle nous avons implémenté l'algorithme PMBJ et comment cette architecture permet d'utiliser efficacement les processeurs. La maquette développée s'appuie sur le solveur IDT d'APOLLO3 pour laquelle une interface spécifique, limitée aux besoins de cette thèse, est mise en œuvre afin de permettre de réaliser la décomposition de domaine. Des mots clés sont utilisés pour décrire la géométrie, les options de maillage et les paramètres du calcul. Le terme *Maquette* ou *Maquette-DDM* sera utilisé dans la suite pour désigner le code dans lequel nous avons implémenté l'algorithme PMBJ accéléré par le CMFD.

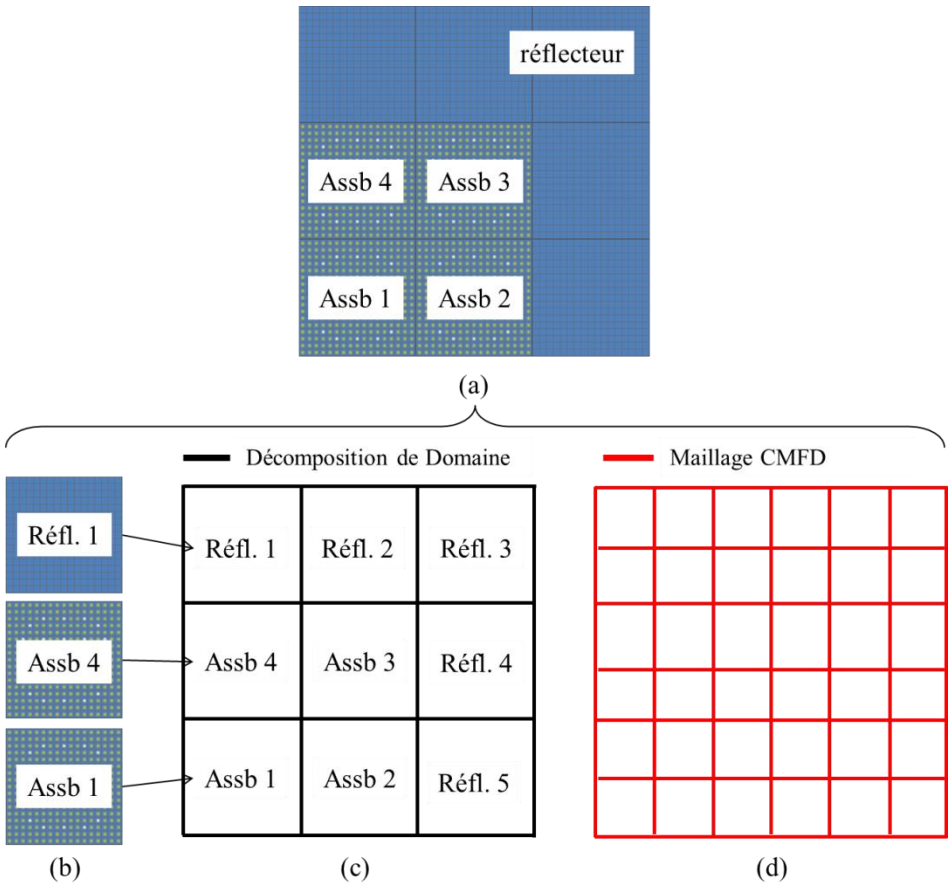


Figure 4.1. Les différents niveaux de discrétisation de la géométrie globale (a) pour le calcul DDM : Géométries transport des sous domaines (b), géométrie pivot (c) décrivant la discrétisation en sous domaines et le maillage CMFD (d).

La Figure 4.1 illustre la gestion de la décomposition de domaine via l'interface. La géométrie du domaine global est donnée Figure 4.1(a). Dans l'interface elle est décrite par la géométrie pivot présentée définissant les différents sous domaines : dans la Figure 4.1(c) la géométrie pivot définit 9 sous domaines. Les géométries physiques

des calculs transports (maillage et définition des matériaux) sont définies par sous domaine comme le montre la Figure 4.1(b) pour les 3 premiers sous domaines. Ces géométries sont ensuite assignées aux zones définies par la géométrie pivot. Le maillage du CMFD donné comme exemple à la Figure 4.1(b) est défini en prenant en compte à la fois la géométrie pivot et la géométrie transport des sous domaines. La définition du maillage CMFD, dont un exemple est donné en Figure 4.1(d), a été décrite à la section 3.3.3.a. Il s'agit d'un maillage conforme, défini par mots clés, inclus dans le maillage fin et contenant la géométrie pivot.

4.2.1 Gestion et distribution des données

ORGANISATION DE LA MEMOIRE POUR LE SOLVEUR IDT

Le solveur IDT est programmé en FORTRAN et organisé en modules. Chacun de ces modules a en charge une composante du solveur multigroupe. Par exemple, la quadrature angulaire, les harmoniques sphériques, la géométrie, les sections efficaces, le flux ou encore le maillage sont gérés par des modules différents. En ce qui concerne la gestion de la mémoire, nous porterons une attention particulière à 4 de ces modules. Le premier est le module qui prépare et contient les sections efficaces multigroupes (module XSL), le second contient le maillage spatial (module MSH), le troisième calcule et stocke les matrices des coefficients du transport (module COF) et le dernier est en charge de la gestion du flux, des courants et de l'intégrale de fission (module FLX). L'ensemble des modules permet d'effectuer un calcul multigroupe. Ils composent ce que l'on appelle ici une Unité de Calcul (UC). Parmi les modules, on peut distinguer deux catégories. La première catégorie est composée des modules contenant des données *non-mutables* et la seconde catégorie est composée de modules contenant des données *mutables*. Par exemple, les modules XSL, MSH et COF, sont initialisés avant la résolution du problème et ne sont plus modifiés au cours des itérations. Ces modules sont non-mutables. Au contraire, le module FLX contient le flux et l'intégrale de fission. Il est modifié pendant les itérations, c'est un module mutable. Nous verrons par la suite que cette distinction permettra d'optimiser la mémoire pour la décomposition de domaine.

ORGANISATION DE LA MEMOIRE POUR LE SOLVEUR DDM

L'implémentation réalisée repose sur le fait qu'une Unité de Calcul est associée à chaque sous domaine. Ainsi chaque sous domaine peut être considéré comme un solveur indépendant, avec la même structure de données que le solveur IDT standard. Cette organisation des données nous a permis de limiter au maximum les modifications du solveur existant. Les directives liées à la décomposition de domaine peuvent alors être vues comme une bibliothèque extérieure qui permet de coupler plusieurs solveurs.

La gestion mémoire de la DDM est basée sur la définition de deux types d'Unités de Calcul : les Unité de Calcul Génératrices (UCG) et des Unités de Calcul Effectives (UCE). Pour illustrer cette organisation de la mémoire, la Figure 4.2 prend l'exemple d'un colorset de 4 assemblages, composé de deux types d'assemblages différents :

l'assemblage type A et l'assemblage type B. Nous définissons la géométrie pivot de la décomposition de domaine au niveau des assemblages : un sous domaine correspond alors à un assemblage.

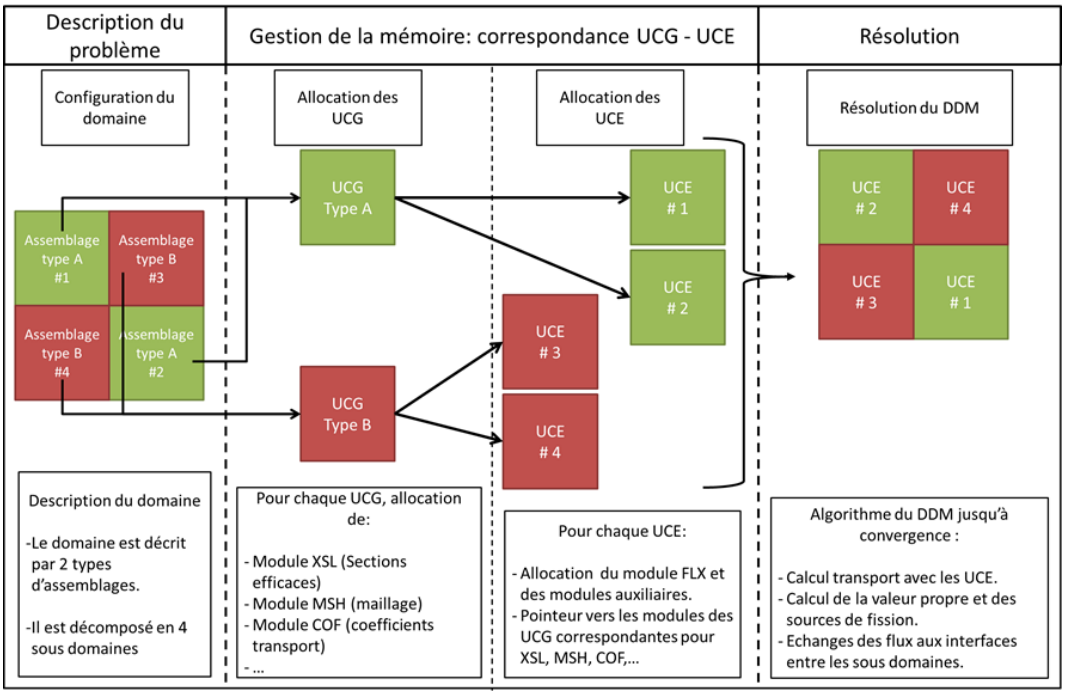


Figure 4.2. Gestion de la mémoire pour l'implémentation du DDM.

La distinction UCG/UCE permet d'optimiser l'utilisation de la mémoire en réalisant un typage des sous domaines. Deux sous domaines sont de même type s'ils ont la même description physique. En d'autres termes, ils sont de même type s'ils ont le même maillage, les mêmes matériaux et la même répartition de ces matériaux sur le maillage. Une UCG est définie par type de sous domaine. Dans l'exemple de la Figure 4.2, nous avons les UCG A et B correspondant respectivement aux assemblages (sous domaines) de types A et B. Les UCG ne contiennent que les données non-mutables des Unités de Calcul. L'UCG est en charge des modules COF, MSH, et XSL parmi ceux cités, en plus d'autres modules non-mutables. Seuls ces modules sont alloués et calculés dans les UCG. Par cette méthode, le gain est double. Si plusieurs sous domaines se réfèrent à la même UCG, comme c'est le cas dans l'exemple, la mémoire n'est pas dupliquée. De plus le calcul des modules contenus dans l'UCG n'est réalisé qu'une seule fois, ce qui permet aussi d'épargner du temps de calcul.

A l'inverse, une UCE est définie pour chaque sous domaine, soit 4 UCE pour le colorset. Pour chacune des UCE, le module FLX est alloué ainsi que les modules auxiliaires mutables. Chaque UCE est associée à l'UCG correspondant à son type de sous domaine. Les UCE accèdent aux modules non-mutables de leur UCG par pointeur, sans copie des données. Les pointeurs des modules non-mutables des UCE 1 et 2 sont associés aux modules de l'UCG A, de même que les pointeurs des modules non-mutables des UCE 3 et 4 avec les modules de l'UCG B. Ainsi les UCE forment des Unités de Calcul complètes pour lesquelles il est possible de réaliser un calcul avec le solveur IDT.

Cet exemple illustre le gain lié à l'architecture proposée : seuls 2 UCG sont créées pour 4 sous domaines. Les UCE sont ensuite associées à leur UCG. Une fois que les UCE sont complètement définies, le calcul en DDM est réalisé par l'algorithme de résolution. Les UCE réalisent les calculs des sous domaines en se comportant comme des solveurs IDT instanciés indépendamment.

Finalement une dernière Unité de Calcul est allouée. Elle permet de gérer la décomposition de domaine, la répartition des sous domaines sur la géométrie pivot et les contacts entre les sous domaines. L'accélération des itérations externes par le CMFD est elle aussi allouée dans cette Unité de Calcul. C'est elle qui pilote le calcul et gère les tests de convergence.

Cette stratégie d'organisation des données permet à la fois d'optimiser l'utilisation de la mémoire, mais aussi de conserver la structure interne du solveur IDT. La définition des UCG est à la charge de l'utilisateur : ce dernier les assigne aux zones de la géométrie pivot. Les UCE sont ensuite générées automatiquement.

BILAN DE L'IMPLEMENTATION

L'implémentation de l'algorithme met en avant plusieurs avantages. Le premier, déjà évoqué, est la possibilité d'épargner de la mémoire en partageant des données par l'utilisation d'unités de calcul génératrices. Le second est que l'architecture du solveur IDT n'a pas été modifiée. L'Unité de Calcul comprend l'ensemble des modules du solveur IDT. Les sous-programmes qui réalisent les itérations multigroupes dans les sous domaines sont ceux utilisés dans le solveur IDT. Chaque unité de calcul se comporte comme un solveur multigroupe indépendant qui résout un problème à source. La méthode de DDM associé à l'algorithme PMBJ permet de paralléliser un solveur avec un minimum de modifications dans les sources du code. Seul l'échange des flux angulaires aux interfaces entre les sous domaines ainsi que le calcul de la source de fission sont à adapter pour appliquer la DDM au solveur. Finalement, cette implémentation permet d'améliorer la localité spatiale et temporelle en distribuant les données par sous domaine. Il est ainsi possible d'adapter la taille des sous domaines à celle de la mémoire cache des processeurs pour améliorer l'efficacité de leur utilisation.

4.2.2 Tests à 1 processeur : influence de la taille des sous domaines sur un colorset 3 x 3 hétérogène

Dans cette section, seront mis en évidence les effets liés à la taille des sous domaines. Pour cela nous diviserons un domaine de calcul en un nombre variable de sous domaines afin de mesurer l'efficacité du calcul vis-à-vis de la taille des sous domaines avec un seul processeur de manière à ne pas être perturbé par les effets liés à l'implémentation parallèle.

Dans un premier test, est évalué l'effet vis-à-vis de l'efficacité de l'utilisation du processeur, en particulier les effets de cache liés à la localité temporelle et spatiale des données. Pour cela, le nombre d'opérations à virgule flottante sera fixé dans chaque

calcul, sans chercher à converger. Ce test permettra de d'estimer des tailles limites pour lesquelles la DDM permet une utilisation efficace du processeur.

Dans le second test, est quantifiée l'influence de la taille des sous domaines sur la convergence de la méthode de décomposition de domaine et ainsi de déterminer s'il est possible de combiner efficacité de l'utilisation des processeurs et bonnes propriétés de convergence.

Le cas test utilisé est un colorset en 2 dimensions de 3 x 3 assemblages en milieu infini, présenté en Annexe B. Il est composé de 9 assemblages UOx ayant des taux de combustion différents. Le motif ne présente aucune symétrie. Chaque assemblage est constitué d'un réseau de 16 x 16 cellules hétérogènes (gaine et combustibles représentés par une géométrie exacte, cf. §1.5.1). Le réseau 16 x 16 a été choisi pour l'étude et essayer diverses décompositions de domaine sur le même colorset. Pour réaliser cette étude, nous avons défini un matériau (et jeu de sections efficaces associé) par région, i.e. 13392 milieux, de manière à maximiser les besoins de communication entre le processeur et la mémoire. Les sections efficaces ainsi définies occupent 4.6 GB. Le même calcul a été réalisé pour différents maillages de DD allant d'un seul domaine (géométrie complète) à 2304 sous domaines (1 cellule par sous domaine). Le nombre de cellules par sous domaine est présenté Tableau 4.1 pour les différentes DD réalisées.

Tableau 4.1. Nombre de cellules par sous domaine pour les différentes DD étudiées.

Nombre de sous domaines	1	4	9	16	36	64	144	256	576	2304
Nombre de cellules par sous domaine	2304	576	256	144	64	36	16	9	4	1

4.2.2.a Effets de la DDM sur la localité des données

L'objet de ce test est de quantifier l'effet de la DDM sur l'efficacité de l'utilisation des processeurs. Nous avons vu dans 2.2.1.a que l'architecture influe sur la manière dont les cœurs du processeur peuvent, ou non, utiliser efficacement leur mémoire cache.

Les coûts de calcul les plus importants dans un calcul de transport sont principalement liés aux itérations internes et au calcul des sources de scattering. Pour ce dernier, le flux multigroupe entier ainsi que les sections efficaces de transfert sont impliquées. Une grande quantité de mémoire est potentiellement requise, dépendant linéairement du nombre de matériaux (sections efficaces) et de régions (flux) nécessaires pour décrire le problème à résoudre. Pour les itérations internes, le coût dépend de la méthode de discrétisation mais dépend aussi de la géométrie ainsi que du nombre de matériaux. La taille requise pour chacune de ces deux tâches dépasse rapidement la taille des caches des processeurs disponibles entraînant des communications supplémentaires entre le cache et le CPU dans les parties le plus couteuses du solveur.

La décomposition de domaine (DD) permet de distribuer la mémoire par petits blocs et ainsi améliorer la localité des données. Le flux, les sections efficaces ainsi que les coefficients pour le transport, de même que tous les tableaux auxiliaires sont définis à l'échelle du sous domaine. Il est ainsi possible de faire entrer l'intégralité des données nécessaires au calcul dans le cache si la taille du sous domaine est suffisamment petite.

La charge de travail à effectuer pour chaque calcul a été fixée puisque l'objectif de ce test est de mesurer l'efficacité de l'utilisation du processeur. Le nombre d'itérations externes est de 10, le nombre d'itérations thermiques par itération externe est de 1 et le nombre d'itérations internes est de 5. De cette manière, quelle que soit la DD, le nombre d'opérations à virgule flottante reste le même dans le calcul des sous domaines et il est ainsi possible d'identifier les coûts provenant des communications entre la mémoire et le CPU. De plus ce test permet aussi de mettre en évidence le coût des échanges du flux angulaire d'interface entre les sous domaines. Les calculs sont réalisés avec un seul processeur d'une machine standard (cf. Annexe C). Dans cette section, l'efficacité de la décomposition de domaine est définie comme le rapport du temps de calcul avec 1 sous domaine par le temps avec n sous domaines :

$$E(n) = T(1)/T(n) .$$

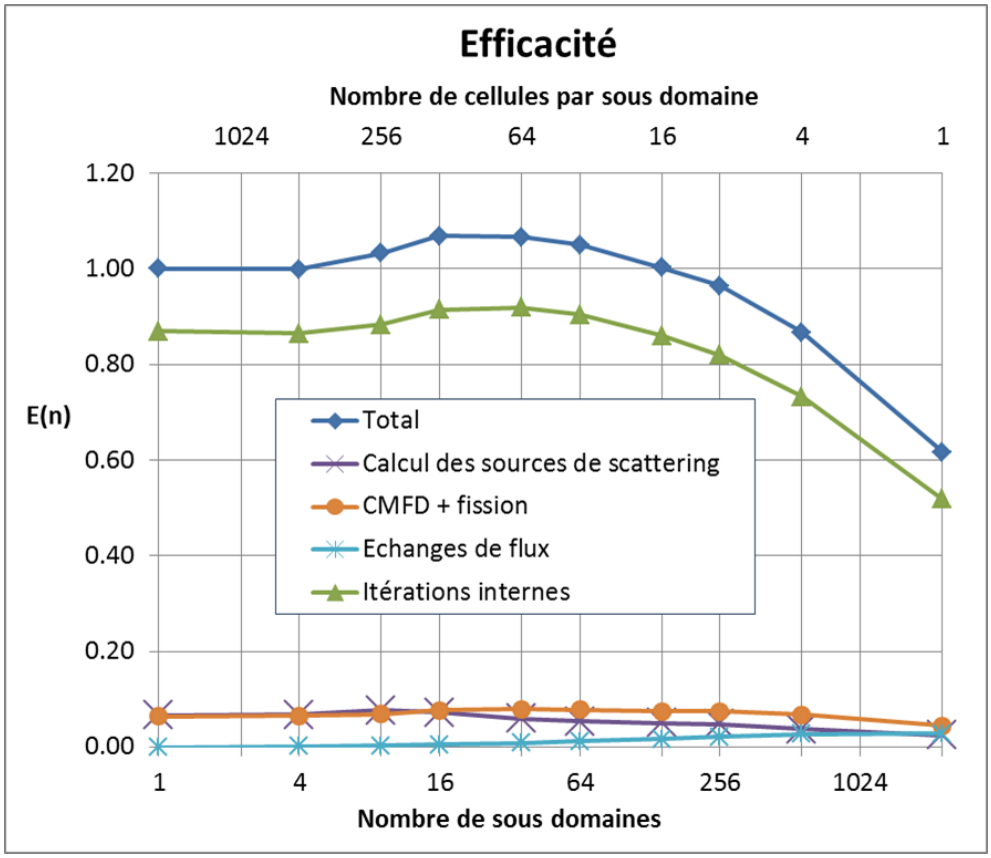


Figure 4.3. Efficacité de la décomposition de domaine.

L'efficacité est présentée à la Figure 4.3 pour les différentes DD. Le graphique montre que l'efficacité est supérieure à 1 pour des sous domaines dont la taille va de 16x16 cellules à 4x4 cellules, divisant respectivement le motif en 9 et 144 sous domaines. Le

gain est de l'ordre de 10% pour des sous domaines de 12x12 ou 8x8 cellules. Les différentes contributions au temps de calcul d'une itération multigroupe sont présentés : celle correspondant aux itérations internes (Itérations internes), celle du calcul des sources multigroupes (Calcul des sources de scattering), celle des échanges de flux entre les sous domaines (Echanges de flux) et celle du temps dédié à l'accélération par le CMFD et au calcul de la source de fission (CMFD + fission).

La principale contribution au temps de calcul est le temps passé dans les itérations internes. C'est aussi celui qui varie le plus par rapport au nombre de sous domaines.

Afin d'expliquer la variation du temps de calcul, est présenté le temps moyen par itération externe pour les éléments contribuant au temps de calcul. Pour une itération externe, la Figure 4.4 présente le temps passé dans 281 itérations internes[1] (Itérations internes), le temps passé dans le calcul des sources multigroupes (Calcul des sources de scattering), le temps passé dans les échanges de flux entre les sous domaines (Echanges de flux) et le temps dédié à l'accélération et au calcul de la source de fission (CMFD + fission).

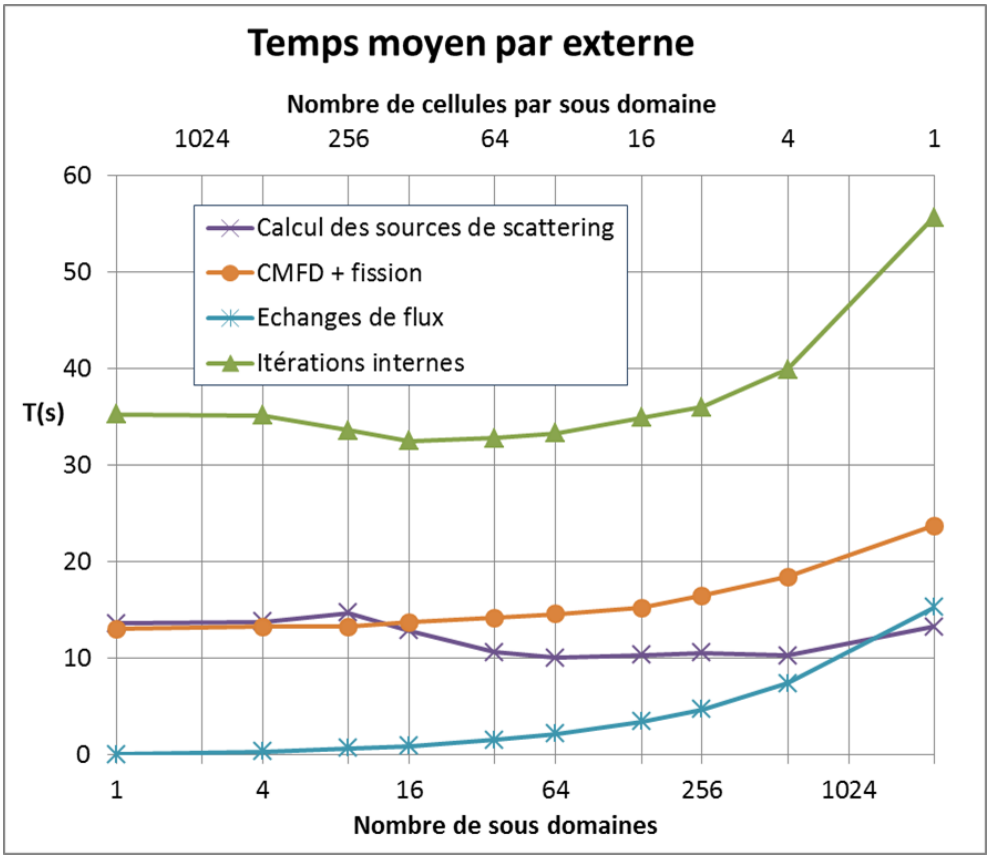


Figure 4.4. Analyse d'une itération multigroupe.

Concernant le calcul de transport, les parties les plus coûteuses sont les itérations internes et le calcul des sources de scattering multigroupes. Le temps pour les itérations internes commence à diminuer à partir de 576 cellules par sous domaine alors que le temps de calcul des sources diminue à partir de 256 cellules par sous

[1] Soit une itération interne par groupe.

domaine. Pour une meilleure compréhension de ce comportement nous allons étudier la variation du temps de calcul des sources par rapport à la taille des sous domaines. Le Tableau 4.2 donne l'occupation mémoire moyenne requise pour calculer la source de scattering dans un groupe en fonction du nombre de cellules par sous domaine.

Tableau 4.2: Estimation de l'occupation mémoire (MB) par sous domaine pour le calcul des sources (évaluation pour 3 moments spatiaux, 3 moments angulaires).

Nombre de sous domaines	1	4	9	16	36	64	144
Nombre cellules par sous domaine	2304	576	256	144	64	36	16
Nombre de régions par sous domaine	13392	3348	1488	837	372	204	93
Mémoire par groupe d'arrivée pour la matrice de transfert par sous domaine	17.7	4.41	2	1.1	0.49	0.26	0.12
Mémoire pour le flux multigroupe par sous domaine	135.5	33.9	15.1	8.5	3.8	2.1	0.9
Mémoire pour le calcul de la source de scattering d'un groupe	162.8	40.7	18.2	10.2	4.6	2.5	1.1

La routine dédiée au calcul des sources nécessite le flux multigroupe en entier ainsi que la ligne des matrices de transfert pour le groupe concerné et des tableaux auxiliaires. La machine utilisée (Intel® Xeon E5-2620 2.00GHz) possède une mémoire cache de 15.3MB. L'occupation mémoire de cette routine devient plus petite que la taille du cache à partir du cas avec 16 sous domaines et le temps de calcul des sources commence lui aussi à décroitre à partir du cas avec 16 sous domaines. Un comportement similaire est observable pour le balayage transport avec un gain d'environ 10% sur le temps de calcul. **Lorsque les données entrent entièrement dans le cache, le temps de calcul est amélioré par leur localité temporelle et spatiale.**

La Figure 4.4 montre aussi que le temps de calcul augmente lorsque le nombre de sous domaine est très élevé, i.e. la taille d'un sous domaine est très petite. Dans une telle situation la quantité de données surfaciques est du même ordre de grandeur que la quantité de données volumique ($N_x + N_y \gtrsim N_r$). Lorsqu'une itération interne est réalisée avec le solveur IDT, le flux de frontière entrant du sous-domaine est chargé dans un tampon pour effectuer le balayage spatial sur les régions, tandis que le flux de frontière sortant est mis à jour à la fin du balayage, et ce pour chaque direction angulaire. D'autres opérations similaires d'utilisation de tableaux auxiliaires sont réalisées à ce niveau et leur coût, répété pour chaque sous domaine, commence à être important par rapport aux opérations à réaliser pour le calcul transport lui-même. Cet effet est lié au fait que le solveur IDT n'a pas été modifié. **Lorsque la quantité de données surfaciques est du même ordre de grandeur que la quantité de données volumiques dans un sous domaine, l'overhead de la DDM devient important.**

On peut aussi noter que lorsque la taille des sous domaines diminue, le nombre d'interfaces entre les sous domaines augmente, ce qui explique l'augmentation du

temps dédié à l'échange des flux d'interface (Echanges de flux). Cependant ce temps reste très faible, moins de 5% du temps de calcul séquentiel, alors que le nombre d'interfaces a été multiplié par 48 entre le calcul avec 1 sous domaine et le calcul avec 2304 sous domaines. **En parallélisant les échanges de flux entre sous domaines, leur contribution au temps de calcul ne devrait alors qu'être marginale.**

Lorsque la taille des sous domaines diminue, le temps de calcul diminue. Ce phénomène est lié à la localité des données dans le cache, comme expliqué par le détail du calcul des sources. Lorsque la taille des sous domaines devient trop petite, le temps de calcule augmente, à cause de l'utilisation de tableaux auxiliaires dans le solveur IDT. Finalement on peut remarquer que le temps de calcul n'augmente pas pour le calcul à 2304 cellules par sous domaine. Ce phénomène est lié à la localité spatiale des données dans le solveur IDT. En effet, lorsque les tableaux ne rentrent plus dans le cache, ils sont chargés par ligne. Le fait que le temps de calcul ne se dégrade pas au-delà de 576 cellules par sous domaine montre que les lignes de cache chargées sont entièrement utilisées. **La localité spatiale des données est bien exploitée par le solveur IDT.**

CONCLUSION

Par ce test il a été montré que :

- Lorsque les données entrent entièrement dans le cache, le temps de calcul est amélioré par leur localité temporelle et spatiale.
- Lorsque la quantité de données surfaciques est du même ordre de grandeur que la quantité de données volumique dans un sous domaine, l'overhead de la DDM devient important.
- En parallélisant les échanges de flux entre sous domaines, leur contribution au temps de calcul ne devrait alors qu'être marginale.
- La localité spatiale des données est bien exploitée par le solveur IDT.

4.2.2.b Influence de la taille des sous domaines sur la convergence

Le cas test jusque-là présenté démontre les avantages de la méthode de décomposition de domaine vis-à-vis de l'efficacité d'utilisation des ressources de calcul. Cependant le temps effectif à prendre en compte est le temps nécessaire à la convergence du calcul. De ce fait, nous avons reproduit les tests précédents en les menant à convergence. Les critères sont fixés à 10^{-5} pour la valeur propre et le flux, 10^{-4} pour la source de fission et 10^{-3} pour les courants d'interface. La Figure 4.5 présente les résultats obtenus : le temps de calcul en fonction du nombre de sous domaines. Deux maillages CMFD ont été utilisés pour ce test : 1 cellule par maille grossière et 4 cellules par maille grossière. Le nombre d'itérations externes est aussi indiqué pour chaque cas.

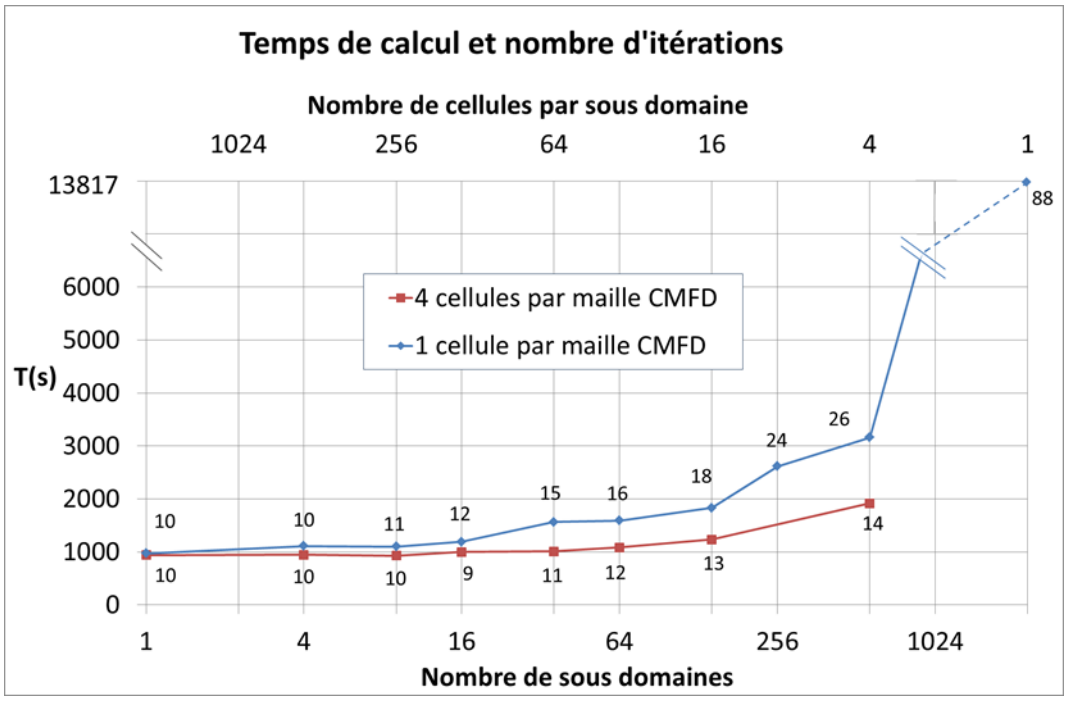


Figure 4.5: Convergence pour différentes tailles de sous domaines.

Lorsque le maillage spatial du CMFD est défini par avec une maille par cellule, la décomposition de domaine est sensible à la taille des sous domaines : le nombre d'itérations et le temps de calcul augmentent avec le nombre de sous domaines. Ce phénomène provient du fait que le CMFD est construit à partir des grandeurs du transport, entachées de l'erreur portée par les flux d'interface. L'efficacité du CMFD est alors réduite lorsque la taille des sous domaines diminue. Cependant, lorsque le maillage du CMFD est composé de 4 cellules par maille, le temps de calcul ainsi que le nombre d'itérations reste stable jusqu'à 36 sous domaines. L'homogénéisation de plusieurs régions permet de compenser les erreurs dues aux flux d'interface. Le maillage CMFD le plus fin n'est pas optimal dans cette situation et l'homogénéisation de plusieurs cellules hétérogènes améliore l'accélération. Le meilleur choix est donc un compromis prenant en compte le maillage CMFD ainsi que la décomposition du domaine.

Entre le calcul direct et le cas avec 64 cellules par sous domaine, le temps de calcul reste relativement constant avec le maillage CMFD le plus grossier. Le nombre d'itérations n'augmente que lorsque le nombre de cellules par sous domaine est inférieur à 36. Alors que le calcul à 1 sous domaine converge en 10 itérations, celui à 64 sous domaines converge en 12 itérations et le temps de calcul augmente de l'ordre de 10%. **Jusqu'à la taille du quart d'assemblage, le temps de calcul DDM reste similaire au calcul Direct.**

Il est intéressant de noter que lorsque la taille des sous domaines devient faible, le nombre d'itérations augmente considérablement. Dans une direction donnée, le flux sortant d'un sous domaine est obtenu par la contribution des sources (de scattering et de fission) plus celle du flux de frontière entrant. Lorsque les sous domaines sont optiquement épais, la contribution des sources à l'intérieur du sous domaine est

dominante. Au contraire, lorsque l'épaisseur optique diminue, la contribution du flux entrant domine. L'analyse de l'épaisseur optique des sous domaines par rapport au rayon spectral est étudiée dans [57] par une analyse de Fourier d'un système en milieu infini. B.W. Kelley y montre que lorsque l'épaisseur optique augmente, le flux d'interface devient moins dominant et les sources limitent la convergence. La diminution de l'épaisseur optique des sous domaines entraine l'augmentation du nombre d'itérations à cause du retard dans la transmission du flux aux interfaces et l'erreur répercutée dans le calcul des sources. **L'épaisseur optique des sous domaines influe sur la convergence de l'algorithme.**

Les résultats présentés dans cette section nous confortent dans l'idée que le DDM accéléré par le CMFD est une bonne solution pour paralléliser efficacement le calcul de cœurs de REL. Un cœur de réacteur comporte de l'ordre de 200 assemblages. Pour calculer un cœur complet en 2 dimensions, le nombre de sous domaines peut aller jusqu'à plus de 1000 en prenant 36 cellules par sous domaine. Cette décomposition permet de distribuer le calcul sur un grand nombre de processeurs avant d'atteindre la limite de scalabilité due à la taille des sous domaines.

4.3 Parallélisme « natif » : parallélisation en mémoire partagée

L'objectif de la DDM développée étant de pouvoir accélérer la résolution de l'équation du transport pour des calculs d'ingénierie, nous avons cherché à optimiser son efficacité sur des machines de bureau standard. Ces dernières sont composées de plusieurs cœurs partageant la même mémoire et donc la parallélisation en mémoire partagée semble naturelle.

Plusieurs arguments viennent renforcer le choix de cette voie de parallélisation :

- En premier lieu, l'optimisation de la mémoire avec les distinctions UCG/UCE est conservée par le parallélisme à mémoire partagée. Tous les threads peuvent accéder aux modules des UCG, placés dans un espace mémoire visible de tous les threads.
- De plus, les données sont intégralement distribuée par sous domaine. Seules des données contenues dans les UCG sont partagées entre les sous domaines et leur accès se fait uniquement en lecture. Les conflits entre les threads ne sont donc pas possibles pour les accès à la mémoire. On peut donc s'attendre à une scalabilité idéale pour le calcul des sous domaines.
- Ensuite la parallélisation coarse grained du solveur DDM est facilitée, contrairement aux aprioris de cette approche en mémoire partagée. Les pointeurs d'accès aux modules sont les variables privées des threads. Pour le calcul d'un sous domaine par un thread, il suffit de les rediriger vers les modules associés au sous domaine.
- Les communications entre les sous domaines (flux entrant à l'interface) se résument à la recopie d'un tableau dans un autre, là encore sans besoin de faire communiquer plusieurs processeurs.

- Enfin, le fait d'avoir une mémoire partagée permet de gérer dynamiquement la répartition des tâches sur les threads. Dans la situation où le nombre de sous domaines est plus grand que le nombre de threads disponibles, leur distribution peut être adaptée dynamiquement, de manière à fluidifier l'enchainement des calculs et équilibrer la répartition des tâches.

IMPLEMENTATION AVEC DESYNCHRONISATION DES TACHES PARALLELES

La bibliothèque OpenMP a été utilisée pour sa simplicité et sa portabilité. Les pointeurs utilisés pour accéder aux modules des Unités de Calcul ont reçu la directive `THREADPRIVATE`. Ils sont alors copiés dans chaque thread. Ainsi, les pointeurs du thread sont associés aux modules de l'UCE pour le calcul du sous domaine. L'Algorithme 4.1 reprend l'algorithme séquentiel avec les directives OpenMP, `!$OMP PARALLEL` et `!$OMP END PARALLEL`, qui indiquent les zones parallélisées. Les tâches effectuées en parallèle sont les listes « ***Pour chaque*** $\alpha \in A$, ***faire*** », c'est-à-dire les tâches effectuées sur les sous domaines. Deux zones parallèles sont définies avec les directives OpenMP. La première (zone parallèle 1) comprend les échanges des flux d'interface, le calcul transport multigroupe et l'homogénéisation des données sur le maillage grossier. L'accélération du flux et le calcul de la source de fission sont dans la seconde (zone parallèle 2). Chaque liste « ***Pour chaque*** $\alpha \in A$, ***faire*** » n'a pas de répartition a priori entre les threads et les sous domaines. Chaque liste parallélisée est définie comme un stock de tâches dans lequel chaque thread « pioche » jusqu'à épuisement. Chaque tâche correspond au traitement d'un sous domaine. L'ordre dans lequel les tâches sont ordonnées dépend de l'algorithme utilisé (PMBJ ou PMBG-S). Pour l'algorithme de PMBJ, l'ordre est basé sur le temps dépensé pour le calcul du sous domaine à l'itération externe précédente : les plus coûteux sont distribués en premier et les derniers viennent ajuster l'équilibrage. Afin de désynchroniser l'enchainement des listes, deux critères sont à respecter. Tout d'abord deux threads ne peuvent pas accéder simultanément au même sous domaine. Ensuite les tâches doivent être effectuées dans l'ordre pour chaque sous domaine : d'abord la mise à jour des flux, ensuite les calculs de transport et enfin les homogénéisations. Ces critères sont gérés par un tableau auxiliaire partagé entre les threads donnant le statut de chaque sous domaine. Ce tableau doit être synchronisé entre les threads pour chaque opération de lecture ou d'écriture. C'est le seule point de synchronisation mémoire entre les threads dans les zones parallèles. La désynchronisation des listes permet de fluidifier l'enchainement des calculs et ainsi réduire les temps d'attente.

Algorithme 4.1. Algorithme PMBJ parallélisé avec OpenMP.

Itérations externes (e)

Tant que λ *et* $Q_\alpha, J_\alpha \; \forall \alpha \in A$ *ne sont pas convergés,* ***faire***

Début de la zone parallèle 1

!$OMP PARALLEL

Mise à jour des flux aux interfaces entre sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A$, ***faire***

$$\begin{cases} (L-H)_\alpha {\psi_\alpha}^{(e+\frac{1}{2})}(x) = {Q_\alpha}^{(e)}(x), & x \in X_\alpha, \\ {\psi_\alpha}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Homogénéisation-condensation sur le maillage grossier pour le CMFD

Pour chaque $\alpha \in A$, ***faire***

$$\left[\phi\,(y),\ J^+(y),\ J^-(y),\ \Sigma_t(y),\ \Sigma_s(y),\ (\chi\nu\Sigma_f)(y)\right]^{(e+1/2)}$$

fin

!$OMP END PARALLEL

Fin de la zone parallèle 1

Calcul des coefficients de diffusion et d'équivalence et construction de la matrice du CMFD

Solution de l'opérateur du CMFD :

$\left(K - H_{\setminus D}\right)\phi^{(e+1)}(y) = F\phi^{(e+1)}(y)/\lambda^{(e+1)}$

Début de la zone parallèle 2

!$OMP PARALLEL

Pour chaque $\alpha \in A$, ***faire***

Accélération des flux dans les sous domaines et sur leurs surfaces

$$\psi_\alpha^{(e+1)}(x) = \psi_\alpha^{(e+1/2)}(x)\frac{\phi^{(e+1)}(y)}{\phi^{(e+1/2)}(y)} \qquad \forall\, x \in y \in X_\alpha\,.$$

Calcul de la nouvelle source de fission normalisée

$${Q_\alpha}^{(e+1)}(x) = \frac{(F\psi)_\alpha{}^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

!$OMP END PARALLEL

Début de la zone parallèle 2

fin

Ce parallélisme coarse grained, de même que le nombre limité de directives OpenMP, permet de limiter les latences et l'overhead associé à la parallélisation, tout en optimisant les capacités des processeurs à réaliser un balayage multigroupe. Le nombre de synchronisation est limité à 2 par itération externe et le recouvrement des tâches est rendu accessible. La partie séquentielle est restreinte au CMFD : construction de la matrice et résolution. La parallélisation du CMFD peut être réalisée de manière indépendante de celle du transport, cela fait partie des voies à suivre pour l'amélioration de la DDM développée.

4.3.1 Scalabilité pour le parallélisme à mémoire partagée

Ce paragraphe est consacré aux performances du parallélisme à mémoire partagée. La notion de scalabilité et un moyen de l'évaluer a été présentée à la section 2.2.1.e : le weak scaling et le strong scaling. Les tests de scalabilité ont été réalisés sur la base du colorset présenté dans l'Annexe B. Il s'agit d'un colorset en 2 dimensions en milieu infini. Il est composé de 9 assemblages UOx. Chaque assemblage est composé d'un réseau de 16x16 crayons permettant de réaliser plus facilement différents maillages de DD.

Afin de mettre en évidence uniquement les propriétés liées à l'implémentation parallèle de la méthode, le nombre d'itérations internes est fixé à 5 et le nombre d'itérations externes est fixé à 10. De cette manière, quelle que soit la DD, le nombre d'opérations à virgule flottante reste le même dans les calculs transport. Les tests ont été réalisés sur une machine standard (cf. Annexe C) avec 12 cœurs et la possibilité d'utiliser l'*hyperthreading* d'Intel (possibilité d'assigner 2 threads par cœur, soit ici jusqu'à 24 threads).

STRONG SCALING

Le strong scaling consiste à résoudre un problème donné avec un nombre croissant de sous domaines, toujours égal au nombre de processeurs. Il permet de d'évaluer jusqu'à quel point la résolution d'un problème peut être accélérée avec l'algorithme parallélisé. Le speedup et l'efficacité sont définis comme à la section 2.2.1.e :

$$S(N) = \frac{t(1)}{t(N)}, \qquad E(N) = \frac{S(N)}{N}.$$

où N est le nombre de threads. Les résultats obtenus sont présentés sur la Figure 4.6 pour le speedup et sur la Figure 4.7 pour l'efficacité :

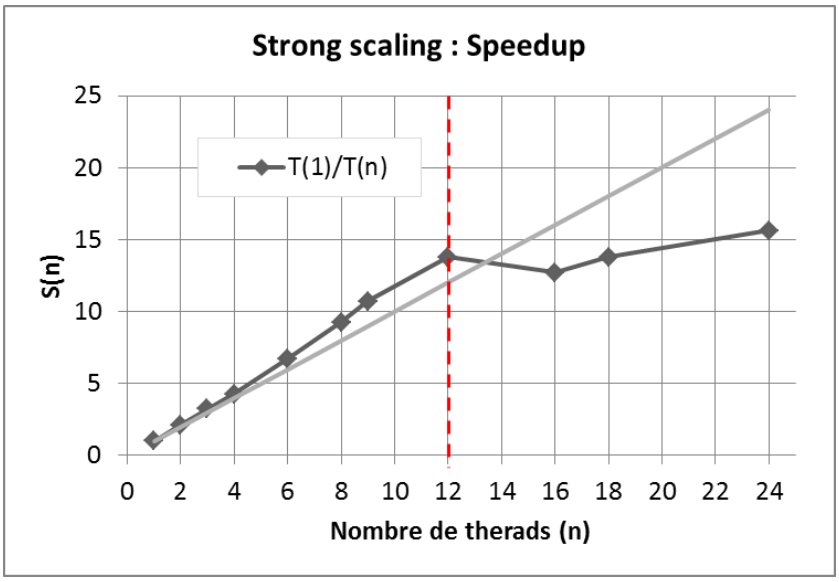


Figure 4.6. Parallélisme à mémoire partagée : speedup strong scaling.

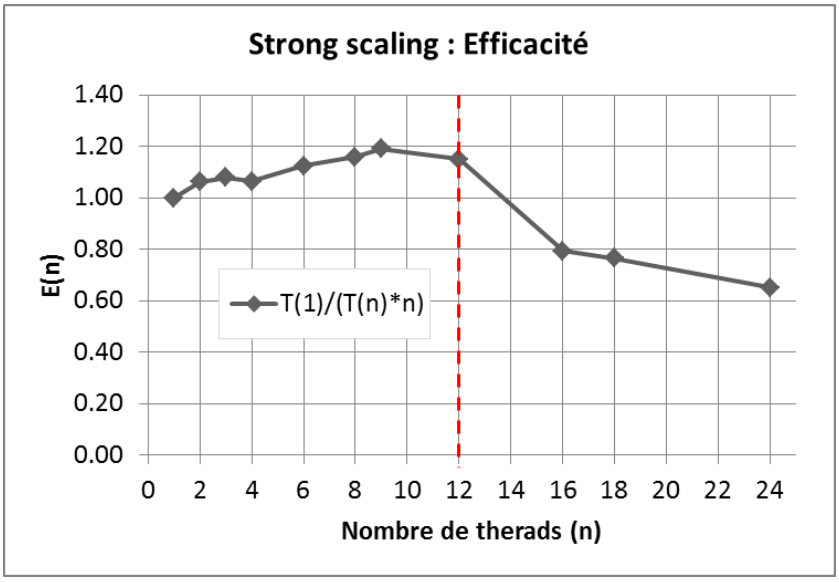


Figure 4.7. Parallélisme à mémoire partagée : efficacité strong scaling.

Les graphiques sont composés de deux parties. De 1 jusqu'à 12 threads, 1 thread est associé à chaque cœur alors que de 13 à 24 threads, l'hyperthreading est activé et chaque cœur supporte jusqu'à 2 threads. Dans la première partie la scalabilité est au-dessus de la courbe idéale (speedup supérieur au nombre de cœurs, efficacité supérieure à 1) grâce aux propriétés de la DDM énoncée dans la section 4.2.2.a. Une fois l'hyperthreading activé, l'efficacité s'écroule et ne suit plus la droite « idéale ». Cependant le speedup progresse jusqu'à 16, permettant à l'utilisateur de réduire le temps de calcul en utilisant l'hyperthreading.

WEAK SCALING

Pour rappel, le weak scaling consiste à accroitre la taille du problème global proportionnellement au nombre de processeurs et de sous domaines. Ce test permet

d'estimer la taille maximale en deçà de laquelle l'algorithme restera performant. Pour le weak scaling la formulation du speedup est modifiée afin d'avoir l'égalité entre le speedup idéal et le nombre de threads dans le cas idéal :

$$S(N) = \frac{N \times t(1)}{t(N)}, \qquad E(N) = \frac{S(N)}{N}.$$

Pour ce test, le calcul est réalisé sur un seul assemblage avec 1 thread, deux assemblages avec 2 threads, et ainsi de suite… Les résultats sont représentés Figure 4.8 et Figure 4.9

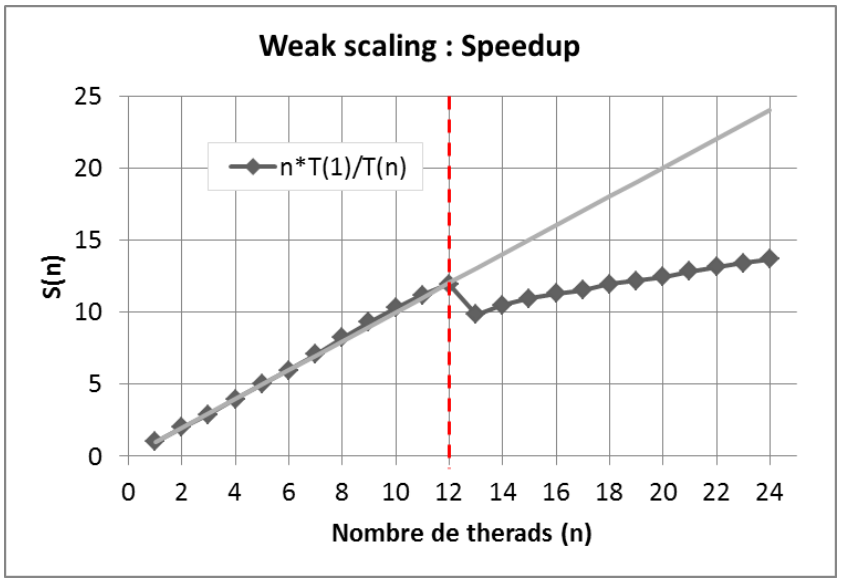


Figure 4.8. Parallélisme à mémoire partagée : speedup weak scaling.

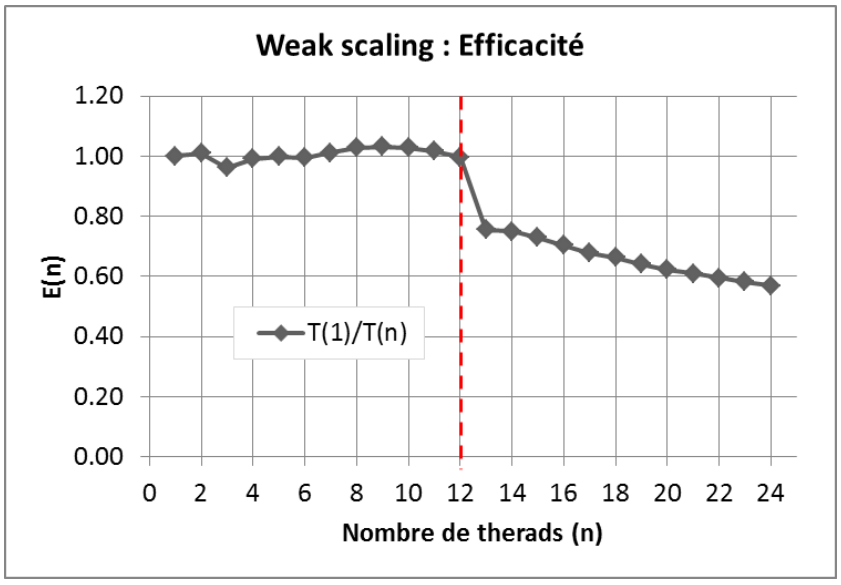


Figure 4.9. Parallélisme à mémoire partagée : efficacité weak scaling.

Là encore, les graphiques sont composés des parties avec et sans hyperthreading. La scalabilité est respectée jusqu'à 12 threads puis se dégrade. Le speedup croît jusqu'à 14 avec 24 threads permettant d'accélérer légèrement le calcul avec l'hyperthreading.

Les tests de scalabilité montrent que l'implémentation du parallélisme à mémoire partagée accélère de manière idéale la méthode de décomposition de domaine lorsque chaque cœur supporte un thread. Une fois l'hyperthreading activé, la scalabilité est dégradée mais le temps de calcul est réduit en utilisant 24 threads. La scalabilité démontrée ici sur les processeurs multi-cœurs nous encourage dans l'utilisation de la parallélisation à mémoire partagée coarsed grained. L'implémentation parallèle de l'algorithme PMBJ permet d'accélérer efficacement des calculs de colorsets d'assemblage de REL sur des machines standard, objectif de ce travail.

EFFET DE LA DESYNCHRONISATION DES TACHES

Dans la présentation de la parallélisation en mémoire partagée (§4.3) nous avons expliqué comment les tâches peuvent être désynchronisées. Afin de mettre en avant l'effet de la désynchronisation des tâches, nous avons décomposé le colorset en 24 sous domaines. Le calcul a été réalisé avec 1, 2, 3, puis 4,… threads de sorte que le nombre de sous domaines n'est pas forcément un multiple du nombre de threads. Il en résulte un déséquilibrage de charge, chaque thread ne pouvant pas calculer le même nombre de sous domaines. La Figure 4.10 représente l'efficacité du calcul avec et sans la synchronisation des tâches.

Comme le montre la courbe avec les tâches synchronisées, l'efficacité diminue fortement lorsque le nombre de threads n'est pas un multiple du nombre de sous domaines comme c'est le cas avec 5, 7, 9, 10 et 11 threads. Ces derniers ne peuvent pas se répartir équitablement les tâches. Certains threads ont donc épuisé les tâches disponibles alors qu'au moins l'un d'entre eux travaille encore.

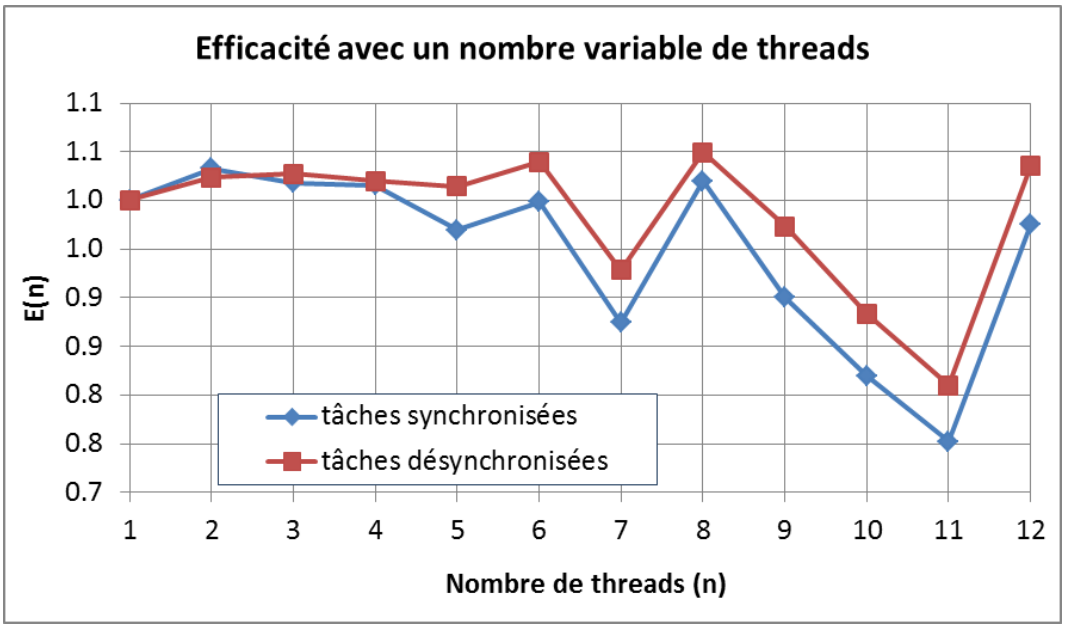


Figure 4.10. Effet de la désynchronisation des tâches.

Cependant, la désynchronisation permet de gagner autour de 10% d'efficacité, principalement en recouvrant le calcul transport des sous domaines par l'homogénéisation sur le maillage grossier (dernière liste de la zone parallèle 1). Par exemple pour le calcul avec 9 threads, chaque thread est responsable du calcul

transport de 2 sous domaines. Il en reste alors 6 à calculer ce qui est fait par 6 des 9 threads. Pendant ce temps, les 3 threads restant homogénéisent les 18 premiers sous domaines, tâche moins coûteuse en termes de temps de calcul. Le coût du calcul *(transport + homogénéisation)* est alors 3 fois celui du calcul transport plus 1 fois celui de l'homogénéisation. Avec la synchronisation, il est de 3 fois le calcul transport plus 3 fois celui de l'homogénéisation. Dans ce test le nombre de sous domaines est relativement petit. Prenons maintenant l'exemple d'un cœur discrétisé en 361 sous domaines (cf. §5.3.6.a) parallélisé sur 12 threads. Chaque thread est en charge du calcul transport de 30 sous domaines, soit un total de 360 sous domaines également distribués entre les threads. Il faut alors assigner un 31ème calcul transport de sous domaine à un thread. Pendant ce calcul, les 11 threads restants peuvent alors se répartir les 360 calculs d'homogénéisation. Le coût de l'homogénéisation n'est visible que pour le dernier sous domaine calculé. Le recouvrement des tâches parallèles permet alors de d'épargner 97% du temps de l'homogénéisation : 30 calculs d'homogénéisation sur les 31 à effectuer sont masqués par le recouvrement.

4.4 Distribution de la mémoire et parallélisme à mémoire distribuée : méthode hybride MPI/OpenMP

Le principal défaut des machines de bureau, basées sur une architecture en mémoire partagée, est le faible nombre de processeurs mais une mémoire disponible limitée. Par exemple, un calcul de cœur complet requiert plusieurs centaines d'assemblages à calculer et plusieurs TeraBytes (TB) de données. Si la contrainte du temps de calcul lié au nombre d'assemblages à calculer peut être considérée comme mineure (il suffit d'attendre assez longtemps pour obtenir le résultat), la limitation de la mémoire est un véritable enjeu. Certains calculs ne sont donc pas accessibles aujourd'hui sur des architectures à mémoire partagée. C'est pour ces raisons que nous avons implémenté le paradigme de parallélisme à mémoire distribuée.

4.4.1 Solution proposée : parallélisme Hybride MPI/OpenMP

L'architecture des calculateurs à mémoire distribuée est constituée de nœuds reliés entre eux par un réseau de communication. Chaque nœud contient généralement plusieurs processeurs qui partagent un même espace mémoire. L'idéal est alors de pouvoir garder les avantages du parallélisme à mémoire partagé à l'intérieur des nœuds tout en ayant accès à la mémoire distribuée.

La solution que nous avons choisi est d'utiliser à la fois les bibliothèques OpenMP et MPI, sans modifier l'algorithme de résolution PMBJ [60]. La bibliothèque MPI permet d'accéder aux architectures à mémoire distribuée alors que l'OpenMP permet de paralléliser en mémoire partagée à l'intérieur des nœuds. De cette manière, l'optimisation de la mémoire avec les distinctions UCG/UCE ainsi que les bons résultats de scalabilité en OpenMP peuvent être conservés grâce au partage de la mémoire à l'intérieur des nœuds de calcul.

Pour illustrer l'implémentation hybride réalisée, la Figure 4.11 reprend l'exemple du colorset de 4 assemblages de la section 4.2.1 (Gestion et distribution des données). Il est composé de deux types d'assemblages différents: l'assemblage type A et l'assemblage type B. Un sous domaine est associé à chacun des 4 assemblages. La parallélisation de cet exemple est réalisée avec 2 processus MPI et 2 threads par processus. Les sous domaines sont distribués sur les nœuds de calcul via les instructions du jeu de données.

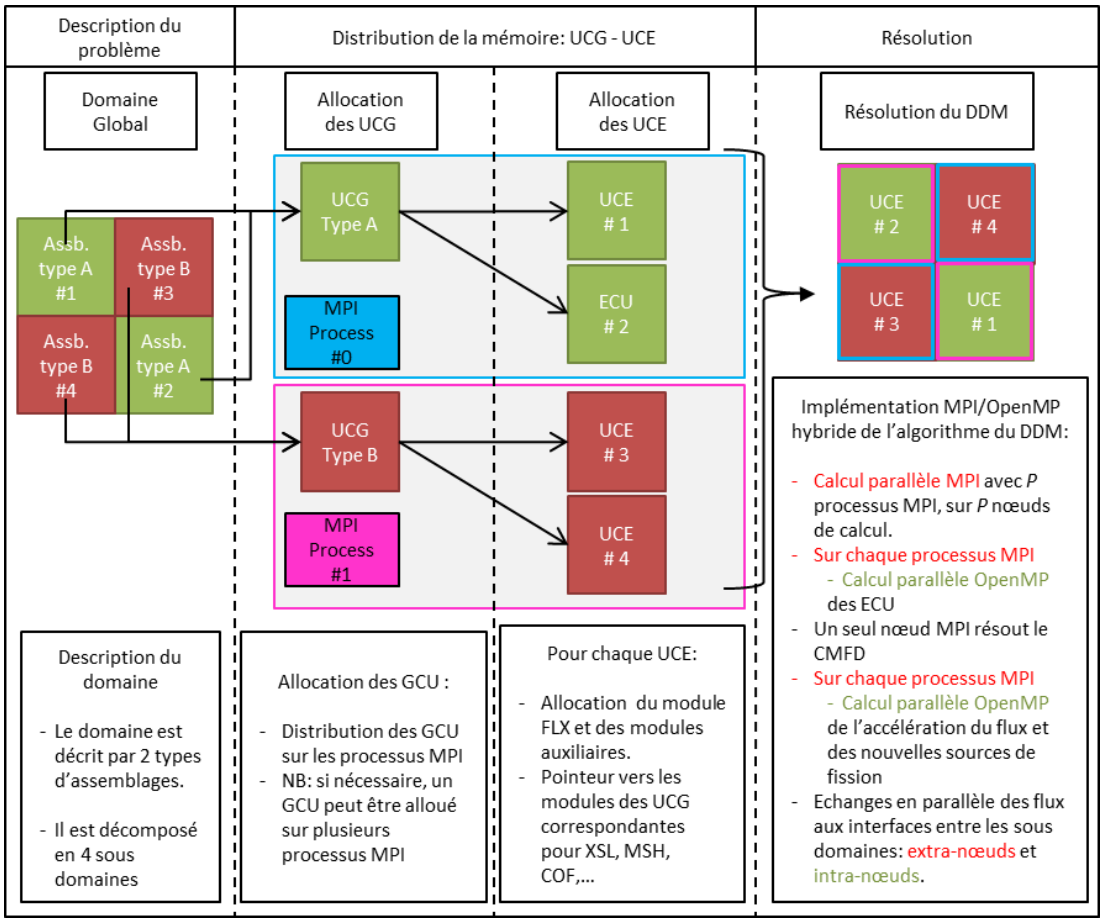


Figure 4.11. Gestion de la mémoire pour l'implémentation du DDM en mémoire partagée.

Dans l'exemple, nous choisissons de répartir les assemblages de type A sur le processus MPI numéro 0 et les assemblages de type B sur le processus 1. Après avoir fait l'inventaire des sous domaines dont il est en charge, chaque processus MPI crée uniquement les UCG nécessaires pour le calcul des sous domaines qui lui ont été attribués. Dans le présent exemple, l'UCG de type A est allouée sur le processus 0 et l'UCG de type B est allouée sur le processus 1. De cette manière, l'optimisation de la mémoire réalisée pour le parallélisme à mémoire partagé est conservée : les UCG ne sont allouées qu'une seule fois pour l'ensemble du calcul à mémoire distribuée. Les UCE sont générées par le processus en charge de leur sous domaine. Dans l'exemple, les UCE 1 et 2 sont allouées sur le processus 0 et les UCE 3 et 4 sur le processus 1. La quantité de mémoire allouée pour les calculs transports avec cette parallélisation en mémoire distribuée est la même que pour le calcul en mémoire partagée. Seule l'unité de calcul de gestion de la DDM et du CMFD est générée sur tous les processus.

Cette manière d'organiser les données permet de conserver les avantages de la gestion de la mémoire du calcul à mémoire partagée au sein d'un nœud tout en accédant au parallélisme à mémoire distribuée. En effet, une implémentation uniquement en MPI n'aurait pas permis de conserver les avantages donnés par les UCG : chaque sous domaine aurait alors dû allouer sa propre UCG. La résolution du problème avec cette implémentation, résumée sur la Figure 4.11, est expliquée en détail ci-dessous.

4.4.2 Modifications de l'algorithme pour l'implémentation hybride

La structure de la réalisation du DDM en mémoire partagée n'est pas modifiée, seuls quelques appels supplémentaires aux routines gérant les communications MPI ont été ajoutés. Les modifications permettant de paralléliser le calcul avec la méthode hybride MPI/OpenMP sont reportées dans l'Algorithme 4.2.

L'algorithme est parallélisé sur P processus MPI distribués sur P nœuds de calcul. Chaque processus, dont l'identifiant est noté p, possède maintenant une liste de A_p sous domaines à calculer tel que $A = \sum_{p=1}^{P} A_p$. Les listes « ***Pour chaque*** $\alpha \in A$, ***faire*** » ont été remplacées par « ***Pour chaque*** $\alpha \in A_p$, ***faire*** », c'est-à-dire restreintes aux sous domaines présents sur le processus MPI p. Au sein d'un nœud, ces listes sont parallélisées en utilisant les directives OpenMP de la même manière que dans l'Algorithme 4.1 (Algorithme PMBJ parallélisé avec OpenMP). L'implémentation hybride permet ainsi de conserver dans chaque nœud certains avantages offerts par le parallélisme à mémoire partagée : l'équilibrage de charge dynamique et l'efficacité des échanges de flux entre les sous domaines d'un même processus.

Algorithme 4.2. Algorithme de DDM parallélisé avec la solution hybride MPI/OpenMP.

P processus MPI
Chaque processus p est en charge de A_p sous domaines tel que $A = \sum_{p=1}^{P} A_p$
Tant que *λ et Q_α, J_α $\forall \alpha \in A$ ne sont pas convergés,* ***faire***

Communications extra-nœud : mise à jour des flux aux interfaces entre sous domaines
Pour chaque $\alpha \in A_p$, ***faire***

```
CALL MPI_SEND / CALL MPI_RECV
```

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \notin N_p : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \notin N_p : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T. \end{cases}$$

fin

```
!$OMP PARALLEL
```

Communications intra-nœud : mise à jour des flux aux interfaces entre sous domaines (intra nœud)

Pour chaque $\alpha \in A_p$, ***faire***

$$\psi_\alpha^{-(e+1)}(x) = \begin{cases} \psi_\beta^{+(e)}(x), & \forall\, \beta \neq \alpha : x \in \Gamma_\alpha^- \cap \Gamma_\beta^+, \beta \in N_p, \\ \psi_\beta^{+(e)}(T^{-1}x), & \forall\, \beta \neq \alpha : (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T, \beta \in N_p. \end{cases}$$

fin

Calcul transport des sous domaines

Pour chaque $\alpha \in A_p$, ***faire***

$$\begin{cases} (L-H)_\alpha \psi_\alpha{}^{(e+1/2)}(x) = Q_\alpha{}^{(e)}(x), & x \in X_\alpha, \\ \psi_\alpha{}^{(e+1)}(x) = \psi_\alpha^{-(e+1)}(x), & x \in \Gamma_\alpha^-. \end{cases}$$

fin

Homogénéisation-condensation sur le maillage grossier pour le CMFD

Pour chaque $\alpha \in A_p$, ***faire***

$$\left[\phi\,(y),\ J^+(y),\ J^-(y),\ \Sigma_t(y),\ \Sigma_s(y),\ (\chi\nu\Sigma_f)(y)\right]^{(e+1/2)}$$

fin

```
!$OMP END PARALLEL
```

Regroupement des grandeurs homogénéisées sur un nœud

```
CALL MPI_REDUCE
```
$\left[\phi\,(y),\ J^+(y),\ J^-(y),\ \Sigma_t(y),\ \Sigma_s(y),\ (\chi\nu\Sigma_f)(y)\right]^{\left(e+\frac{1}{2}\right)}$

Calcul des coefficients de diffusion et d'équivalence et construction de la matrice du CMFD

Solution de l'opérateur du CMFD :

$(K - H_{\setminus D})\phi^{(e+1)}(y) = F\phi^{(e+1)}(y)/\lambda^{(e+1)}$

Distribution du flux CMFD et de la valeur propre sur tous les nœuds

```
CALL MPI_BCAST
```
$[\phi\,(y), \lambda]^{(e+1)}$

```
!$OMP PARALLEL
```

Pour chaque $\alpha \in A_p$, ***faire***

Accélération des flux dans les sous domaines et sur leurs surfaces

$$\psi_\alpha^{(e+1)}(x) = \psi_\alpha^{(e+1/2)}(x)\frac{\phi^{(e+1)}(y)}{\phi^{(e+1/2)}(y)} \qquad \forall\, x \in y \in X_\alpha.$$

Calcul de la nouvelle source de fission normalisée

$$Q_\alpha{}^{(e+1)}(x) = \frac{(F\psi)_\alpha{}^{(e+1)}(x)}{\lambda^{(e+1)}}.$$

fin

```
!$OMP END PARALLEL
```

fin

Les communications extra-nœud sont les suivantes :

- **Les échanges de flux aux interfaces entre sous domaines placés sur des processus différents** sont réalisés hors de la partie parallélisée en mémoire partagée. Ce sont des communications point-à-point de type `MPI_SEND/MPI_RECV`. Ces communications sont réalisées avant le calcul des sous domaines et ne permettent que l'utilisation de l'algorithme PMBJ. L'implémentation des algorithmes de type PMBG-S n'a pas été réalisée pour l'implémentation hybride. Le détail de l'algorithme de communications est donné dans en Annexe D.
- **Le second point de communications concerne le CMFD**, défini à l'échelle globale. Il requiert 2 communications. La première consiste à transmettre les paramètres homogénéisés au processus qui résout le CMFD. Ce dernier, le processus maitre dans notre implémentation, doit ensuite distribuer la solution à tous les autres nœuds : le flux scalaire et la valeur propre du CMFD. Ces communications sont les plus coûteuses puisqu'il s'agit de communications collectives et que leur volume est important. Le CMFD global est alloué dans tous les processus dans la réalisation actuelle. Chaque processus calcule les paramètres homogénéisés de ses sous domaines (flux scalaires, courants partiels et sections efficaces) sur le maillage du CMFD global. Une opération de réduction est ensuite utilisée pour les regrouper dans le processus maitre : `MPI_REDUCE`. La distribution de la solution obtenue par le processus maitre est réalisée par l'appel à `MPI_BCAST`. Cette implémentation n'est certainement pas optimale. Cependant, le besoin de paralléliser le CMFD, déjà évoqué à la section 3.3.4.b dans l'étude de la fraction du temps de calcul dédiée au CMFD, ne nous a pas incité à optimiser ces communications, la parallélisation du CMFD s'avérant nécessaire. Par exemple, une DDM appliquée au CMFD permettrait de se passer de ces communications collectives.
- **Finalement les critères de convergences** sont vérifiés à l'échelle globale. Une communication collective permet de communiquer les erreurs (fission, courants et flux) des sous domaines de chaque processus pour tester le respect des critères de convergence. Cette dernière communication est négligeable par son volume de données, d'autant qu'à ce point les processus sont déjà synchronisés.

Les sous programmes utilisés dans l'implémentation hybride sont les même que ceux développés pour l'algorithme à mémoire partagée. Les spécificités présentées ci-dessus ont été implémentées dans de nouveaux sous programmes appelés uniquement lorsque le parallélisme à mémoire distribué est utilisé, à la manière d'une bibliothèque permettant de coupler plusieurs calculs à mémoire partagée.

PARAMETRES A PRENDRE EN COMPTE POUR LA DISTRIBUTION DES SOUS DOMAINES SUR LES PROCESSUS

Trois paramètres principaux sont pris en compte pour distribuer les sous domaines entre les processus MPI :

- Le 1er est **l'équilibrage de charge**. Pour que l'efficacité du parallélisme soit optimale, il est important de veiller à ce que la charge de travail soit également réparti

entre les processus. La répartition des processus doit prendre en compte le nombre de sous domaines ainsi que la charge de travail associé à chaque sous domaine. La charge de travail dépendant à la fois du nombre de mailles par sous domaine et du rayon spectral de chaque sous domaine, elle est donc difficilement évaluable précisément.

- Le 2nd paramètre concerne **l'optimisation de la mémoire**. En regroupant sur le même nœud plusieurs UCE qui partagent la même UCG, des recopies sont évitées. IDT fait un bilan de la mémoire allouée dans les modules, il est alors possible d'optimiser la distribution de sous domaines sur les différents processus lorsque le calcul à effectuer est particulièrement sollicitant en mémoire.

- Le 3ème est **l'optimisation des communications extra-nœud pour l'échange de flux entre les sous domaines**. Suivant la répartition des sous domaines sur les processus MPI, le nombre de communications extra-nœud varie. La répartition des sous domaines sur les nœuds peut être réalisée de manière quelconque. La topographie des processus MPI n'a alors a priori aucune propriété de régularité géométrique. L'algorithme doit permettre de réaliser un maximum de communications simultanément, ce qui n'est pas trivial avec une répartition quelconque. L'algorithme gérant les communications implémenté pour une topographie quelconque est présenté en Annexe D.

Ces 3 paramètres sont actuellement gérés par l'utilisateur qui peut alors prendre en compte les critères qui sont les plus importants pour lui : optimisation des ressources, optimisation de l'équilibrage de charge ou optimisation des communications. La répartition des UCE sur les processus MPI n'affecte pas l'algorithme. Le choix de l'optimisation dépend du calcul à réaliser. En effet, l'optimisation de la mémoire peut être le premier choix pour un calcul d'évolution avec un nombre de groupe élevé. La quantité de mémoire dédiée aux sections efficaces et aux coefficients du transport est importante et peut rapidement dépasser les mémoires disponibles si elle n'est pas correctement distribuée. A l'inverse, pour un calcul avec peu de groupes en énergie et/ou peu de matériaux, les contraintes mémoires sont moindres. L'utilisateur peut alors chercher à minimiser le déséquilibrage de charge ainsi que le coût des communications extra-nœud. Notons tout de même que le poids des communications des flux d'interface doit être relativement faible dû au nombre limité de communications : une seule communication est réalisée par itération externe.

4.4.3 Bilan de l'implémentation

En conclusion, la méthode hybride développée permet d'accéder au parallélisme à mémoire distribuée sans modification de l'algorithme. Elle permet aussi de garder certains avantages du parallélisme à mémoire partagée comme l'équilibrage de charge dynamique, l'efficacité des échanges des flux (intra-nœud), l'optimisation de la mémoire avec les UCG,… Cependant l'accélération par le CMFD s'apparente à une voie d'étranglement, extrêmement plus pénalisante que pour le calcul en mémoire partagée. En effet, la résolution du CMFD est réalisée en séquentiel sur un seul nœud. L'accès à la mémoire distribuée permettant d'accéder à plus de processeurs, on se rapproche de la limite donnée par la loi d'Amdahl. De plus pour calculer la matrice du

CMFD, les grandeurs transport homogénéisées doivent être transmises au processus réalisant le calcul. De même, la solution calculée par l'accélération doit être transmise à tous les nœuds de calcul. Ces communications collectives sont coûteuses. La distribution de la mémoire et la parallélisation du CMFD constituent des voies d'amélioration de l'algorithme. Ce dernier point ainsi que l'optimisation de l'algorithme de communication des flux d'interfaces n'ont pas été réalisés dans le travail présenté ici. De ce fait nous n'avons pas encore mené d'étude de la scalabilité de l'implémentation en mémoire distribuée. Les résultats obtenus dans l'état actuel de la réalisation ne permettraient pas de tirer d'enseignement par rapport au potentiel de l'algorithme. Malgré cela, nous avons pu effectuer avec succès des calculs réalistes présentés dans le chapitre suivant.

4.5 Conclusion

Dans ce chapitre nous avons décrit la manière dont nous avons implémenté l'algorithme PMBJ accéléré par le CMFD dans une maquette.

Tout d'abord l'architecture de la réalisation a été décrite. Nous avons vu que la gestion de la mémoire est optimisée en évitant les copies de données redondantes entre sous domaines. De plus le test réalisé sur un colorset d'assemblages de REP nous a permis d'illustrer le fait que la méthode permet d'améliorer la localité des données dans le cache. Nous avons aussi étudié les tailles limites de sous domaines pour lesquelles la DDM est efficace. Cette étude nous a permis d'estimer qu'il est possible d'utiliser un grand nombre de sous domaines pour paralléliser un calcul de cœur.

Ensuite nous avons présenté la parallélisation en mémoire partagée coarse grained qui permet de conserver les avantages vis-à-vis de l'optimisation de la mémoire. Une étude de scalabilité sur une machine standard a démontré une scalabilité idéale pour le calcul d'un colorset d'assemblages, conformément aux attentes.

Finalement nous avons présenté l'implémentation pour le parallélisme à mémoire distribué. La solution hybride MPI/OpenMP choisie permet de garder certains avantages du parallélisme à mémoire partagée tout en ayant accès à un plus grand nombre de processeurs ainsi qu'une plus grande quantité de mémoire. Cette implémentation permet de distribuer plusieurs sous domaines par processus MPI puis d'utiliser le parallélisme à mémoire partagée dans chaque processus. La topographie des processus MPI peut être quelconque. Cependant l'algorithme de communication n'est pas optimal.

Cette implémentation via une maquette a un but de démonstration du potentiel de la méthode. Cependant elle ne répond pas aux attentes pour une utilisation dans le cadre d'études de recherche ou industrielle. Par exemple, la possibilité de pouvoir gérer automatiquement la DD et l'équilibrage de charge, ainsi que l'insertion de l'algorithme de résolution et son architecture dans une chaine de calcul complète sont à prendre en compte pour l'intégration dans APOLLO3. Dans le chapitre suivant, nous présenterons plusieurs exemples d'applications réalistes utilisant ce développement.

Chapitre 5. Mise en œuvre et validation de la méthode sur divers cas d'application à 2D et 3D

5.1 Introduction

Nous avons réalisé plusieurs applications pour illustrer les capacités de la Maquette à reproduire des calculs haute-fidélité avec les ressources disponibles. Ces résultats seront présentés dans cette section.

Tout d'abord, nous présenterons le calcul du Benchmark C5G7 dans la section 5.2. Ce benchmark propose plusieurs configurations d'un « mini-cœur » sans homogénéisation spatiale avec des bibliothèques de sections efficaces à 7 groupes d'énergie. Cette première application permettra de valider les résultats fournis par la Maquette DDM par comparaison à la référence Monte Carlo du benchmark. De plus elle permettra d'évaluer la précision de notre solveur vis-à-vis des autres méthodes déterministes.

Les applications qui seront présentées par la suite auront pour but de démontrer les capacités de la Maquette en termes de précision, de gestion de la mémoire et d'ordre de grandeur de temps de calcul. L'étude de la scalabilité ainsi que l'optimisation du temps de calcul ne sont pas les objectifs de cette section. En effet, un travail d'optimisation, voire d'automatisation, sont à effectuer avant de présenter des résultats intéressants à ce sujet. On cherchera donc à montrer en priorité la faisabilité de calculs standards en physique des cœurs mais couteux en temps de calcul et/ou en mémoire (lié à la résolution de l'équation du transport avec une discrétisation spatiale et énergétique fine) que la Maquette rend dès-à-présent accessible.

Nous proposerons ainsi le calcul d'un grand cœur de REL (état de début de cycle d'un cœur UOx) avec un réflecteur lourd en 2D[2]. Cette application permettra de mettre en avant la capacité de la Maquette à traiter un problème de grande taille nécessitant de calculer 241 assemblages combustibles et le réflecteur environnant. Dans notre approche, nous avons réalisé ce calcul en prenant le nombre de groupes utilisé pour les calculs de « référence » déterministes. Il s'agit du maillage SHEM [61] à 281 groupes d'énergie. La description géométrique est similaire à celle d'un calcul standard réseau en milieu infini, c'est-à-dire une représentation des crayons combustibles sans homogénéisation. Le résultat de ce calcul sera comparé à un résultat de référence Monte Carlo obtenu avec le code TRIPOLI-4® [6]. Finalement, cet exercice se poursuivra par une estimation des besoins en mémoire et en temps de

[2] Notons que le calcul d'un cœur de REL disposant d'un plan de chargement mixte UOx-MOx aurait permis de mettre en avant les capacités de la méthode pour le traitement des interfaces entre assemblages combustibles (hypothèse du mode fondamental). Toutefois nous avons privilégié une configuration UOx pour laquelle nous disposons d'une référence de calcul Monte Carlo.

calcul nécessaires pour réaliser un calcul d'évolution isotopique. Ce type de calcul est particulièrement couteux en mémoire : le nombre de matériaux nécessaires pour décrire le cœur est lié au nombre de régions dans lesquelles les concentrations isotopiques peuvent évoluer différemment. Une simulation d'un tel calcul sera présentée en prenant en compte un volume de données représentatif d'une évolution du cœur.

Nous discuterons dans une troisième partie du calcul d'un colorset de 9 assemblages en 3D. Il s'agit d'un calcul à 281 groupes, avec un maillage spatial raffiné. Bien que le nombre d'assemblages soit nettement inférieur au calcul de cœur en 2D, le nombre de régions est supérieur d'un facteur 3. Dans ce test, les assemblages 3D présentent une description axiale fine (une centaine de mailles pour représenter les 4 mètres de hauteur active) tandis que la description radiale est similaire à celle utilisée pour le cas du cœur à 2D.

Enfin, la dernière application que nous proposons permet d'illustrer les capacités de la maquette mise en œuvre au cours de cette thèse. Il s'agit d'un calcul de cœur en 3D, reprenant la discrétisation radiale du calcul de cœur 2D et la discrétisation axiale du calcul de colorset 3D, mais avec un nombre de groupes d'énergie réduit à 26. Ce calcul permettra d'illustrer les possibilités offertes par l'algorithme PMBJ accéléré par le CMFD implémenté dans la Maquette. Dans cette application, nous définirons ainsi près de 8000 sous domaines, dépassant largement les possibilités de parallélisation offertes par les clusters disponibles.

5.2 Validation de la méthode sur le cas « C5G7 MOX Benchmark »

Le Benchmark C5G7 [4, 5] a été proposé pour tester la capacité des méthodes de transport à traiter des configurations de cœurs de réacteur sans homogénéisation spatiale. L'objectif du travail présenté dans cette section est ainsi de tester les capacités de la Maquette à reproduire des résultats de benchmark connus et d'évaluer la précision du solveur vis-à-vis des autres méthodes déterministes.

Dans ce benchmark, des configurations 2D et 3D ont été sélectionnées et des références Monte Carlo très précises ont été obtenues, permettant une comparaison de la distribution de puissance fine à l'échelle du crayon combustible. Dans cette section, nous étudions la configuration 2D du benchmark original [4] ainsi que les configurations 3D de l'extension [5]. Le benchmark est un « mini-cœur » composé de 16 assemblages combustibles de type UOx et MOx avec 3 plans de symétrie, ce qui permet de n'en étudier qu'un huitième comme présenté Figure 5.1.

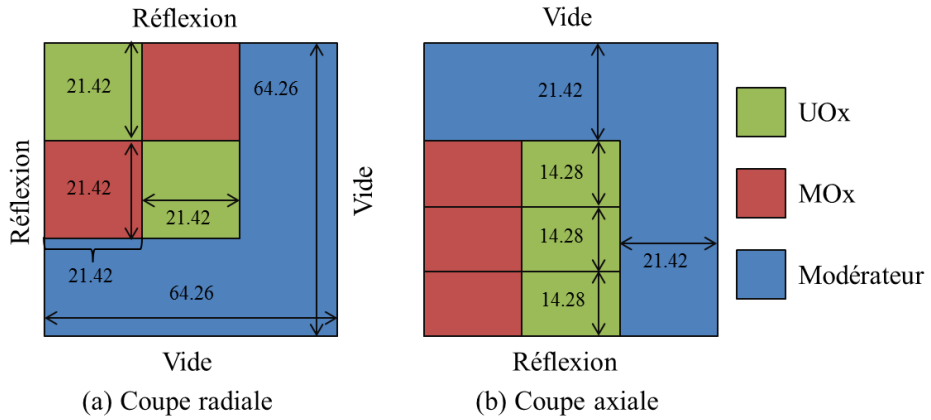


Figure 5.1. Dimensions et positions des assemblages du C5G7.

La coupe radiale du cœur est reportée sur la Figure 5.1(a) : il s'agit de 4 assemblages UOx et MOx entourés d'un réflecteur constitué de modérateur uniquement (eau légère). La description axiale est donnée sur la Figure 5.1(b). Les assemblages sont divisés en trois zones de 14.28 cm, pour une hauteur combustible totale de 42.84 cm. Les 3 zones axiales permettent de décrire les configurations barrées que nous présenterons par la suite.

La géométrie des crayons est donnée sur la Figure 5.2 et le positionnement des crayons dans les assemblages est présenté sur la Figure 5.3.

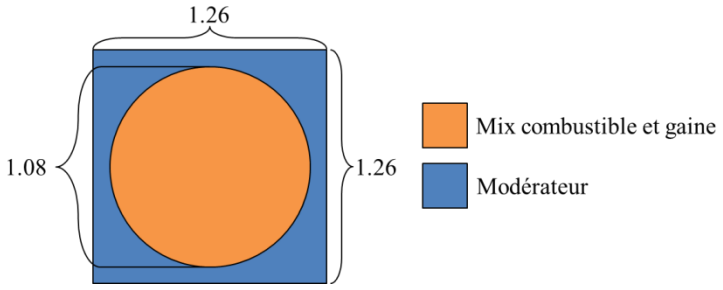


Figure 5.2. Coupe radiale de la modélisation d'un crayon.

Les crayons combustibles sont constitués d'une seule zone homogène (intégrant le combustible et la gaine) et représentés par un cylindre. A l'extérieur du crayon combustible se trouve le modérateur. Ce dernier est décrit par le même matériau que le modérateur présent dans la Figure 5.1 (modérateur à l'extérieur des assemblages combustibles constituant le réflecteur). Les bibliothèques de sections efficaces pour décrire les matériaux sont fournies avec le benchmark. Elles sont données à 7 groupes avec des sections efficaces de scattering prenant en compte des chocs isotropes (degré d'anisotropie P0).

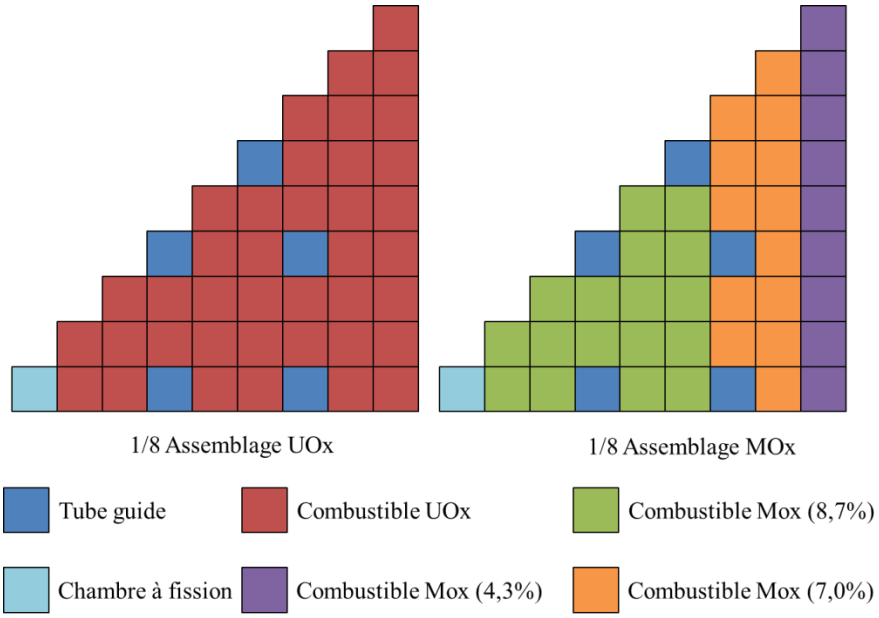


Figure 5.3. Position des crayons dans les assemblages UOx et MOx.

Les assemblages sont composés de 264 crayons combustibles, de 24 tubes guides et de la chambre à fission au centre de l'assemblage. L'assemblage UOx est constitué d'un seul type de crayon combustible alors que pour l'assemblage MOX trois types de crayons avec des teneurs en plutonium différentes sont positionnés dans l'assemblage.

La partie du modérateur située au-dessus du cœur, dans le prolongement des assemblages combustible, est décrite Figure 5.4.

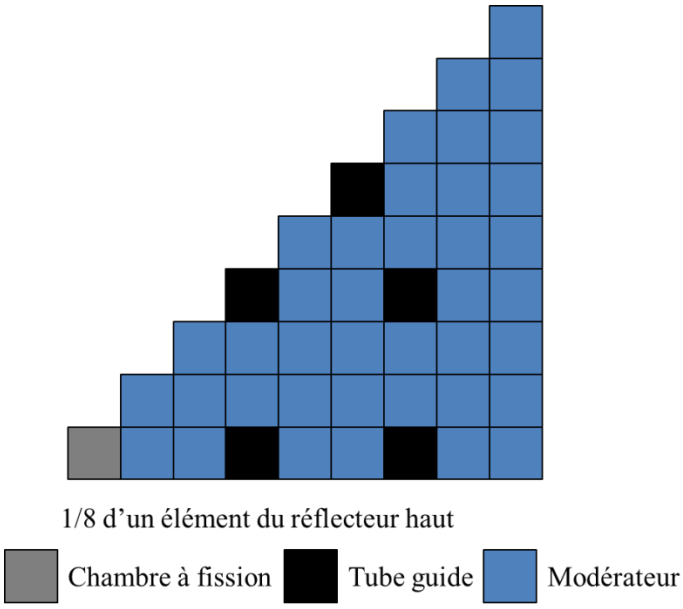


Figure 5.4. Réflecteur au-dessus des assemblages.

Les tubes guides ainsi que la chambre à fission sont prolongés dans la partie haute du réflecteur.

5.2.1 Paramètres de discrétisation du solveur IDT de la Maquette

Dans les assemblages combustibles, la discrétisation radiale correspond à la description géométrique présentée à la Figure 5.2. Chaque cellule est décrite par un cylindre contenant le combustible ou le tube guide homogénéisé, entouré par le modérateur. Axialement, la géométrie est discrétisée en mailles de 3.57 cm, soit 12 plans au total dans le combustible. Nous utilisons une expansion spatiale linéaire du flux ce qui explique l'utilisation d'un nombre limité de mailles.

Le réflecteur est discrétisé par des mailles cartésiennes de 1.26 cm de côté pour la description radiale, ce qui correspond à la taille des cellules dans les assemblages combustibles. La discrétisation axiale est la même que dans les assemblages combustibles, soit des pas de 3.57 cm.

Cependant, afin de représenter correctement le gradient du flux à l'interface cœur-réflecteur, le maillage est raffiné pour les mailles au contact des assemblages combustibles. Ainsi, la première maille est divisée en 2 pour le réflecteur radial et divisée par 3 pour le réflecteur haut.

Le nombre de mailles spatiales est de 3965 pour la configuration 2D et 56258 pour les configurations 3D.

La Figure 5.5 illustre le maillage des surfaces de chaque cellule combustible ou région homogène :

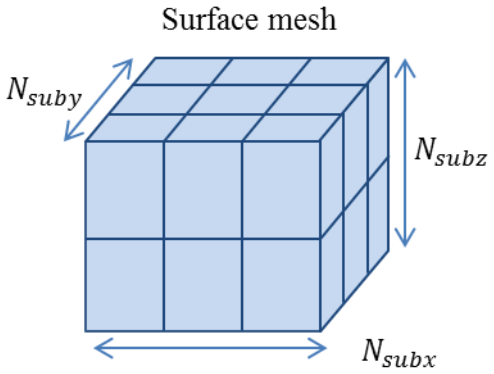


Figure 5.5. Maillage des surfaces d'une cellule en 3D.

Nous avons ainsi 6 mailles pour les surfaces verticales et 9 mailles pour les surfaces horizontales en 3D. En 2D, il y a 3 mailles par surface.

La quadrature angulaire utilisée est une quadrature S_8 à niveaux symétriques pour laquelle les directions sont déplacées vers les axes x et y du plan de la géométrie [55]. Ainsi, il y a 80 directions pour les calculs 3D et 40 pour le calcul 2D.

Le CMFD est défini sur le maillage énergétique à 7 groupes. La discrétisation radiale est constituée d'un maillage régulier de 1.26 cm x 1.26 cm, correspondant à la dimension des cellules. Axialement, les mailles ont une hauteur de 3.57 cm. Le maillage spatial ainsi défini correspond à la discrétisation cartésienne du calcul transport dans les assemblages combustibles.

Le choix de ces paramètres est le résultat d'un compromis temps de calculs et précision des résultats. Les critères de convergence utilisés sont de 10^{-5} sur la valeur propre et le flux, 10^{-4} pour la source de fission et les courants d'interface.

5.2.2 Configuration 2D

La configuration 2D est celle représentée sur la Figure 5.1(a). Pour le calcul DDM, la décomposition est réalisée en 9 sous domaines de la taille d'un assemblage. Le calcul est convergé en 10 itérations externes et une moyenne de 102 itérations internes par sous domaine.

Les résultats obtenus sont présentés dans le Tableau 5.1. L'écart sur la valeur propre est de 6 pcm, ce qui est dans l'intervalle à plus ou moins trois écarts types (que nous nommerons *intervalle de confiance* dans la suite) de la solution de référence du Benchmark. Au niveau du crayon chaud, l'erreur sur le taux de fission est de 4 pcm. L'erreur maximale inférieure à 0.8%. Près de 90% des crayons sont dans l'intervalle de confiance de la solution de référence.

Tableau 5.1. Comparaison référence - calcul Maquette.

	Maquette	Référence
Valeur propre	1.18662	1.18655
Erreur en pcm (écart type associé)	6	3
Ecarts crayons spécifiques		
Taux maximum	2.498	2.498
Erreur en % (écart type associé)	0.004	0.07
Distribution des erreurs en pourcent (écart type associé)		
Erreur maximale	0.792	0.240
Erreur moyenne	0.215	0.139
RMS	0.264	0.145
% du nombre de crayon dans l'intervalle à :		
1 écart type	36%	
3 écarts types	89%	
Taux de fission moyen par assemblage		
UOx intérieur	1.866	1.867
MOX	0.802	0.802
UOx extérieur	0.529	0.529
Erreur en % UOx intérieur	-0.013	
Erreur en % MOX	0.030	
Erreur en % UOx extérieur	-0.045	

Les résultats des participants au benchmark [4] datant de plus de 10 ans, ils ne sont pas très représentatifs de l'état de l'art actuel. Cependant, seules 4 solutions présentent de meilleurs résultats pour la valeur propre, 2 pour l'erreur maximale et 4 pour la fraction des crayons dans l'intervalle de confiance. Les solveurs permettant d'obtenir ces résultats sont principalement basés sur la méthode des caractéristiques. Pour les calculs avec cette méthode, le maillage spatial est très raffiné puisque celles-ci utilisent l'approximation des sources constantes dans chaque région. Ainsi, pour obtenir la précision des résultats présentés, plusieurs dizaines de régions par cellule combustible sont nécessaires. Le solveur IDT utilisé dans la Maquette permet quant à lui un développement spatial linéaire du flux. De ce fait, nous pouvons obtenir des résultats d'une précision similaire avec un nombre de mailles nettement réduit. Les résultats présentés ici ont été obtenus avec seulement 2 mailles par cellule

combustible : le crayon et le modérateur. Avec une telle discrétisation, le temps de calcul est de moins de 4 secondes, parallélisé avec 9 threads en mémoire partagée sur une machine standard (cf. Annexe C).

5.2.3 Configurations 3D de l'extension du Benchmark C5G7

Les configurations 3D se déclinent en 3 versions, présentées sur la Figure 5.6. La première, figure (b), est la configuration non barrée. Les barres sont complètement extraites mais toujours présentes dans la partie du réflecteur haut. Dans la configuration barrée A, sur la figure (c), la barre est insérée dans la zone haute de l'assemblage UOx intérieur. Enfin dans la configuration barrée B, sur la figure (d), la barre est insérée jusque dans la zone centrale de l'assemblage UOx intérieur. De plus une barre est insérée dans la zone haute des assemblages MOx. Aucune barre n'est insérée dans l'assemblage UOx extérieur.

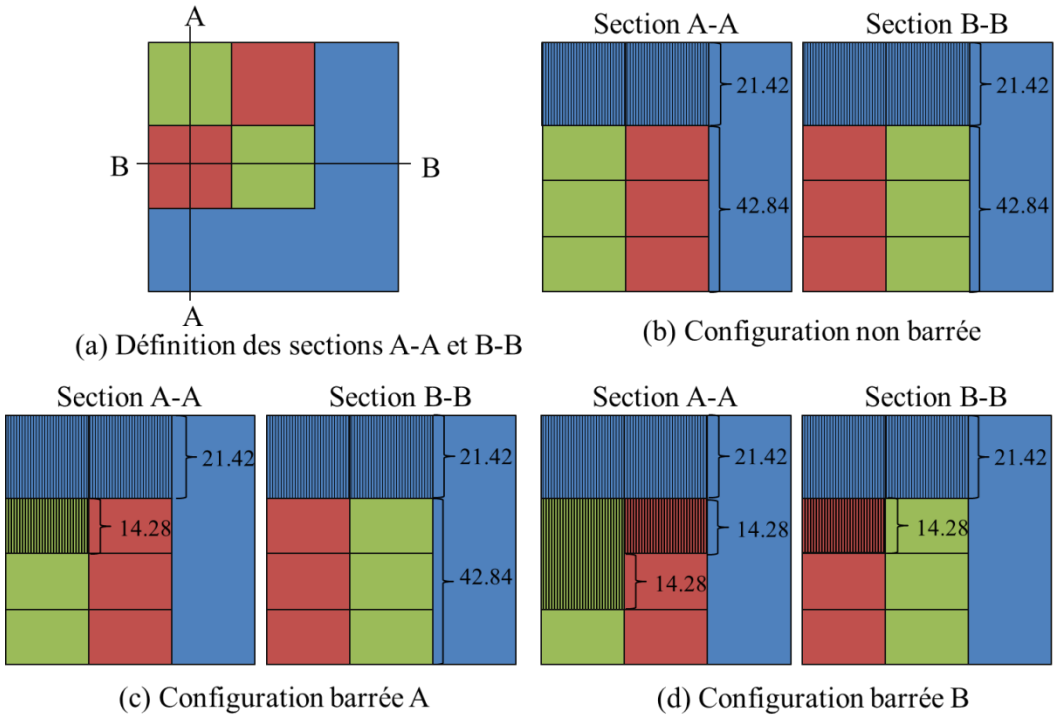


Figure 5.6. Description des différentes configurations 3D.

La décomposition de domaine est réalisée en définissant 36 sous domaines 3D. La Figure 5.7(a) représente les divisions radiales. Cette dernière décomposition est la même que pour le calcul 2D, soit 9 sous domaines. La Figure 5.7(b) représente la décomposition axiale : la géométrie est divisée en 4 sous domaines. Les 3 premières divisions correspondent aux 3 zones de la partie active (zones basse, zones centrale et zones haute) tandis que la dernière division correspond au réflecteur haut.

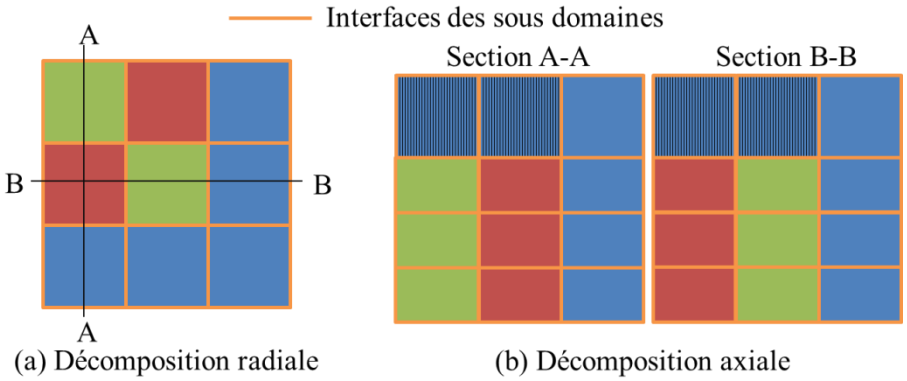


Figure 5.7. Décomposition de domaine pour les calculs 3D.

Le Tableau 5.2 présente le nombre d'itérations ainsi que le temps de calcul pour les 3 configurations. Les calculs ont été réalisés avec 9 threads sur un nœud du cluster de calcul à notre disposition (partition Slim, cf. Annexe C) du cluster de calcul disponible dans notre laboratoire. Les calculs convergent en moins de 25 itérations externes, avec en moyenne 5.5 itérations internes par itération externe. Le temps de calcul, sensiblement le même pour toutes les configurations, est de l'ordre de 15 minutes.

Tableau 5.2. Nombre d'itérations et temps de calcul pour les configurations 3D du C5G7.

Configuration	Nombre d'itérations d'externes	Nombre moyen d'itérations internes par sous domaine	Temps de calcul (s)
Non barrée	25	665	942
Barrée A	25	666	945
Barré B	24	652	920

Pour simplifier l'analyse, nous ne présenterons ici que la configuration barré B pour laquelle les erreurs sont les plus grandes. Cependant, les trois configurations ont été étudiées et les écarts vis-à-vis de la référence Monte-Carlo sont du même ordre de grandeur. Les résultats de la configuration non barrée et de la configuration barrée A sont donnés en Annexe E.

Le Tableau 5.3 synthétise l'ensemble des résultats de calcul sur la configuration barrée B selon les critères demandés par le benchmark : l'écart à la valeur propre et les erreurs sur les taux de fission à l'échelle du crayon pour les différentes zones axiales.

L'écart à la valeur propre de référence est de -22 pcm. Cette écart est plutôt faible comparé aux écarts présentés par les participants au benchmark [5]. En effet, pour ces derniers, les écarts indiqués sont plutôt de l'ordre la centaine de pcm pour la configuration barrée B.

Tableau 5.3. Comparaison référence - calcul Maquette, configuration barre B.

	Maquette	Référence
Valeur propre	1.07753	1.07777
Erreur en pcm (écart type associé)	-22	2.8

	Partie Basse		Partie Centrale		Partie Haute	
	Maquette	Réf.	Maquette	Réf.	Maquette	Réf.
Ecarts crayons spécifiques et erreurs associées						
Taux maximum	1.196	1.200	0.550	0.554	0.215	0.217
Erreur en % (écart type associé)	-0.363	(±0.09)	-0.608	(±0.15)	-0.982	(±0.24)
Distribution des erreurs en %						
Erreur maximale	1.388	0.240	1.284	0.260	1.711	0.380
Erreur moyenne	0.322	0.146	0.372	0.181	0.467	0.285
RMS	0.416	0.150	0.455	0.184	0.586	0.290
Nombre de crayon dans l'intervalle à :						
1 écart type	28%		29%		39%	
3 écarts types	75%		75%		83%	
Taux de fission moyen par assemblage et erreurs en %						
UOx intérieur	247.29	247.75	105.99	106.56	40.82	41.12
MOX	126.12	125.78	81.44	81.41	29.44	29.42
UOx extérieur	91.93	91.64	65.20	65.02	30.77	30.68
Erreur UOx intérieur (écart type associé)	-0.185	(±0.091)	-0.534	(±0.056)	-0.744	(±0.035)
Erreur MOX (écart type associé)	0.266	(±0.073)	0.037	(±0.058)	0.067	(±0.034)
Erreur UOx extérieur (écart type associé)	0.318	(±0.055)	0.275	(±0.046)	0.304	(±0.032)

Les écarts sur les taux de fission sont plus élevés en comparaison des meilleurs résultats obtenus par les participants au benchmark. Au point chaud (le crayon où le taux de fission est maximal), notre solution est entre la 6ème et la 9ème valeur sur les 14 participants, selon la zone axiale considérée. Concernant le nombre de crayons dans

l'intervalle de confiance du calcul Monte Carlo de référence, seuls 4 participants ont de meilleurs résultats que ceux obtenus ici. Il en est de même pour l'erreur maximale par zone axiale.

Cependant, nous pouvons noter que le compromis temps de calcul/précision est nettement plus élevé avec la Maquette-DDM en comparaison de celui correspondant aux meilleurs résultats des participants au benchmark. En effet, nous n'utilisons qu'une quadrature S_8 et moins de 60 000 mailles spatiales, ce qui est au moins un ordre de grandeur inférieur aux discrétisations utilisées par les participants.

ANALYSE DE PERFORMANCES

Au point 3 de la section 2.3.3 (page 45) nous avons émis l'hypothèse que l'algorithme PMBJ épargnait du temps de calcul en permettant à chaque sous domaine d'adapter localement le nombre d'itérations internes. Ce phénomène est observable lorsque le rayon spectral et le maillage diffèrent entre les sous domaines. C'est en l'occurrence le cas avec le C5G7. En effet, les assemblages combustibles sont plus maillés que les sous domaines du réflecteur et donc chaque itération est plus coûteuse comme le montre la Figure 5.8. Il s'agit d'un histogramme 3D : chaque groupe de 3 x 3 barres représente une zone active du cœur.

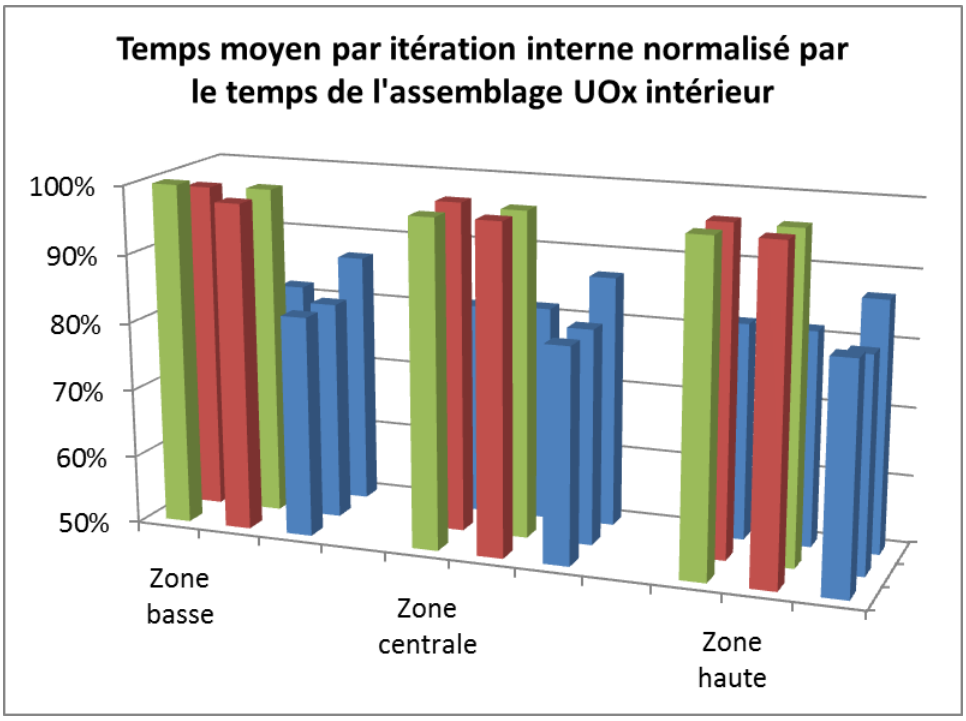


Figure 5.8. Temps moyen par itération interne normalisé par le temps de l'assemblage UOx intérieur.

Le temps d'une itération interne d'un sous domaine du réflecteur est de l'ordre de 80% du temps d'une itération interne d'un sous domaine composé d'assemblage combustible. Cependant, le rayon spectral des sous domaines contenant le réflecteur, composé d'eau, est très supérieur à celui des assemblages combustibles du fait que le rapport $c = \Sigma_{s0}/\Sigma_t = 0{,}985$ est proche de 1. Cela est vérifié, comme le montre la

Figure 5.9 : le nombre d'itérations pour les sous domaines dans le réflecteur est multiplié par un facteur de l'ordre de 1.4 par rapport à l'assemblage UOx central.

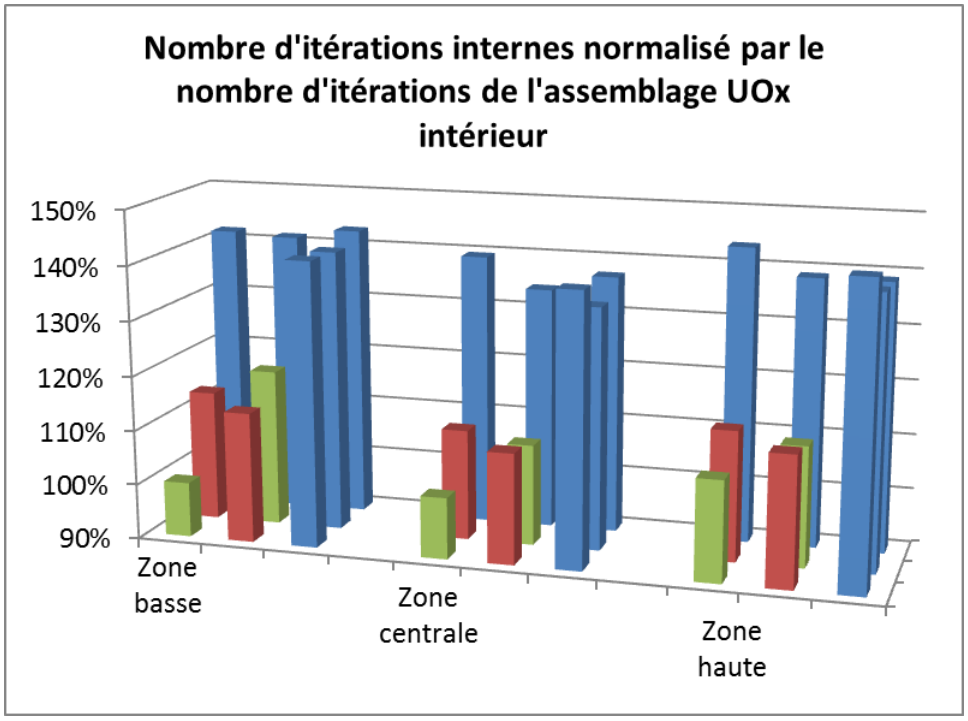


Figure 5.9. Nombre d'itérations internes normalisé par le nombre d'itérations de l'assemblage UOx intérieur.

Ainsi, en adaptant localement le nombre d'itérations internes, le temps de calcul est équilibré entre les différents sous domaines, comme le montre la Figure 5.10.

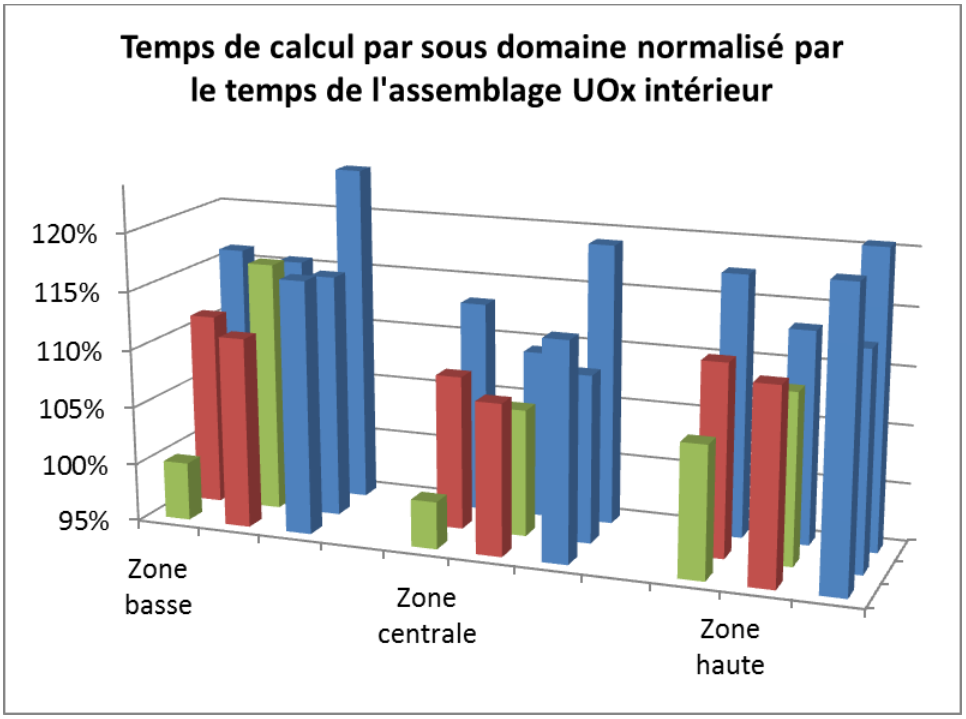


Figure 5.10. Temps de calcul par sous domaine normalisé par le temps de l'assemblage UOx intérieur.

Le temps de calcul effectif est donné par le temps le plus élevé parmi les sous domaines comme présenté à l'équation (2.16). Le déséquilibrage de charge est alors de 24% à cause du sous domaine du réflecteur en coin. Ce déséquilibrage apparait avec l'algorithme PMBJ mais est caché lorsque la décomposition de domaine est appliquée seulement au balayage spatio-angulaire (algorithmes que nous avions référencé par PB1G) comme nous l'avons montré au point 3 de la section 2.3.3. En effet, dans les algorithmes PB1G, le nombre d'itérations dans un groupe est le même pour chaque sous domaine et contraint par le rayon spectral le plus grand. Le temps de calcul d'une itération interne est celui du sous domaine le plus coûteux. En utilisant l'estimation de l'équation (2.14), le temps de calcul serait alors supérieur de 16% par rapport à celui de l'algorithme PMBJ.

5.2.4 Conclusion

Les résultats obtenus avec le solveur de la Maquette DDM sont satisfaisants. Ils offrent un très bon compromis entre précision (les résultats étant placés parmi les meilleurs des participants au benchmark) et coût de calcul. En effet, le développement linéaire du flux dans les cellules permet de réduire considérablement le nombre de mailles spatiales.

Les discrétisations spatiales et angulaires utilisées pour le calcul des différentes configurations du C5G7, présentées au §5.2.1, ont donné de bons résultats. De ce fait nous les utiliserons pour les configurations de calcul proposées par la suite.

5.3 Calcul 2D d'un grand cœur de REL avec baffle lourd dans un état début de cycle

Une première application dite « réaliste » est présentée dans cette section. Il s'agit d'une modélisation en 2D d'un grand cœur de REL avec baffle lourd. Ce cœur est constitué d'un plan de chargement correspondant à un état en Début De Cycle (DDC). Aucune simplification n'est adoptée pour la description géométrique et physique de la partie active du cœur : chaque cellule (cf. §1.5.1) est décrite exactement c'est-à-dire avec le combustible, la gaine et le modérateur. Les objectifs de cette analyse sont doubles :

- D'une part, il s'agit d'évaluer la précision des résultats du solveur de la Maquette-DDM sur une configuration représentative d'un grand cœur de REL en les comparant aux résultats de référence obtenus avec le code de Monte Carlo TRIPOLI-4® [6].
- D'autre part, il s'agit d'évaluer les performances de la méthode en termes de temps de calcul et d'utilisation des ressources mémoires pour deux états du cœur : un état DDC et un état représentatif d'un point d'évolution (la différence entre les deux étant le nombre de matériaux, c'est-à-dire le nombre de jeux de sections efficaces).

La première partie de cette section décrit les grandeurs d'intérêt et les erreurs associées vis-à-vis du calcul de référence TRIPOLI-4®. Ensuite, seront décrites les caractéristiques technologiques du cœur puis les hypothèses de modélisation retenues assurant la meilleure précision ainsi que le modèle utilisé pour générer les sections efficaces macroscopiques. Le calcul de l'assemblage en milieu infini est accompagné d'éléments de validation à cette même échelle (comparaison aux résultats de référence Monte-Carlo). Les résultats du calcul de cœur seront alors confrontés aux résultats de référence. Nous comparerons ainsi la valeur propre et la distribution des sources de fission à l'échelle de la cellule. Enfin, dans une dernière partie, les performances de la maquette sont présentées pour le calcul en DDC du cœur 2D. Ce calcul est réalisé dans une configuration similaire à celle d'un calcul d'évolution, c'est-à-dire en prenant en compte un nombre de matériaux permettant de faire évoluer distinctement chaque zone (couronne) des crayons combustibles dans le cœur.

5.3.1 Calcul de référence dans le code de Monte Carlo TRIPOLI-4®

Une simulation Monte Carlo consiste à calculer des valeurs moyennes, taux de fission par exemple, en simulant l'histoire des neutrons. Elle peut être considérée comme référence puisqu'elle ne requiert a priori aucune approximation de discrétisation spatiale (3D exact) et énergétique (sections efficaces ponctuelles à l'état de l'art). De plus, à la valeur moyenne calculée est associée une incertitude statistique caractérisant l'intervalle de confiance. Cet intervalle de confiance est représenté par l'écart type, racine carrée de la variance de grandeur calculée par la simulation. L'écart type est noté σ dans la suite de ce document. L'une des propriétés de l'écart

type est qu'il y a 99.7% de chances que la grandeur à estimer soit dans l'intervalle $[\bar{x} - 3\sigma; \bar{x} + 3\sigma]$, où $\bar{x}$ est la valeur moyenne. Dans la suite, le terme *incertitude* associée à un résultat issu d'un calcul Monte Carlo fera référence à l'intervalle de confiance à 3σ.

Les sections efficaces multigroupes utilisées par les codes déterministes (dans notre cas le code APOLLO2 pour produire les sections autoprotégées) sont issues de l'évaluation JEFF3.1.1 [8]. La bibliothèque d'application produite pour le code TRIPOLI-4® utilise des données nucléaires de la même origine. Ainsi, l'origine des données nucléaires et le processus de traitement de ces données pour la production des bibliothèques d'application pour les codes déterministes et Monte-Carlo sont totalement cohérents.

Dans le cadre de l'analyse, nous comparons la distribution du taux de fission et la valeur du k_{eff} avec le calcul de référence de la manière suivante :

$$\Delta\tau(r) = \frac{\tau(r) - \tau_{T4}(r)}{\tau_{T4}(r)} \; et \; \Delta k_{eff} = \frac{\lambda - \lambda_{T4}}{\lambda_{T4}},$$

où $\tau(r)$ et λ sont le taux de fission dans la région r et la valeur propre du solveur déterministe et $\tau_{T4}(r)$ et λ_{T4} le taux de fission dans la région r et la valeur propre de TRIPOLI-4®. Pour les comparaisons, les distributions spatiales des taux de fission sont normalisées à l'unité.

5.3.2 Description physique et technologique du cœur

La Figure 5.11 représente 1/8ème du plan de chargement du cœur [62]. Le cœur complet est constitué de 241 assemblages combustibles entouré par un réflecteur lourd.

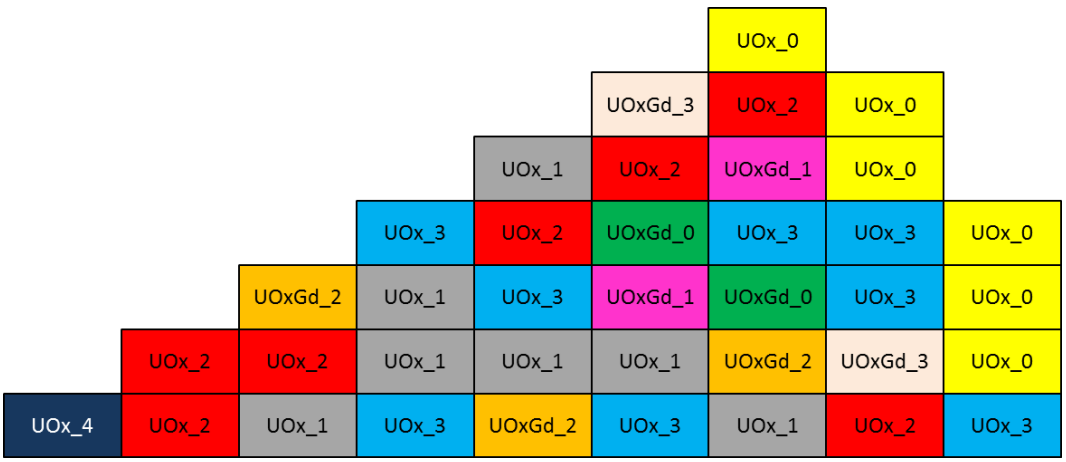


Figure 5.11. Description du chargement du combustible (Quart Nord Est du cœur).

Deux types d'assemblages composent le plan de chargement de ce cœur : les assemblages UOx et les assemblages UOx-Gd contenant des crayons gadolinium. Le numéro qui accompagne la désignation du type d'assemblage correspond au nombre de cycles pendant lesquels l'assemblage a séjourné dans le cœur. Le taux de combustion associé est donné dans le Tableau 5.4.

Tableau 5.4. Taux de combustion des assemblages.

	Taux de combustion (MWj/t)	
Cycle	UOx	UOx-Gd
0	0	0
1	14198	19060
2	31797	33997
3	45634	48982
4	56225	

Ces taux de combustion permettent de représenter le plan de chargement d'un cœur en DDC avec une gestion du combustible par quart (4 lots d'assemblages).

Les assemblages combustibles (Figure 5.12) sont constitués d'un réseau régulier de 17 x 17 crayons avec 24 tubes guides (TG). Les assemblages UOx-Gd disposent de 12 crayons gadolinium (Gd). Il y a 265 crayons combustibles standards sans poison dans les assemblages UOx et 253 dans les assemblages UOx-Gd. La longueur du côté d'un assemblage est 21.455 cm.

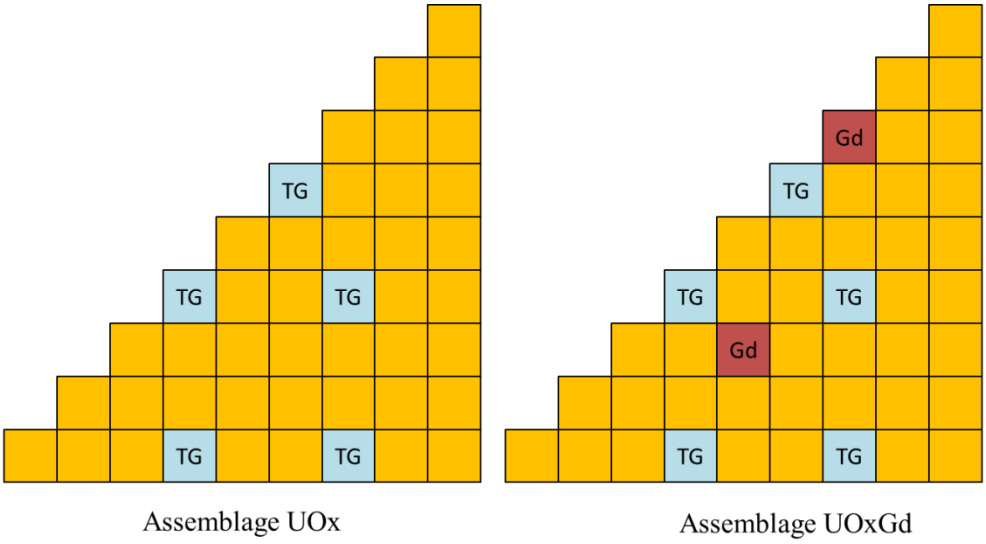


Figure 5.12. Caractéristiques des assemblages UOX et UOx-Gd (géométrie 1/8ème).

Les dimensions des crayons et tubes guides des assemblages sont les données technologiques standard pour les réacteurs à eau pressurisés. L'enrichissement en ^{235}U des assemblages UOx et UOx-Gd utilisés pour cette modélisation est de 4.9%. Le calcul est réalisé avec une concentration en bore de 1600 ppm et la densité moyenne de l'eau dans le cœur est de 0.7243 g/cm^3. Les températures combustible et modérateur sont décrites Tableau 5.5. Pour l'analyse, nous ferons l'hypothèse d'un cœur isotherme.

Tableau 5.5. Températures des différents milieux du Cœur.

	T (°C)	T (°K)
Combustible	501	774
Gaine	351	624
Modérateur	301	574

5.3.3 Modélisation des assemblages et du réflecteur

Les assemblages sont constitués d'un réseau de 17 x 17 crayons et modélisés sans lame d'eau en périphérie. Les cellules sont décrites finement avec le modérateur, la gaine et le combustible. Les crayons UOx sont modélisés avec 4 couronnes et les crayons UOx-Gd avec 11 couronnes (discrétisation spatiale nécessaire pour l'autoprotection). La Figure 5.13 présente la géométrie des crayons UOx, UOx-Gd et des tubes guides utilisée pour modéliser les assemblages constituant le cœur.

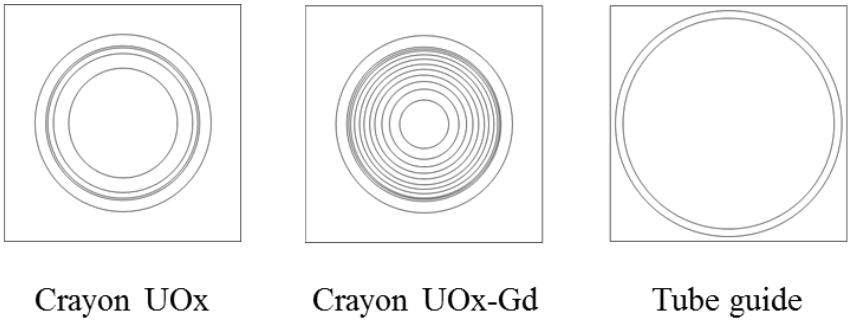


Figure 5.13. Modélisation des crayons UOx, UOx-Gd et des tubes guides.

Pour des raisons de simplification et afin de s'affranchir d'éventuels biais de modélisation géométrique entre déterministe et Monte-Carlo, le réflecteur lourd a été modélisé par un milieu homogène composé à 95% d'acier et 5% d'eau borée à la même concentration que celle du cœur. Notons toutefois que les proportions en eau sont représentatives d'un réflecteur lourd standard et que l'objectif de représentativité des effets de spectre à l'interface cœur-réflecteur est atteint malgré cette hypothèse d'homogénéisation des matériaux. Le réflecteur lourd environnant le cœur est discrétisés en blocs carrés de la taille d'un assemblage. Ces éléments sont positionnés tout autour du cœur et forment une couronne autour des assemblages combustibles. Finalement, de l'eau borée complète la géométrie extérieure au réflecteur de manière à former un maillage cartésien global. La partie extérieure au cœur de la modélisation est représentée Figure 5.14.

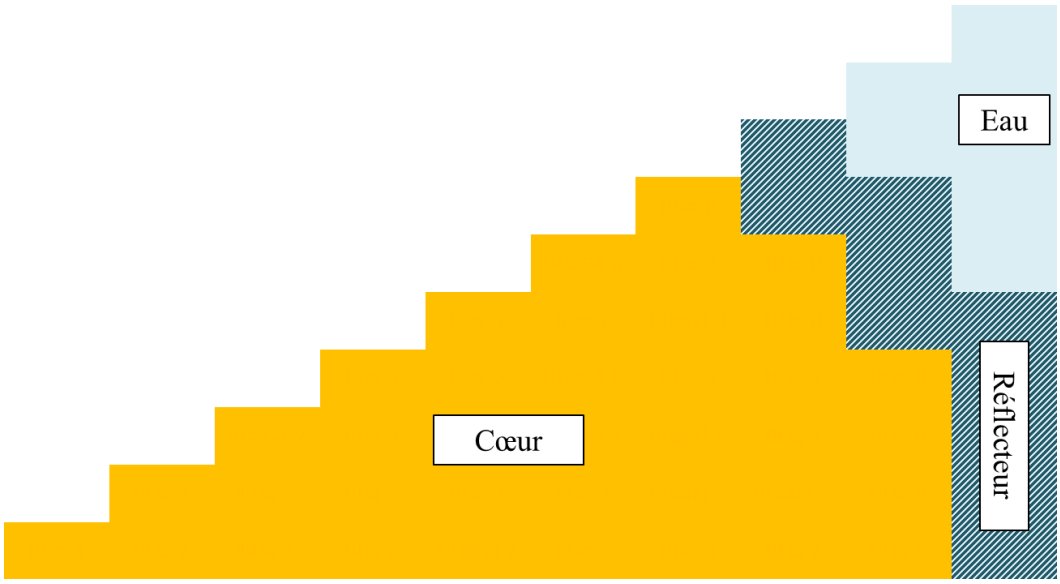


Figure 5.14. Modélisation de la géométrie extérieure au cœur.

Les simplifications qui ont été apportées dans cette modélisation ont pour but de limiter la complexité des données d'entrée pour la Maquette-DDM et ainsi les erreurs potentielles. Des conditions aux limites de vide sont imposées sur les surfaces externes du cœur. Afin d'assurer la cohérence entre le modèle de référence dans le code de Monte Carlo TRIPOLI-4® et la modélisation adoptée avec la Maquette-DDM, les hypothèses concernant la géométrie et la composition isotopique des matériaux sont appliquées de manière identique aux deux simulations.

5.3.4 Génération des bibliothèques de sections efficaces du calcul déterministe et validation du schéma à l'échelle assemblage

L'objectif de la première étape de calcul est de fournir un ensemble de sections efficaces multigroupes autoprotégées qui seront utilisées pour le calcul à l'échelle du cœur. Ce calcul d'autoprotection des sections efficaces a été réalisé par assemblage en milieu infini avec le code de transport APOLLO2. Pour ce calcul, un regroupement des cellules a été effectuée (cf. Figure 5.15) : 7 types de cellules permettent de décrire un assemblage UOx et 9 types de cellules pour un assemblage UOx-Gd. Avec ces regroupements, le nombre de matériaux (c'est-à-dire le nombre de jeux de sections efficaces) utilisés pour décrire un assemblage UOx est de 26 et 41 pour un assemblage UOx-Gd. On dispose ainsi d'un total de 292 matériaux (jeux de sections) pour la description des 5 assemblages UOx et des 4 assemblages UOx-Gd présents dans le cœur. Les sections efficaces ont été autoprotégées sur un maillage à 281 groupes d'énergie basé sur le maillage SHEM [61] jusqu'à l'ordre P3 pour le scattering, sans homogénéisation spatiale. La discrétisation spatiale est celle présentée sur la Figure 5.13.

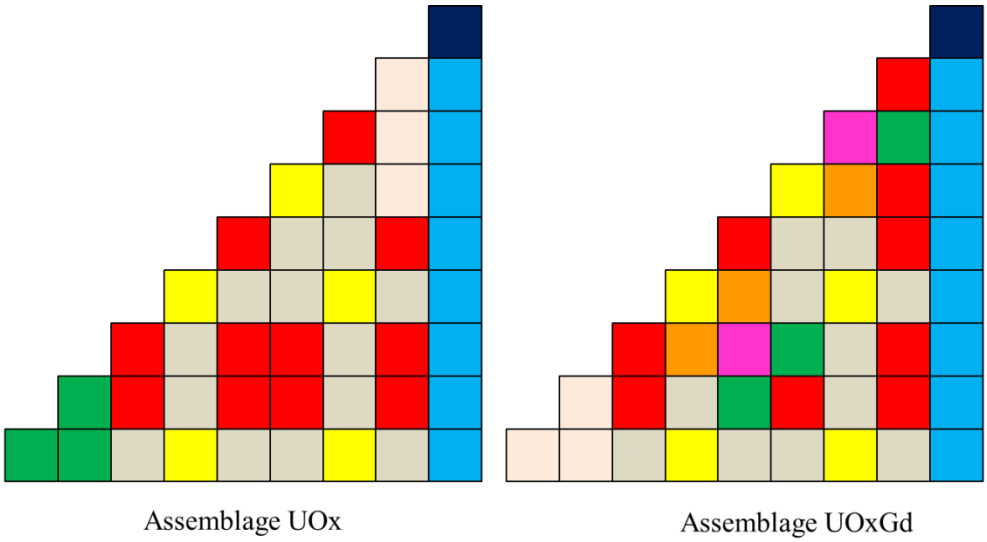


Figure 5.15. Typage des cellules des assemblages UOx et UOx-Gd.

Le calcul d'évolution des concentrations isotopiques a été réalisé avec le code APOLLO2. Le flux dans l'assemblage est calculé avec le solveur MOC [16] sur 1/8ème de la géométrie de l'assemblage. Le schéma de production des sections efficaces pour les assemblages aux différents taux de combustion répertoriés est composé de 4 étapes : calcul d'évolution isotopique, calcul de reprise avec autoprotection, calcul du nouveau flux avec le solveur MOC, édition des sections efficaces. Des pas intermédiaires ont été utilisés entre les valeurs des taux de combustion des assemblages présents dans le cœur afin de suivre correctement l'évolution des concentrations isotopiques. Les sections efficaces sont éditées jusqu'à l'ordre P3.

Les sections efficaces du réflecteur ont été obtenues par un calcul de traverse 1D comprenant une partie combustible de 180 cm modélisée par un assemblage UOx neuf homogénéisé discrétisé en 500 mailles et le réflecteur homogène sur 21.455394 cm discrétisé en 50 mailles. Un seul milieu réflecteur a été généré pour tout le cœur.

Ces calculs permettent aussi d'extraire les concentrations isotopiques de chaque matériau en vue de leur utilisation dans le modèle de référence dans le de code de Monte Carlo.

VALIDATION DE L'ETAPE DE PRODUCTION DES SECTIONS MULTIGROUPES A L'ECHELLE ASSEMBLAGE

Afin de valider numériquement les sections efficaces multigroupe autoprotégées, nous avons comparé le résultat produit par le solveur IDT utilisant ces sections avec un calcul de référence réalisé avec TRIPOLI-4® à l'échelle de l'assemblage. La simulation Monte Carlo est obtenue avec 10^8 histoires neutrons pour chaque assemblage. L'incertitude associée à la distribution de puissance crayon par crayon est inférieure à 1% à 3 écarts types. Celle sur la valeur propre est inférieure au pcm.

Lors de cette première étape de validation, nous avons cherché la meilleure réponse du solveur IDT par rapport à plusieurs paramètres : quadrature angulaire et ordre de la quadrature, flux constant ou linéaire par maille, sous maillage des surfaces des

cellules, anisotropie des sections, et nous avons retenu deux critères de sélection : le cout du calcul et la précision. Les paramètres retenus sont : quadrature S_8, développement linéaire du flux, 3 sous mailles par surfaces et développement des sections de transfert jusqu'à l'ordre P3 : les résultats correspondants sont présentés dans le Tableau 5.6 pour les assemblages UOx et dans le Tableau 5.7 pour les assemblages UOx-Gd.

Tableau 5.6. Comparaisons IDT – TRIPOLI-4® (T4) pour les assemblages UOx en milieu infini.

	UOx 0	UOx 1	UOx 2	UOx 3	UOx 4
Ecart k_∞ assemblage (pcm)	-60	-28	23	31	15
Ecart Max Crayon (%)	0.6	0.7	0.6	0.7	0.9
Nombre de cellules dans l'intervalle à ± 3σ T4	100%	100%	100%	100%	99%
3 σ T4	<0.69%	<0.75%	<0.78%	<0.84%	<0.96%

Tableau 5.7. Comparaisons IDT – TRIPOLI-4® (T4) pour les assemblages UOx-Gd en milieu infini.

	UOx-Gd 0	UOx-Gd 1	UOx-Gd 2	UOx-Gd 3
Ecart k_∞ assemblage (pcm)	-255	-1	5	23
Ecart Max Crayon (%)	0,8	0,8	0,7	0,7
Nombre de cellules dans l'intervalle à ± 3σ T4	97%	99%	99%	98%
3 σ T4	<1,00%	<0,73%	<0,78%	<0,85%

Les différences en réactivité sont plus importantes pour les assemblages neufs, en particulier pour l'assemblage UOx-Gd 0 avec -255 pcm. Pour les assemblages usés, l'erreur est inférieure ou égale à 30 pcm. Concernant la distribution du taux de fission, au moins 97% des cellules combustibles ont un écart inférieur à l'incertitude statistique du calcul de référence dans le cas le plus défavorable.

La Figure 5.16 présente la distribution des écarts pour l'assemblage UOx-Gd 0. Le taux de fission dans les crayons gadolinium (cellules avec un rectangle noir sur la Figure 5.16) est bien calculé malgré un fort gradient de flux. Les taux calculés par IDT sont compris dans la plage d'incertitude à 3σ.

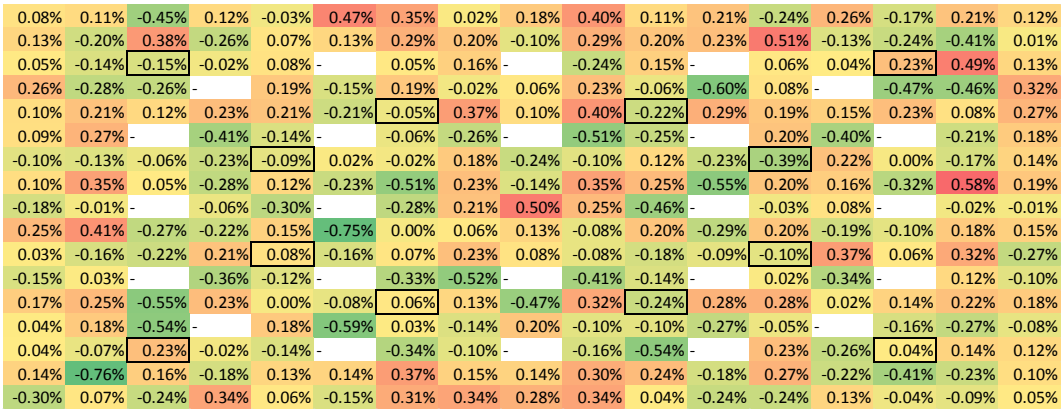


Figure 5.16. Distribution des écarts sur les taux de fission entre TRIPOLI-4® et IDT pour l'assemblage UOx-Gd 0.

Les résultats obtenus à l'échelle de l'assemblage nous permettent de confirmer la cohérence des voies de modélisation entre le calcul de référence dans le code de Monte-Carlo TRIPOLI-4® et le calcul déterministe avec le solveur IDT.

5.3.5 Mise en œuvre de la modélisation cœur 2D dans le code de transport Monte-Carlo TRIPOLI-4®

A l'échelle du cœur en 2D, les comparaisons entre la solution déterministe et la solution Monte-Carlo de référence portent sur la valeur propre et la distribution du taux de fission 2D à l'échelle du crayon combustible. L'objectif d'accéder, pour la solution de référence, à un niveau de précision satisfaisant à l'échelle du crayon combustible implique une bonne maitrise des paramètres de simulation (convergence des sources, calcul de la variance,…). Pour les calculs avec le code de Monte-Carlo TRIPOLI-4®, nous avons utilisé la méthode des *répliques indépendantes* afin de calculer les valeurs moyennes ainsi que les variances associées à celles-ci. Cette méthode permet de prendre correctement en compte les corrélations batch à batch. Dans la pratique, le calcul consiste à réaliser un grand nombre de simulations indépendantes pour lesquelles le générateur de nombres aléatoires est initialisé différemment. Les valeurs moyennes et les variances associées sont alors calculées directement à partir des valeurs fournies par chacune des simulations.

Pour l'exercice sur le cœur 2D, nous avons ainsi réalisé 1800 simulations indépendantes. Pour chaque simulation indépendante, ce sont 1700 batchs actifs de 10^4 particules qui ont été simulés (retrait des 200 premiers batchs pour s'assurer de la convergence des sources) soit un total de 17.10^6 histoires de neutrons par simulation (30.6 milliards de neutrons ont été simulés au total en cumulant les 1800 simulations indépendantes). La distribution de puissance 2D à l'échelle du crayon est représentée Figure 5.17.

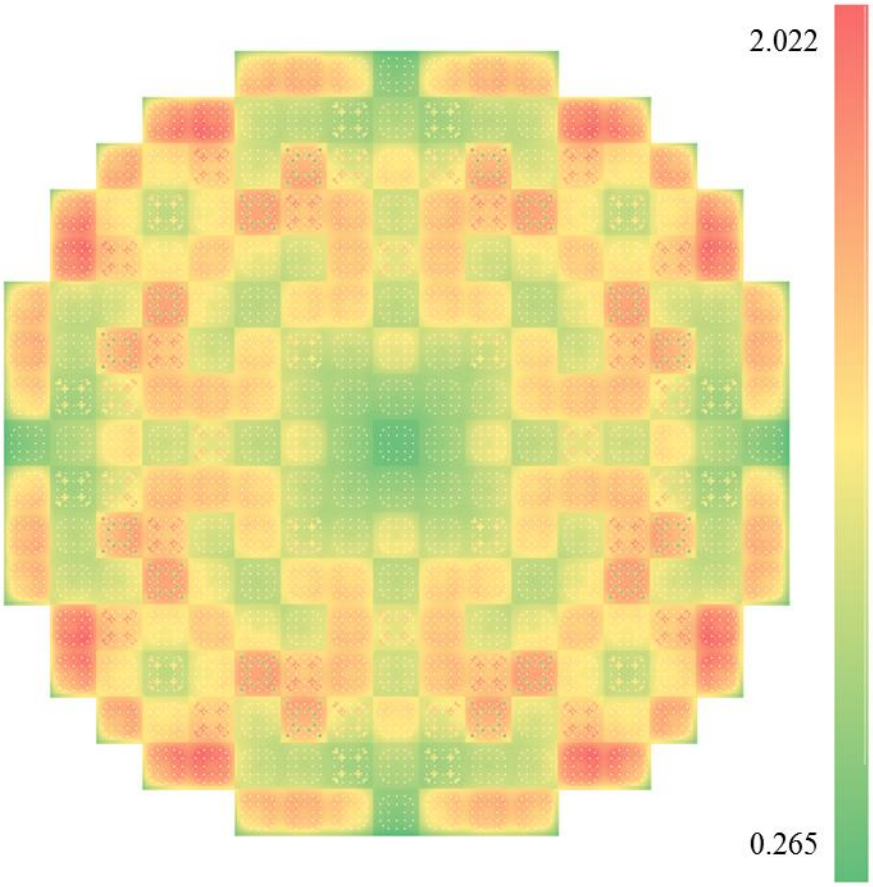


Figure 5.17. Distribution de puissance 2D à l'échelle du crayon (TRIPOLI-4®).

Le Tableau 5.8 présente la valeur propre ainsi que le maximum et le minimum du taux de fission. L'incertitude statistique sur la valeur propre est de 2 pcm. Pour la distribution de taux de fission, l'incertitude par crayon maximale à 3σ est inférieure à 2%.

Tableau 5.8. Convergence statistique du calcul de cœur avec TRIPOLI-4®.

k-effectif	1.02024
3σ (pcm)	2.1
Taux fission Moyen	1.00
Taux fission Minimal	0.2670
Taux fission Maximal	2.014
3σ Maximum	1.95 %

5.3.6 Calcul de cœur 2D avec la Maquette-DDM et comparaison au calcul de référence

5.3.6.a Discrétisation retenue pour le solveur Maquette-DDM

Les paramètres de calculs pour la Maquette-DDM déterminés dans le paragraphe 5.3.4 (quadrature S_8, développement linéaire du flux, 3 sous mailles par surface (cf. Figure 1.2) et développement des sections de transfert jusqu'à l'ordre P3) pour le calcul

d'assemblage sont utilisés pour le calcul de cœur. La discrétisation du calcul de cœur pour la Maquette-DDM est présenté Figure 5.18 pour le 1/8ème du cœur (dans la pratique, le calcul est réalisé sur le cœur complet afin d'évaluer les capacités de la maquette).

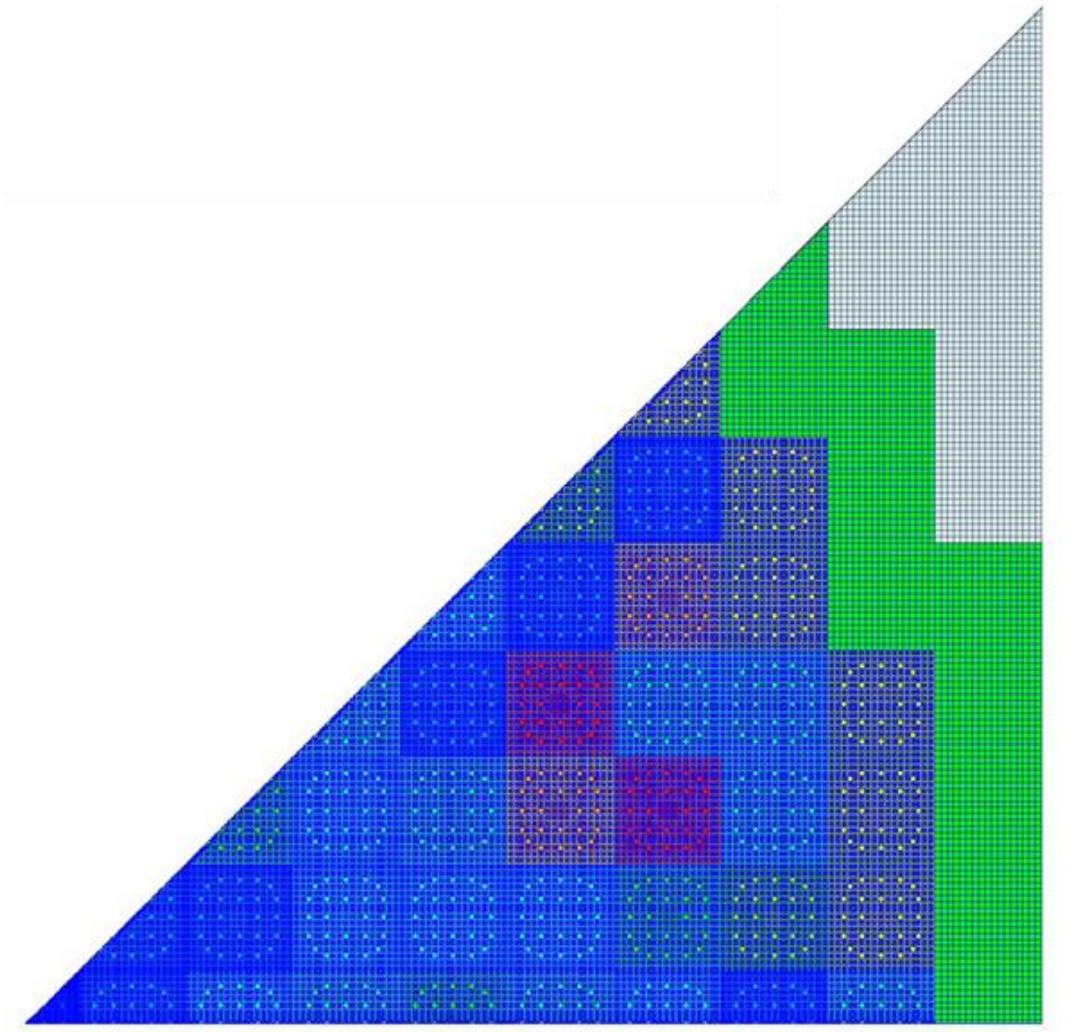


Figure 5.18. Maillage de calcul Maquette-DDM représenté sur 1/8ème du cœur.

Le réflecteur et l'eau servant à compléter le motif sont discrétisés par un réseau cartésien de 17 x 17 mailles. Le calcul est réalisé sur la géométrie du cœur complet. Le nombre de régions est de ~400,000.

Le domaine de calcul global est décomposé en 361 sous domaines, chaque sous domaine étant de la taille d'un assemblage combustible. Ainsi 241 sous domaines sont des assemblages combustibles et 120 sont des éléments du réflecteur.

Les calculs ont été réalisés avec un développement des sections de transfert jusqu'à l'ordre P1 et P3 afin de mettre en évidence l'effet de l'anisotropie de la source de scattering, en particulier à l'interface cœur-réflecteur.

Dans un premier temps, l'analyse portera sur la comparaison des résultats à ceux du calcul de référence. Dans un second temps nous présenterons les performances de la Maquette vis-à-vis du temps de calcul et de l'occupation mémoire.

5.3.6.b Comparaison du calcul Maquette-DDM au calcul de référence

Les résultats des comparaisons sont présentés dans le Tableau 5.9, la Figure 5.20 et la Figure 5.19. Le Tableau 5.9 récapitule les écarts en réactivité ainsi que l'écart maximal sur le taux de fission 2D à l'échelle du crayon pour les deux bibliothèques de sections à l'ordre P1 et P3.

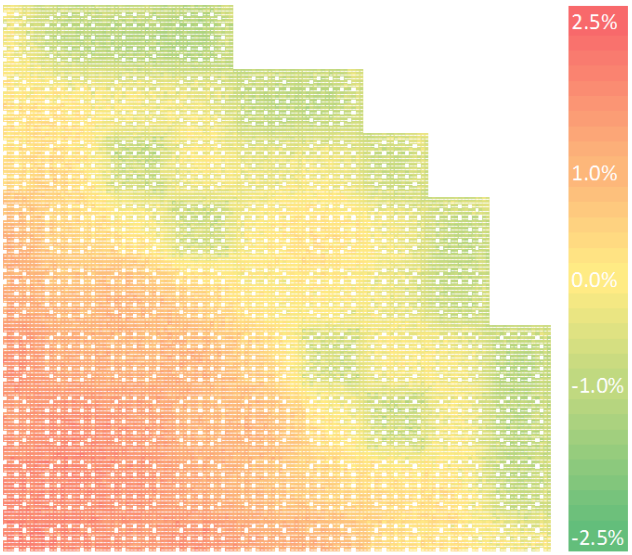
Tableau 5.9. Calcul cœur : comparaison calcul TRIPOLI-4® (T4) - calcul Maquette [63].

	Sections à l'ordre P1	Sections à l'ordre P3
ECARTS VALEUR PROPRE		
Ecart keff (pcm)	-48	-10
3σ T4 (pcm)	2.1	
ECARTS SUR LA DISTRIBUTION DES TAUX DE FISSION 2D CRAYON		
Nombre de crayons combustibles pour lesquels l'écart est inférieur à 3σ T4	51.8%	91.4%
3σ T4 Maximum	<1,95%	
Ecart Max Crayon	2.30%	0.89%
Ecart Min Crayon	-1.49%	-1.42%
RMS	0.76%	0.32%
POINTS CHAUDS		
Taux fission Crayon Max	1.993	2.020
Ecart Taux fission T4 Crayon Max	-0.33%	0.04%
3σ T4 Crayon Max	0.78%	

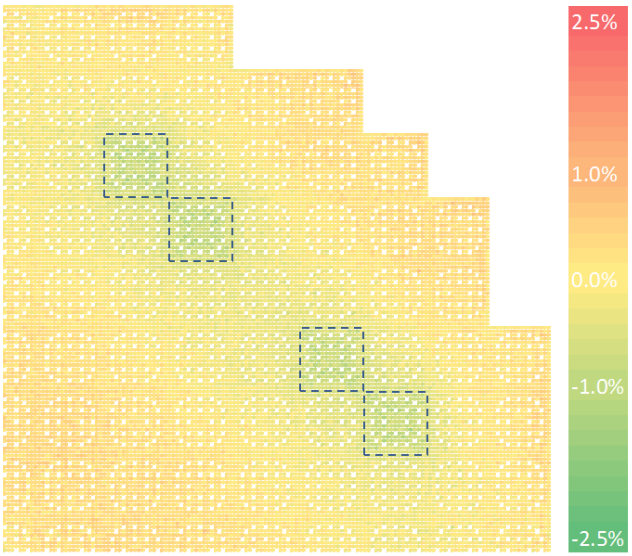
L'écart en réactivité est supérieure à l'incertitude statistique du résultat TRIPOLI-4® pour le calcul P1 comme pour le calcul P3, respectivement à -48 pcm et -10 pcm mais ces écarts à la référence Monte-Carlo sont néanmoins très satisfaisants.

Les écarts entre les calculs P1 et P3 sont plus marqués pour le calcul de la distribution 2D des taux de fission. Pour le calcul P1, 51% des cellules sont dans l'intervalle à ±3σ du calcul de référence (intervalle de confiance). Cependant l'erreur maximum est de 2.30%. En augmentant l'ordre d'anisotropie (ordre P3), 91% des cellules sont dans l'intervalle de confiance du calcul de référence. De plus l'erreur maximale est réduite à 1.42%.

La Figure 5.19 présente la distribution des écarts avec TRIPOLI-4® pour les calculs P1 et P3. Les zones les plus sensibles sont à proximité de l'interface avec le réflecteur ainsi que dans les assemblages gadolinium neufs.



(a) Ecart TRIPOLI4 – Maquette Sections à l'ordre P1



(b) Ecart TRIPOLI4 – Maquette Sections à l'ordre P3

Figure 5.19 Ecarts sur la distribution du taux de fission par rapport à TRIPOLI-4® pour les calculs Maquette P1 et P3.

Pour le calcul à l'ordre P1, on observe un déséquilibre entre le centre du cœur et la périphérie. L'erreur est négative au centre et positive à proximité du réflecteur. Lorsque le calcul est réalisé avec un ordre d'anisotropie P3, cet écart centre-périphérie disparait et l'erreur est ainsi diminuée. Augmenter l'ordre d'anisotropie du scattering permet de réduire l'erreur à l'interface cœur-réflecteur. Cependant dans les deux cas, les assemblages UOx-Gd neufs présentent une erreur à la référence plus importante (assemblages entourés d'un carré noir sur la Figure 5.19). Cette erreur, déjà présente lors du calcul d'assemblage où IDT sous estimait la réactivité de plus de 250 pcm se retrouve encore pour les calculs de cœur en P1 et P3. Nous n'avons pas cherché à identifier l'origine de cette erreur.

La recherche des points chauds à l'échelle du crayon combustible est un paramètre d'intérêt pour la sûreté. Pour les calculs P1 et P3, l'écart au point chaud est

respectivement de -0.33% et 0.04%. La Figure 5.20 présente la position des crayons ayant les taux de fission maximal à ±0.1% et ±2% pour les calculs Maquette et TRIPOLI-4®.

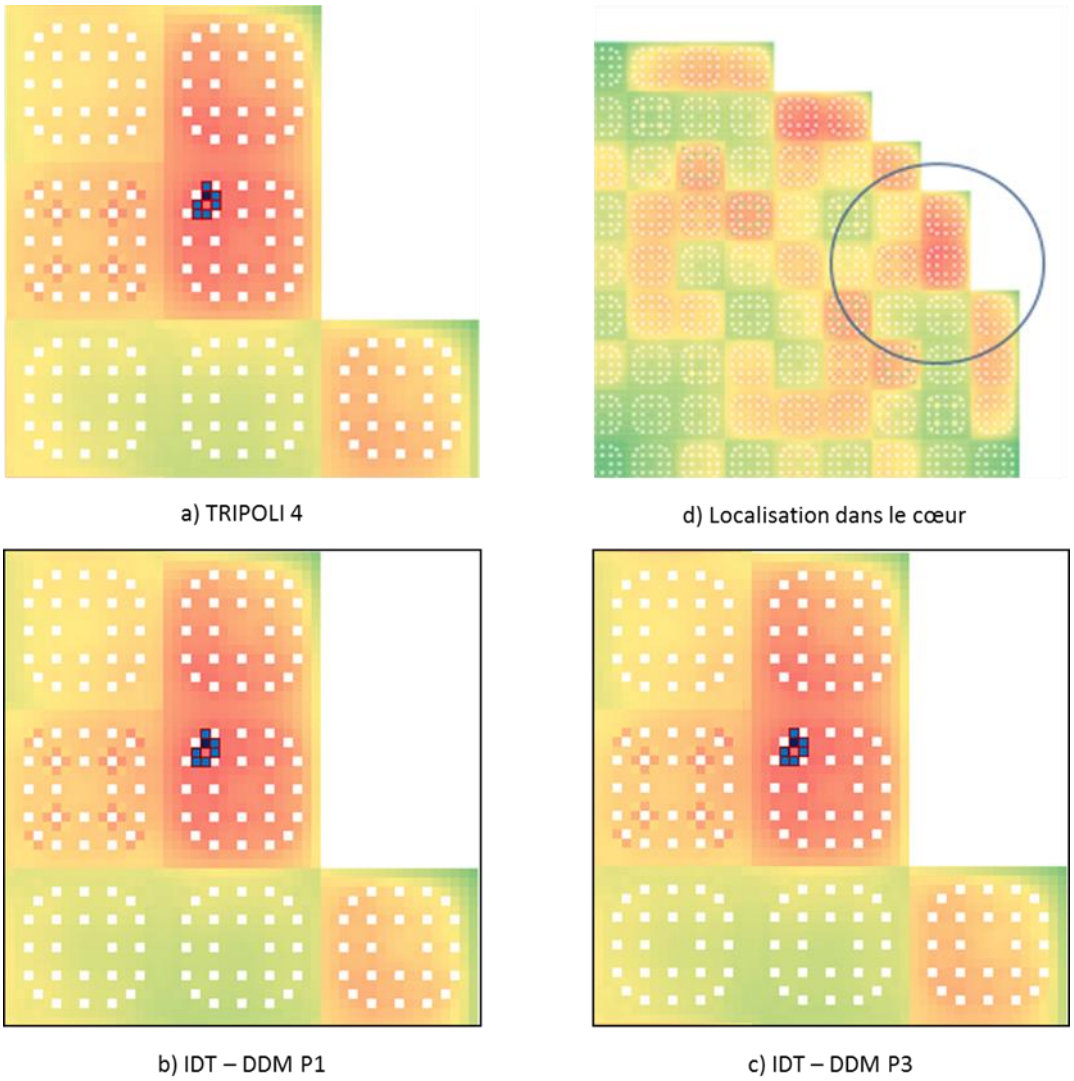


Figure 5.20. Position des taux de fission maximum ±0.1% (bleu foncé) et ±2% (bleu clair) pour les calculs TRIPOLI-4® (a) et DDM en P1 (b) et P3 (c) et position de ces crayons dans le cœur (d).

Outre la valeur du point chaud, ce sont les positions de ceux-ci qui sont très bien évaluées par la Maquette-DDM par rapport à la position calculée par la référence Monte-Carlo.

L'ensemble des résultats présentés dans ce paragraphe est très satisfaisant en termes de précisions par rapport au calcul de référence. Le paragraphe suivant présente les performances de la Maquette vis-à-vis du temps de calcul et de l'occupation mémoire.

5.3.6.c Performances du calcul DDM

Le second volet de cette analyse concerne l'efficacité de la Maquette-DDM. Deux points sont abordés dans cette section : le temps de calcul et l'occupation mémoire. Les calculs Maquette ont été réalisés sur une sur une machine de bureau standard : 6-

proc Intel(R) Xeon(R) E5-2620 2.00GHz, 15.3 MB de mémoire cache par processeur et 64 GB de mémoire RAM.

c.1. Temps de calcul en mémoire partagée

La discrétisation du calcul de cœur nous amène à résoudre un problème à 15 milliards de degrés de liberté. L'accélération par le CMFD est définie sur un maillage grossier comprenant 64 mailles par sous domaine (assemblage) et 26 groupes en énergie. Le nombre de degrés de liberté associé à l'accélération est de 6.10^5. Le calcul a été parallélisé sur 12 threads, en utilisant l'hyperthreading d'Intel. Les temps de calculs sont présentés dans le Tableau 5.10.

Tableau 5.10. Calcul de cœur : temps de calcul et nombre d'itérations.

	DDM P1	DDM P3
Temps de calcul	2h35	5h37
% solveur transport	83	92
% CMFD (coefficients + solveur)	15 (13+1.5)	7 (6.5+0.5)
% échanges flux	<1.3	<1
Nombre d'itérations externes	22	21
Nombre d'internes par sous domaine ($x10^4$)	1.17	1.14
Estimation du temps séquentiel	1.25 jours	2.75 jours

Le temps de calcul total est d'environ 2h35 pour le calcul réalisé avec un ordre d'anisotropie P1 et 5h37 pour le calcul réalisé avec un ordre d'anisotropie P3. Le temps dédié à l'accélération est respectivement, pour les calculs à l'ordre P1 et P3 de 23 minutes (15%) et 28 minutes (7%). La construction des coefficients du CMFD (13 et 6.5 %) est parallélisée sur les sous domaines. En revanche, la résolution qui est séquentielle ne représente que quelques pourcents du temps de calcul, ce que l'on peut considérer comme négligeable. Le temps de communication entre les sous domaines est lui aussi très faible (<1.5% en P1 et P3), ce qui est très encourageant. Le second point positif est le nombre d'itérations externes. En effet, malgré la taille du problème à résoudre notablement plus grande que celle des cas tests étudiées jusque-là, le nombre d'itérations externes n'a pas augmenté considérablement : 22 itérations externes. Au total, en moyenne 40 itérations internes par groupes ont été nécessaires pour atteindre la convergence. Notons que ce calcul est particulièrement exigeant vis-à-vis de la DD : les sous domaines dans les coins sont très éloignés du cœur et le flux qui est calculé dépend uniquement de la solution dans le cœur transmise à travers plusieurs sous domaines. La convergence des flux d'interface de ces sous domaines est alors particulièrement difficile. La stabilité du nombre d'itérations est donc très satisfaisante.

c.2. Occupation mémoire

Le Tableau 5.11 présente les principaux éléments qui occupent de la mémoire lors d'un calcul. Il s'agit du flux, des sections efficaces et des coefficients pour la méthode des caractéristiques courtes (MOSC). Le stockage des sections efficaces et des

coefficients pour le MOSC ne représentent qu'une petite partie de la mémoire dû à la simplification dans la description des matériaux : 292 matériaux pour décrire l'ensemble des assemblages combustibles. Le flux représente plus de 90% de l'occupation mémoire. Un enjeu important de la DDM est l'overhead lié au stockage du flux aux interfaces entre les sous domaines. Dans le cas présent, le stockage des flux surfaciques représente seulement 37% du total dédié au stockage du flux pour le calcul P1 et 18% pour le calcul P3.

Tableau 5.11. Occupation mémoire pour les calculs de cœur en P1 et P3.

	DDM P1	DDM P3
Flux + intégrale de fission (GB)	14	29
Sections efficaces (GB)	0.1	0.2
Coefficients transport (MOSC) (GB)	1.3	2.8
Total (GB)	**15.4**	**32**

L'occupation mémoire totale est adaptée aux capacités de la machine, le calcul n'occupant que 22% de la RAM pour le calcul P1 et 45% pour le calcul P3.

Un calcul 2D en début de cycle a pu être réalisé sur une machine de bureau. Malgré cela l'objectif est de réaliser un calcul similaire en évolution. La conséquence est une augmentation d'un facteur ~100 du nombre de matériaux nécessaire pour décrire l'état d'irradiation. Les besoins en mémoire sont alors bien plus importants.

5.3.7 Les besoins pour un calcul d'évolution

Dans cette section, nous allons tout d'abord évaluer les ressources nécessaires pour un calcul d'évolution de cœur en 2D, puis nous démontrerons la capacité du solveur à réaliser ce type de calcul.

Lorsque l'on souhaite réaliser un calcul d'évolution, le matériau de chaque région est susceptible d'évoluer indépendamment des autres et il appartient au modélisateur de définir les régions pour lesquelles on souhaite suivre l'évolution isotopique. Les regroupements utilisés dans la section précédente pour le calcul DDC ne sont plus possible.

5.3.7.a Les besoins en mémoire

Le Tableau 5.12 fait le bilan des matériaux utilisés pour modéliser l'évolution d'un cœur. Nous avons supposé que chaque région du cœur peut évoluer différemment des autres. Cette hypothèse implique d'attribuer un matériau différent dans chaque région combustible, mais aussi dans la gaine et le modérateur. Pour ces derniers, une telle différentiation n'est pas forcément nécessaire mais nous l'avons réalisée afin de proposer une modélisation avec des contraintes mémoires maximales. Le nombre total de matériaux dans le cœur passe de 292 pour la modélisation en DDC à plus de 4.10^5 pour le calcul d'évolution.

Tableau 5.12. Bilan des matériaux modélisés dans le cœur.

	Calcul statique DDC		**Prérequis pour un calcul en évolution**	
Type d'assemblage	UOx	UOx-Gd	UOx	UOx-Gd
Nombre d'assemblages différents dans le cœur	5	4	181	60
Nombre de matériaux / assemblage	26	41	1638	1728
Nombre total de matériaux dans le cœur	**292**		**400,156**	

Les régions composant le réflecteur n'ont pas été considérées comme évoluant, ainsi nous ne définissons qu'un seul matériau pour décrire l'ensemble du réflecteur.

Le calcul de cœur en évolution a été simulé en créant un matériau et son jeu de sections efficaces pour chaque région des assemblages combustibles. Ces matériaux ont été générés par copie de ceux utilisés dans la modélisation du calcul DDC. Les sections efficaces étant les mêmes, le résultat du calcul est identique, cependant l'occupation mémoire est nettement différente comme le montre le Tableau 5.13.

Tableau 5.13. Occupation mémoire pour un calcul de cœur en évolution.

	Maquette P1	Maquette P3
Flux + intégrale de fission (GB)	14	29
Sections efficaces (GB)	155	302
coefficients transport (MOSC) (GB)	920	2016
Total (GB)	1089	2347

La répartition de la mémoire est différente du cas précédent. La taille mémoire du flux n'est pas modifiée, en revanche la quantité de mémoire requise pour le stockage des sections efficaces et des coefficients du MOSC est de plus de 1 TB en P1 et 2.3 TB en P3. Réaliser un tel calcul sur une machine standard n'est plus possible. De ce fait, l'utilisation de la parallélisation en mémoire distribuée s'avère nécessaire. Pour cela nous utilisons le solveur de la Maquette-DDM avec son mode hybride MPI/OpenMP sur un serveur de calcul.

5.3.7.b Réalisation du calcul en mémoire distribuée

La contrainte de ce calcul étant la quantité de mémoire requise, l'objectif est alors de distribuer au mieux les sous domaines des assemblages combustibles sur les nœuds de calcul.

Le calcul P1 a été réalisé sur la partition Eris du serveur de calcul disponible dans le laboratoire (cf. Annexe C). Il s'agit d'un cluster comportant 36 nœuds biprocesseurs, chaque processeur ayant 4 cœurs. Les processeurs sont des Intel® Xeon X5667 3.06GHz, 12 MB de mémoire cache et 6 GB de mémoire par cœur, soit 1,728 GB de

mémoire pour la totalité de la partition. La quantité de mémoire requise pour le calcul en P1 étant d'environ 1TB, l'espace nécessaire est disponible sur cette partition. Nous utilisons donc cette machine avec 36 processus MPI correspondant aux 36 nœuds, à l'intérieur desquels les directives OpenMP parallélisent le calcul sur 8 threads. Le nombre total de cœurs utilisés est de 288. La distribution des sous domaines sur les processus MPI est présenté Figure 5.21. Ici, les cases correspondent aux sous domaines. Celles avec le même numéro correspondent aux sous domaines d'un même processus.

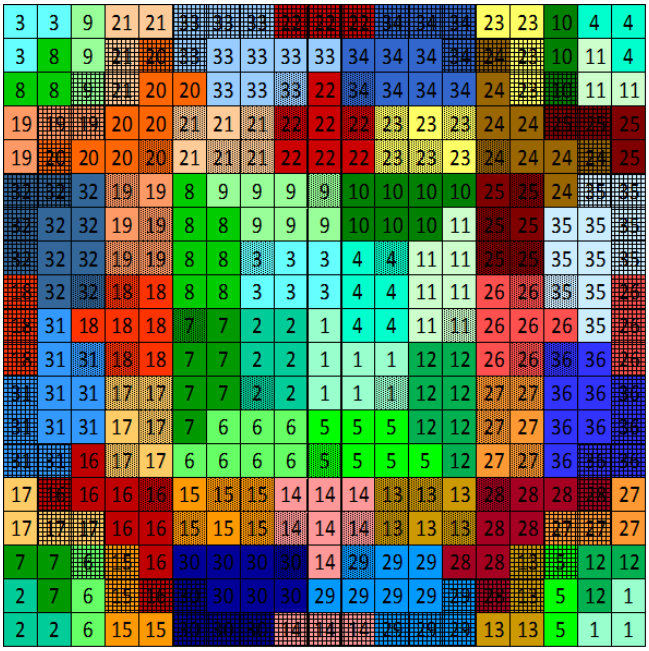


Figure 5.21. Distribution des sous domaines sur 36 processus MPI.

Avec cette répartition, chaque nœud est responsable de 6 ou 7 assemblages combustibles. L'occupation mémoire est de 26.4 GB pour le nœud le moins chargé et 31.8 GB pour le plus chargé. En plus de cela, les nœuds sont responsables de 2 ou 4 sous domaines du réflecteur soit un nombre total de sous domaines par nœud compris entre 9 et 11. La charge de travail n'est certainement pas idéalement répartie. Cependant le temps de calcul d'un sous domaine réflecteur est nettement inférieur à celui d'un assemblage combustible. Les calculs les plus coûteux, qui sont ceux des assemblages combustibles, sont réalisés en premier. Ensuite, le ou les threads disponibles calculent un ou plusieurs sous domaines réflecteurs. Malgré le déséquilibrage de charge dû à la répartition des sous domaines sur les processus MPI, les threads optimisent ainsi la répartition du travail. On peut aussi noter que la topographie des processus MPI ne définit pas des zones contigües. Par exemple le processus noté 1 est présent au milieu du cœur et en périphérie, en bas à droite. Au milieu du cœur il est en contact avec les processus 2, 3, 4, 5 et 12. En bas il l'est avec les processus 5 et 12. Cette partition permet d'associer des assemblages au centre du cœur avec des sous domaines réflecteur sans ajouter de nouveaux voisins à un nœud.

La mémoire nécessaire sur ce nœud n'étant pas suffisante pour le calcul en P3, nous avons utilisé une autre partition du serveur de calcul : la partition Callisto-large.

Celle-ci est composée de 10 nœuds biprocesseurs ayant chacun 10 cœurs. Les processeurs sont des Intel® Xeon E5-2680 V2 à 2.8GHz, 25 MB de mémoire cache et 256 GB de mémoire par nœud soit 2,560 GB de mémoire sur la totalité de la partition. L'espace mémoire est alors suffisant pour réaliser le calcul en P3 (2,350 GB). Le calcul est distribué en 10 processus MPI, chacun parallélisé sur 20 threads, soit au total 200 cœurs pour ce calcul. La distribution des sous domaines sur les processus MPI est présenté Figure 5.22.

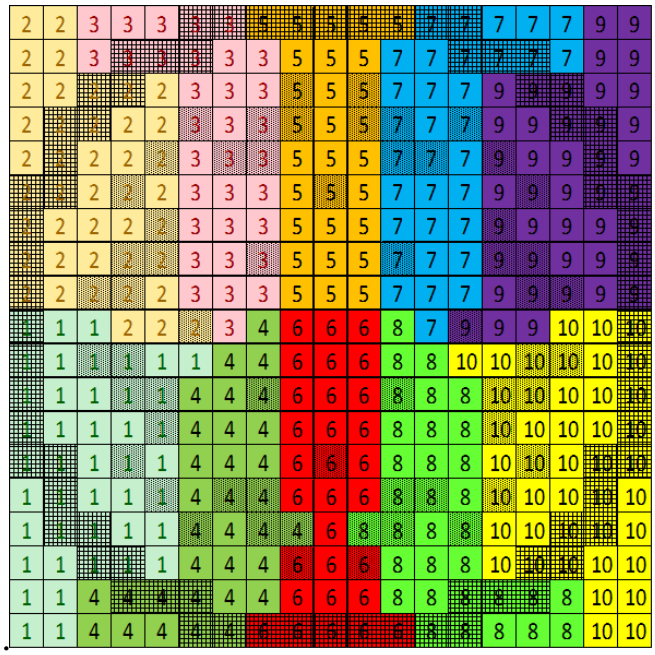


Figure 5.22. Distribution des sous domaines sur 10 processus MPI.

Chaque nœud est en charge de 24 assemblages sauf le nœud 6 qui en reçoit 25. L'occupation mémoire est de 230 GB pour les nœuds les moins chargés et 237 GB pour le nœud 6. Le nombre de sous domaines du réflecteur varie entre 13 et 27 selon le nœud. Le nombre de sous domaines varie fortement entre les nœuds de calcul. Cependant, le nombre d'assemblages combustible (241) est supérieur au nombre de threads (200). Certains threads devront alors calculer 2 assemblages. Le temps de calcul des sous domaines est alors au minimum le temps de calcul de deux assemblages calculés l'un après l'autre. Pendant le calcul du second assemblage, les 15 ou 16 threads restant pourront aisément calculer les sous domaines réflecteurs restants, sans ralentir l'exécution.

Les temps de calculs sont présentés Tableau 5.14. Tout d'abord, notons que des barrières de synchronisation ont été placées avant et après chaque communication de manière à relever précisément les temps de communication et obtenir une estimation enveloppe du déséquilibrage de charge. C'est pourquoi le temps de synchronisation de 9% pour le calcul P1 et 3% pour le calcul P3 est supérieur à celui d'une implémentation optimale. De plus nous avons accordé moins d'importance à l'équilibrage de charge qu'à l'optimisation de la distribution de la mémoire.

Tableau 5.14. Simulation d'un calcul d'évolution : répartition des temps de calcul.

	Maquette DDM P1	Maquette DDM P3
Machine	Eris	Callisto - large
# de threads – # de nœuds	288-36	200-10
Temps de calcul	8.7 minutes	22.2 minutes
% solveur transport	46	50
% CMFD (coefficients + solveur)	26 (5+21)	13 (3+10)
% communications MPI CMFD	16	30
% échanges flux	<1	<1
% communications MPI flux	3	1
% synchronisation	9	4
Temps CPU	19 heures	31 heures

Le temps de calcul en P1 est de 8.7 minutes et celui du calcul P3 est de 22 minutes. Pour un calcul d'évolution, il est possible d'utiliser la solution du pas de temps précédent pour initialiser le calcul de flux pour réduire le nombre d'itération jusqu'à convergence. Le temps de calcul présenté ici est alors une enveloppe du temps de calcul de flux pour un calcul d'évolution.

Le temps de communication MPI du flux est faible : 15 secondes pour le calcul P1 et 13.3 secondes pour le calcul P3. La topographie des processus MPI pour le calcul P1 (Figure 5.21) n'est clairement pas optimisée pour réduire le volume de communications MPI, cependant le temps de communication associé est du même ordre de grandeur que pour le calcul P3 dont la topographie est beaucoup plus régulière (Figure 5.22).

Par ailleurs, le temps de communication pour le CMFD est de 83 secondes pour le calcul P1 et de 400 secondes pour le calcul P3. Ces communications impliquent des opérations de réduction qui sont des opérations collectives. Alors que le calcul P1 est parallélisé sur 36 nœuds, le calcul P3 est parallélisé sur 10 nœuds. Contrairement à ce qu'on pourrait attendre intuitivement, le coût des communications collectives est plus important pour le calcul parallélisé sur 10 nœuds. Cet exemple illustre le fait que qu'une même implémentation n'est pas optimale pour deux machines différentes. Le coût des communications collectives liées au CMFD ainsi que le coût de sa résolution seront réduits une fois celui-ci parallélisé.

Le speedup estimé par rapport au temps CPU est de 45% pour le calcul P1 et 42% pour le calcul P3. Ces speedup sont mauvais mais ne démontrent pas un manque de scalabilité de la méthode. Nous rappelons que, d'une part l'optimisation de la distribution mémoire a été optimisé au détriment de l'équilibrage de charge, et d'autre part que le nombre de sous domaine par nœud n'est pas un multiple du nombre de threads. Cette application a pour but de démontrer la faisabilité d'un calcul requérant

la quantité de mémoire représentative d'un calcul en évolution plutôt que l'optimisation de la parallélisation.

Finalement, nous notons que le temps de l'initialisation du calcul, et en particulier le temps de calcul des coefficients de transport, n'est pas pris en compte dans les temps affichés. La raison est que le calcul des UCG doit être optimisé. Actuellement il est réalisé de manière séquentielle dans chaque nœud et le calcul des coefficients de chaque UCG est parallélisé sur les directions. Dans ces conditions, le temps de calcul pour une UCG est d'environ 2 minutes pour le calcul P3 : le calcul des coefficients est parallélisé avec 10 threads sur les 20 disponibles. Pour le calcul P1, le temps est de 2.8 minutes, parallélisé avec 8 threads alors que 10 tâches sont à paralléliser pour le calcul des coefficients (les directions d'un octant), entrainant un important déséquilibrage de charge. C'est pourquoi cette méthode de parallélisation des UCG n'est pas intéressante dans notre application. Le temps de calcul des UCG restera cependant important et devra être optimisé lors de l'implémentation de l'algorithme de la Maquette DDM dans APOLLO3.

5.3.8 Conclusion

Avec cette application nous avons démontré la qualité de la solution du calcul de cœur en 2D : les écarts sont de l'ordre du pourcent sur le taux de fission à l'échelle du crayon combustible par rapport au calcul de référence Monte Carlo. Ensuite nous avons montré qu'il était possible de réaliser un calcul de cœur en 2D décrit finement pour les variables d'espace et d'énergie avec une machine de bureau standard et en quelques heures. Le nombre d'itérations reste stable malgré l'augmentation de la taille du problème à résoudre et la présence de nombreux sous domaines éloignés du cœur. Ensuite, nous avons réalisé une simulation d'un calcul d'évolution en décrivant un nombre de matériaux très élevé. La quantité de mémoire requise est alors de l'ordre du TB. Ce calcul a été réalisé en mémoire distribuée et converge en moins d'une demi-heure en partant d'un flux initial plat. Cependant, le coût du CMFD ainsi que le déséquilibrage de charge qui ne sont pas optimisés, ne permettent pas de montrer de résultats intéressants en termes de scalabilité. **On retiendra principalement que la Maquette permet de réaliser efficacement des calculs de cœur en 2D décrits finement en termes de composition des matériaux, de discrétisation spatiale et de discrétisation énergétique, le tout dans un temps de calcul raisonnable**.

5.4 Cas d'un colorset d'assemblages 3D avec insertion d'une barre de contrôle

La seconde application proposée est le calcul d'un colorset de 9 assemblages en 3D. La hauteur des assemblages est celle du cas réel (choix d'une hauteur active de 4m) avec un réflecteur modélisant les parties supérieures et inférieures des assemblages combustibles (hauteur réflecteur de 20cm en partie haute et basse). Pour l'exercice, une barre d'absorbant est insérée jusqu'à mi-hauteur de l'assemblage central entrainant une forte déformation axiale du flux. Les assemblages qui composent ce colorset sont de deux types (UOx et UOx-Gd) et correspondent à différents états d'irradiation (différents taux de combustion), de manière à ce que la distribution radiale ne soit pas symétrique.

Aucune simplification n'est adoptée pour la description géométrique des assemblages : chaque cellule est décrite exactement c'est-à-dire avec le combustible, la gaine et le modérateur. Seules les parties correspondant au réflecteur haut et bas ont été simplifiées pour faciliter la modélisation (homogénéisation des structures et du caloporteur dans des proportions représentatives du cas réel).

Les objectifs de cette analyse sont multiples. D'une part il s'agit d'évaluer la capacité de la Maquette à réaliser des calculs 3D finement discrétisés (espace et énergie). D'autre part, le calcul d'une configuration présentant de fortes hétérogénéités de flux (à la fois axialement et radialement) et de forts gradients aux interfaces permet de tester la robustesse de la méthode mise en œuvre.

La première partie de cette section décrit les caractéristiques technologiques du colorset. Ensuite nous présentons les hypothèses de modélisation ainsi que la méthode de génération des sections efficaces macroscopiques. Le calcul de l'assemblage UOx barré en milieu infini est accompagné d'éléments de validation à cette même échelle (comparaison aux résultats de référence Monte-Carlo pour le cas de l'assemblage barré). Enfin, les résultats et les performances du solveur sont présentés.

5.4.1 Description physique et modélisation

La Figure 5.23 représente une coupe radiale du colorset. Ce dernier est constitué de 9 assemblages UOx et UOx-Gd à différents taux de combustion. Les assemblages utilisés pour décrire ce colorset sont issus de l'étude présentée dans la section précédente (calcul de cœur 2D). Ainsi, les dimensions géométriques radiales ainsi que les taux de combustion sont identiques à ceux décrits à la section 5.3.2. Nous considérons ce colorset en milieu infini (condition de réflexion). La numérotation de la Figure 5.23(b) sera utilisée pour repérer les assemblages dans l'édition des résultats.

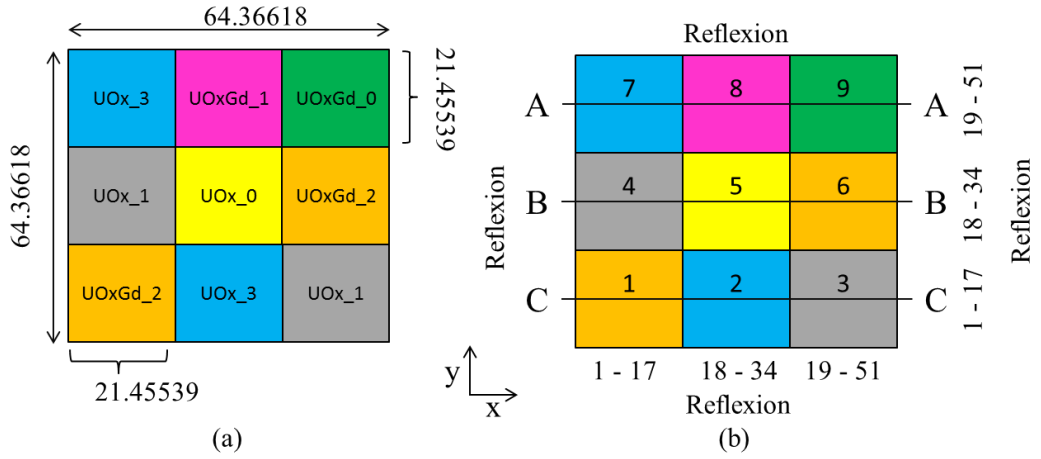


Figure 5.23. Coupe radiale du colorset 3D. (a) Dimensions. (b) Numérotation des assemblages et repères des crayons sur chaque axe (1 - 51).

La Figure 5.24 présente les coupes axiales du colorset. La hauteur active des assemblages combustible est de 4 mètres. Les réflecteurs haut et bas ont une hauteur de 20 cm. Pour des raisons de simplification, ils ont été modélisés par un milieu homogène composé à 50% d'acier et 50% d'eau borée à la même concentration que celle dans les assemblages.

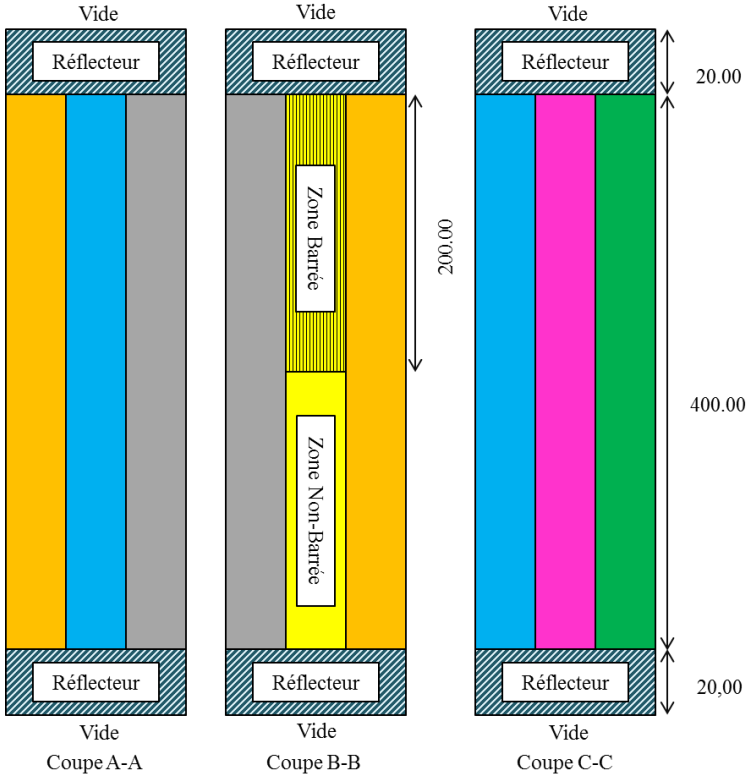


Figure 5.24. Coupe axiale du colorset 3D.

La barre de contrôle, constituée d'un absorbant B4C, est insérée à mi-hauteur dans l'assemblage UOx neuf central (coupe B-B indiquée sur la Figure 5.23(b)). La barre de contrôle n'est pas modélisée dans le réflecteur haut.

Radialement, le colorset est considéré en milieu infini. Des conditions aux limites de vide sont imposées sur les surfaces externes hautes et basses du colorset.

La modélisation des assemblages est exactement la même que celle employée pour le calcul de cœur présenté à la section précédente (cf. §5.3.3). Axialement, la seule discontinuité dans la partie active des assemblages se situe au niveau de l'interface entre la partie barrée et la partie non-barrée. Les tubes guides barrés et non barrés sont représentés sur la Figure 5.25. L'absorbant est dans la partie centrale du tube guide comme présenté sur la Figure 5.25(b). Il est entouré d'une gaine en acier.

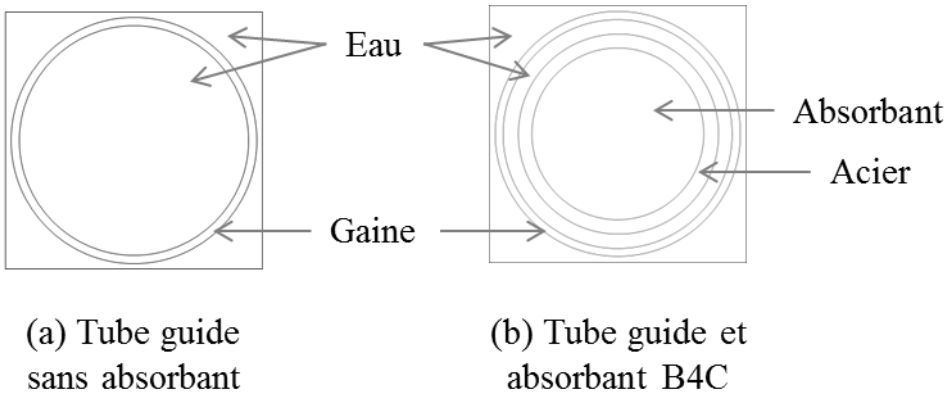


Figure 5.25. Modélisation des tubes guides sans (a) et avec absorbant B4C (b).

5.4.2 Génération des bibliothèques de sections efficaces multigroupes autoprotégées et validation du calcul à l'échelle de l'assemblage

Les sections efficaces utilisées pour ce calcul sont les mêmes que celles utilisées pour le calcul de cœur en 2D, à l'exception de la bibliothèque de sections utilisée pour l'assemblage barré. Les sections efficaces pour ce dernier, qui est un assemblage UOx neuf, ont été obtenues par le même procédé que pour les autres assemblages : calcul d'autoprotection avec APOLLO2 puis édition des sections efficaces à 281 groupes jusqu'à l'ordre P3, sans homogénéisation spatiale. Pour représenter la barre de contrôle, un milieu correspondant à la gaine en acier et un autre représentant l'absorbant sont utilisés pour représenter la géométrie de la Figure 5.25(b).

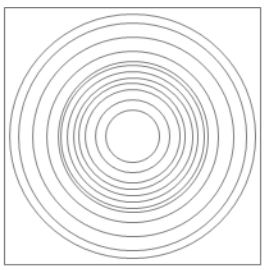


Figure 5.26. Discrétisation du tube guide barré.

Le maillage spatial des absorbants est raffiné suivant les recommandations habituelles comme présenté sur la Figure 5.26.

Les sections efficaces du réflecteur ont été obtenues par un calcul de traverse 1D comprenant une partie combustible de 200 cm modélisée par un assemblage UOx neuf homogénéisé discrétisé en 500 mailles et le réflecteur homogène d'une hauteur de 20.0 cm discrétisé en 50 mailles. Un seul milieu réflecteur a été généré pour tout le réflecteur haut et bas du colorset.

VALIDATION DE L'ETAPE DE PRODUCTION DES SECTIONS MULTIGROUPES A L'ECHELLE ASSEMBLAGE

L'étape de validation à l'échelle assemblage est déjà partiellement réalisée puisque nous utilisons les mêmes bibliothèques que pour le calcul de cœur. Les éléments de validation par rapport à TRIPOLI-4® ne concernent dans cette section que l'assemblage UOx barré.

Le calcul de la référence Monte Carlo avec le code TRIPOLI-4® est obtenu en simulant $7.5.10^7$ neutrons pour l'assemblage. L'incertitude associée au calcul de la distribution de puissance à l'échelle du crayon est inférieure à 1.08% à 3 écarts types avec ces paramètres. Celle sur la valeur propre est inférieure au pcm.

Nous avons utilisé ici les mêmes paramètres pour IDT que ceux utilisés pour le calcul de cœur en 2D soit :

- Quadrature S_8,
- Développement linéaire du flux,
- 3 sous mailles par surfaces et
- Développement des sections de transfert jusqu'à l'ordre P3

Les résultats sont présentés dans le Tableau 5.6 pour l'assemblage UOx barré.

Tableau 5.15. Comparaisons IDT – TRIPOLI-4® (T4) pour les assemblages UOx avec absorbant B4C en milieu infini.

	UOx 0 avec absorbant B4C
Ecart k_∞ assemblage (pcm)	-95
Ecart Max Crayon	1.13%
Nombre de cellules dans l'intervalle à ± 3σ T4	98%
3 σ T4	<1.08%

L'écart sur la valeur propre est plus élevé que dans le cas non-barré mais reste inférieur à 100 pcm. Concernant la distribution radiale des taux de fission, 98% des crayons combustibles présentent des écarts à la référence inférieurs à la valeur 3σ du calcul Monte-Carlo. La distribution des écarts est donné Figure 5.16.

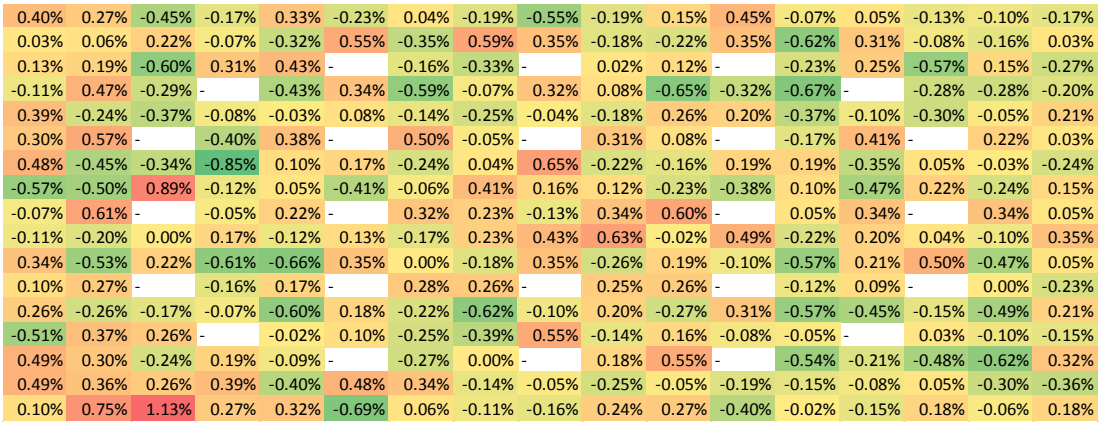


Figure 5.27. Distribution des écarts sur les taux de fission entre TRIPOLI-4® et IDT pour l'assemblage UOx 0 avec absorbant B4C.

Le taux de fission dans les crayons autour des tubes guides contenant l'absorbant est bien calculé malgré la présence d'un fort gradient de flux. On note que la distribution des écarts n'est pas totalement symétrique. L'origine en est la solution produite par le code Monte Carlo qui est convergé mais pas rigoureusement symétrique (le ratio entre le taux de fission dans un crayon ramené à la moyenne est inférieur dans tous les cas à l'incertitude statistique cumulée à 3σ).

Ce résultat obtenu à l'échelle de l'assemblage nous permet de confirmer la cohérence des voies de modélisation entre le calcul de référence dans le code de Monte-Carlo TRIPOLI-4® et le calcul déterministe avec le solveur IDT. Les sections efficaces multigroupes autoprotégées à l'échelle de l'assemblage nous permettent d'obtenir des résultats satisfaisants par rapport au calcul de référence et la discrétisations spatiale du solveur IDT permet d'avoir des écarts sur la distribution de puissance inférieure au pourcent.

5.4.3 Discrétisation retenue pour le solveur DDM

Les paramètres de calcul pour le solveur de la Maquette-DDM déterminés dans le paragraphe 5.4.2 pour le calcul d'assemblage sont utilisés pour le calcul du colorset :

- Quadrature angulaire S_8,
- Développement linéaire du flux
- Développement des sections de transfert jusqu'à l'ordre P3.

La discrétisation radiale est présenté sur la Figure 5.28. Elle est la même que pour les calculs d'assemblages en milieu infini.

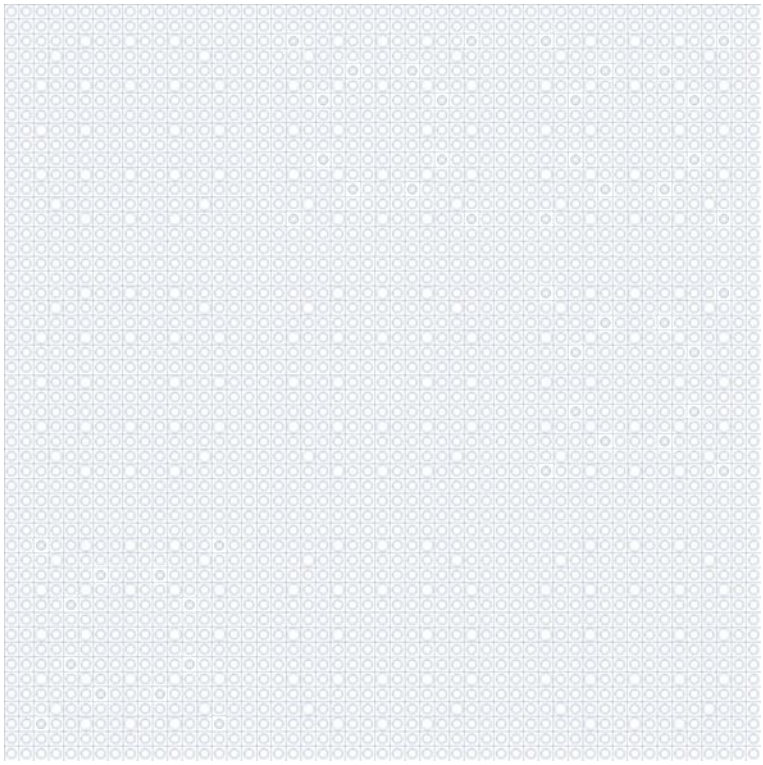


Figure 5.28. Maillage de calcul Maquette-DDM, coupe radiale du colorset.

Axialement, la hauteur active des assemblages combustibles est discrétisée en mailles de 4 cm afin de calculer précisément la distribution de puissance fine. La Figure 5.29 rappelle le maillage des surfaces de chaque cellule utilisé pour la discrétisation spatiale avec le solveur IDT (cf. Figure 1.2 pour plus de détails).

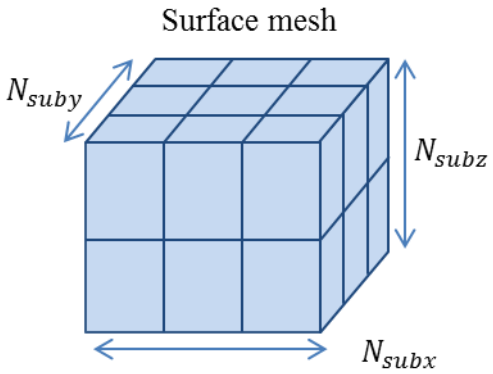


Figure 5.29. Maillage des surfaces d'une cellule en 3D.

Nous avons ainsi 6 mailles pour les surfaces verticales et 9 mailles pour les surfaces horizontales comme représenté sur la figure. Le nombre total de régions est de 1,536,600. Bien que la géométrie soit seulement de 9 assemblages, le nombre de régions est plus de 3 fois supérieure à celui de la configuration de cœur en 2D.

Le domaine de calcul est décomposé en 198 sous domaines, soit 20 sous domaines par assemblages plus les réflecteurs hauts et bas. Ainsi 180 sous domaines permettent de décrire la partie active des assemblages combustibles et les 18 restant permettent de décrire les réflecteurs hauts et bas.

Le CMFD est basé sur un maillage énergétique à 26 groupes, 64 mailles radiales et 5 mailles axiales par sous domaine. Le calcul est réalisé avec un développement des sections de transfert jusqu'à l'ordre P3.

5.4.4 Résultats du calcul du colorset avec la Maquette DDM

La Figure 5.30 présente la distribution axiale du taux de fission intégré radialement par assemblage.

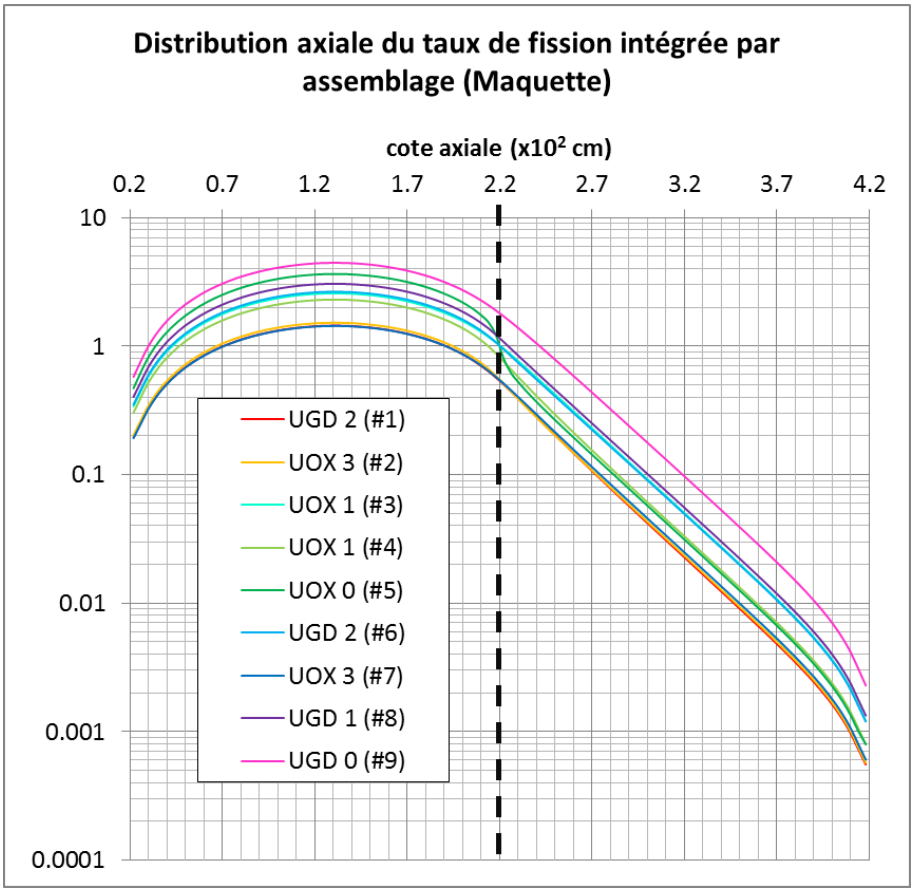


Figure 5.30. Distribution axiale du taux de fission intégré par assemblage (Maquette).

La barre est insérée dans le colorset jusqu'à la cote de 220 cm soit le centre de la zone active (indiquée par les pointillés sur la figure). Le pic de taux de fission est situé à la cote 130 cm et le maximum est positionné dans l'assemblage UOx-Gd 0 (#9). Le taux de fission décroit très rapidement dans la partie supérieure barrée. En effet, entre le plan où le taux est maximal et le haut du colorset, la variation du taux de fission est de plus de 3 ordres de grandeur. Le gradient de flux dans ce colorset est très important. L'axial offset est de -85% ($AO = (\tau_h - \tau_b)/(\tau_h + \tau_b)$). Notons que cette configuration n'est certes pas représentative d'une configuration d'étude habituelle mais elle permet néanmoins d'évaluer les performances de la Maquette sur une configuration complexe d'un point de vue physique.

La distribution radiale du taux de fission est présentée sur la Figure 5.31. Le taux de fission est intégré axialement, crayon par crayon. Le ratio entre la valeur maximale,

dans l'assemblage #9, et la valeur minimale, dans l'assemblage UOx 3 (#7), est de l'ordre de 5.

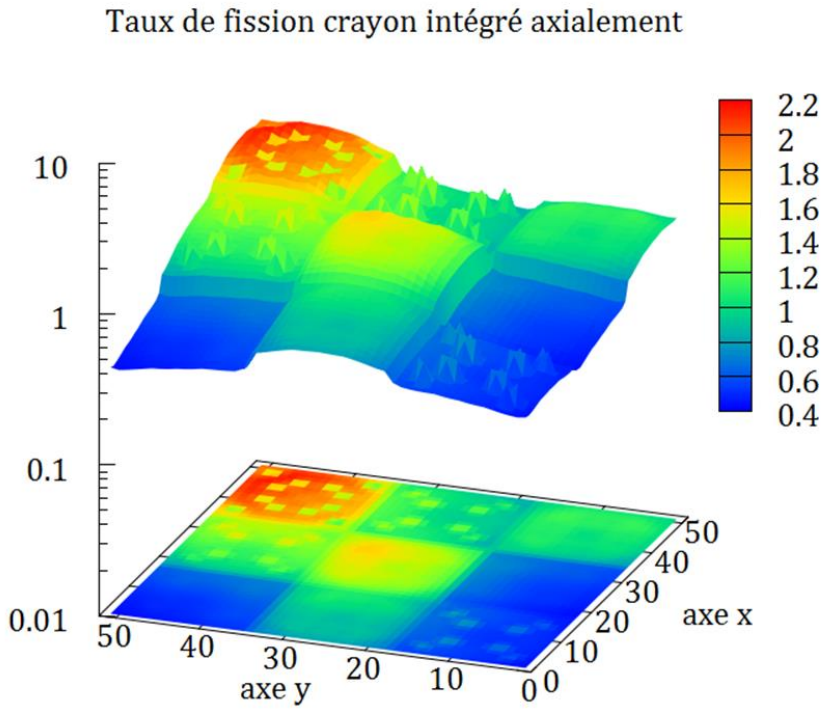


Figure 5.31. Taux de fission intégré axialement.

La barre de contrôle est insérée à mi-hauteur dans l'assemblage central. Sur la Figure 5.31, le taux de fission dans un crayon combustible est intégré axialement ce qui ne permet pas de visualiser la déformation de la nappe selon la position axiale dans le motif. Nous présentons sur la Figure 5.32 la distribution radiale des taux de fission dans plusieurs plans :

- Sur la Figure 5.32(a), il s'agit du plan où le taux de fission est maximal (cote=130cm). Le taux de fission dans l'assemblage central est proche de celui de l'assemblage #9 où est le taux est maximum.
- Sur la Figure 5.32(b), il s'agit de la distribution radiale dans le dernier plan de la partie active en partie basse (cas sans barre). Le taux de fission est atténué dans l'assemblage central (proximité de la barre de contrôle).
- Sur la Figure 5.32(c), il s'agit de la distribution radiale dans le premier plan de la partie active dans lequel est insérée la barre de contrôle (cote=222cm). Les taux varient peu, à l'exception de l'assemblage central dans lequel la barre est insérée : il est nettement diminué.
- Enfin, la Figure 5.32 (d) présente la distribution au plan où le taux est minimal, c'est-à-dire tout en haut du colorset. Le taux de fission décroit de 3 ordres de grandeur entre la cote d'insertion de la barre et le haut du colorset.

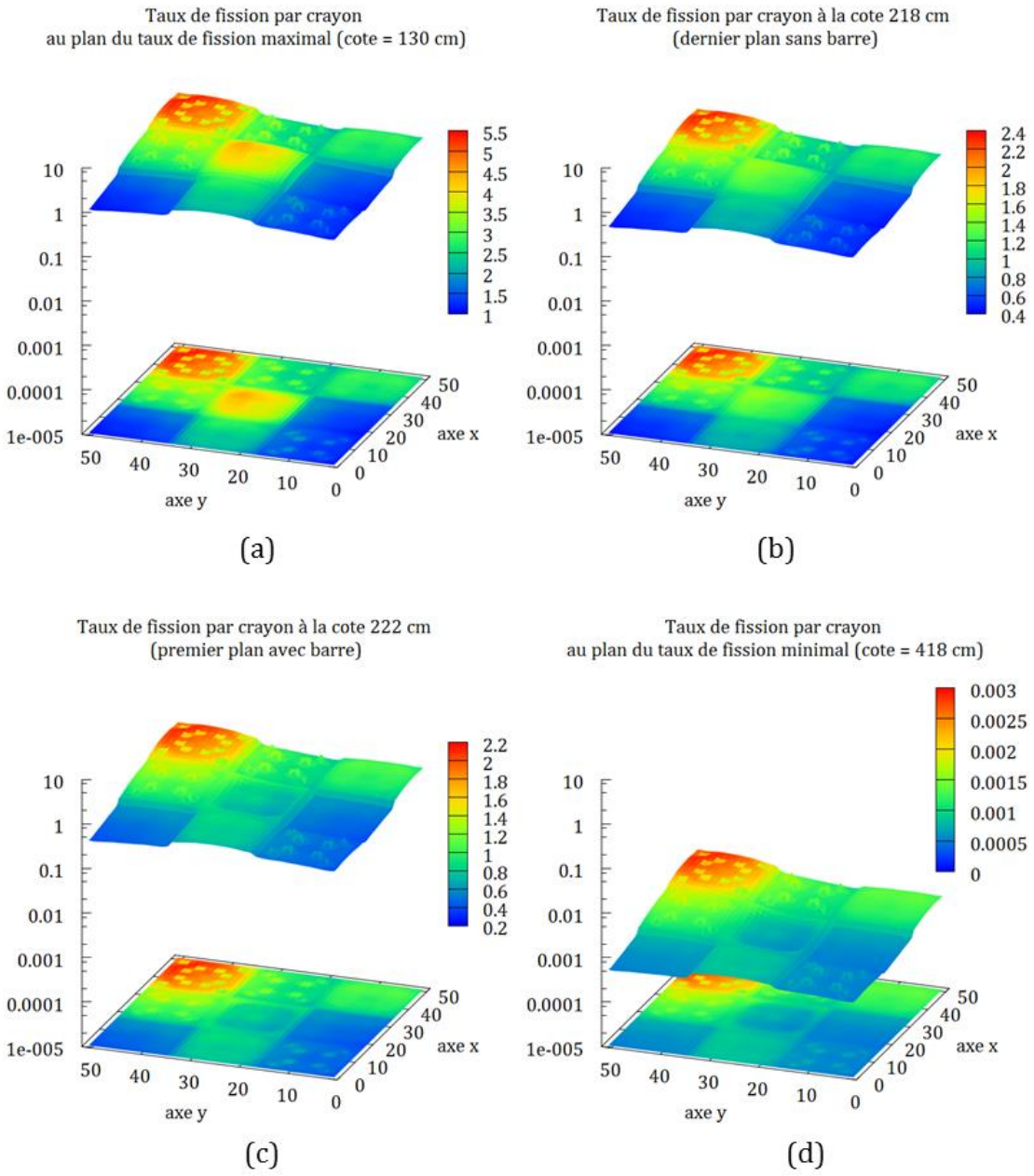


Figure 5.32. Distribution du taux de fission par crayon : (a) au plan de puissance maximal, (b) au dernier plan sans barre, (c) au premier plan avec barre, (d) au plan de puissance minimale.

L'interface entre la partie haute, barrée, et la partie basse est à la cote 220. La Figure 5.33 présente le gradient du flux à cette interface. Dans les assemblages périphériques, le gradient est de l'ordre de -9% entre la cote 218 cm et la cote 222 cm. En revanche, dans l'assemblage barré, le gradient est de -45% environ sur les 4 cm. La barre de contrôle entraine un fort gradient de flux localement, dans l'assemblage barré.

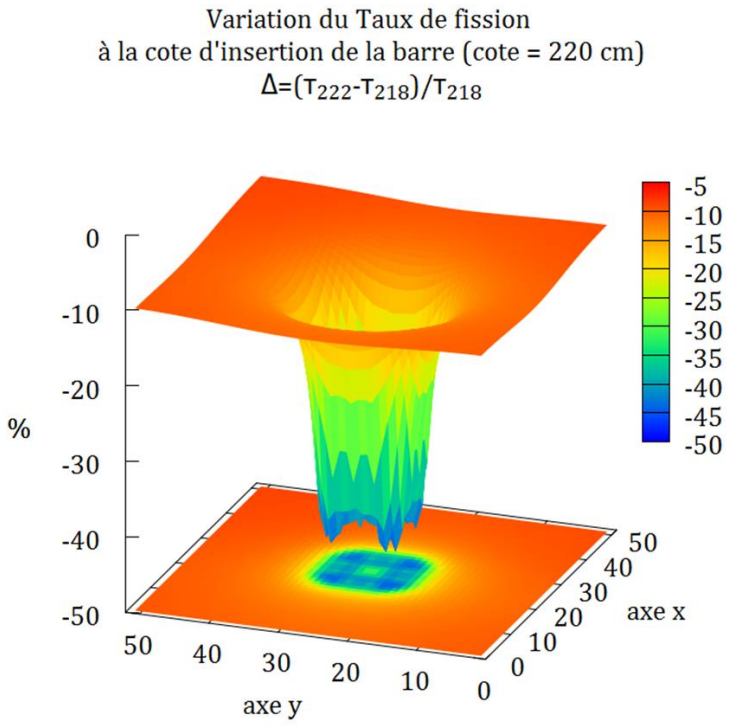


Figure 5.33. Variation du taux de fission à la cote d'insertion de la barre.

5.4.5 Réalisation du calcul

MACHINE DE CALCUL UTILISEE ET DISTRIBUTION DES SOUS DOMAINES SUR LES PROCESSUS

Le calcul a été réalisé sur la partition Eris du serveur de calcul (cf. Annexe C). Il s'agit d'un cluster comportant 36 nœuds avec 8 cœurs par nœud. Nous utilisons cette machine avec 50 processus MPI et 4 threads par processus : le nombre total de cœurs utilisés est de 200 pour les 198 sous domaines. Ainsi, 2 cœurs ne sont pas utilisés, soit 1% du nombre total de cœurs réservés pour le calcul.

La distribution des sous domaines sur les processus MPI est présentée sur la Figure 5.34. Les cases correspondent aux sous domaines des 9 assemblages et les numéros sont les indicatifs des processus MPI. Dans les assemblages combustibles, les 20 sous domaines sont répartis sur 5 nœuds, soit 45 nœuds pour la partie active. 5 nœuds sont ensuite utilisés pour distribuer les sous domaines de la partie réflecteur. Les processus 47 et 49 ne contiennent que 3 sous domaines, ainsi l'équilibrage de charge n'est pas optimal. Cependant nous cherchons ici à démontrer la faisabilité d'un tel calcul plutôt que les performances parallèles.

46	46	46	46	47	47	47	50	50
1	6	11	16	21	26	31	36	41
1	6	11	16	21	26	31	36	41
1	6	11	16	21	26	31	36	41
1	6	11	16	21	26	31	36	41
2	7	12	17	22	27	32	37	42
2	7	12	17	22	27	32	37	42
2	7	12	17	22	27	32	37	42
2	7	12	17	22	27	32	37	42
3	8	13	18	23	28	33	38	43
3	8	13	18	23	28	33	38	43
3	8	13	18	23	28	33	38	43
3	8	13	18	23	28	33	38	43
4	9	14	19	24	29	34	39	44
4	9	14	19	24	29	34	39	44
4	9	14	19	24	29	34	39	44
4	9	14	19	24	29	34	39	44
5	10	15	20	25	30	35	40	45
5	10	15	20	25	30	35	40	45
5	10	15	20	25	30	35	40	45
5	10	15	20	25	30	35	40	45
48	48	48	48	49	49	49	50	50
#1	#2	#3	#4	#5	#6	#7	#8	#9

Figure 5.34. Distribution des sous domaines sur 50 processus MPI pour le calcul du colorset 3D.

COUT EN MEMOIRE

Le Tableau 5.16 présente l'occupation mémoire pour les différents types de sous domaines. La principale composante de l'occupation mémoire est dédiée au flux et particulièrement au flux d'interface qui représente jusqu'à 92% de la mémoire allouée dans les sous domaines réflecteur. Cependant, pour les sous domaines combustibles, l'occupation mémoire des coefficients est du même ordre de grandeur que celle du flux, alors que le nombre de matériaux est nettement réduit par rapport à un calcul d'évolution.

Tableau 5.16. Occupation mémoire pour les différents types de sous domaines du colorset 3D.

	UOx	UOx-Gd	Uox-B4C	Réflecteur
Flux + intégrale de fission	2.62 GB	2.65 GB	2.71 GB	2.12 GB
Fraction dédiée au flux d'interface	74%	74%	72%	92%
Nombre de matériaux	26	41	28	1
Sections efficaces	21 MB	31 MB	21 MB	0.7 MB
Coefficients transport	2.20 GB	3.24 GB	2.71 GB	113 MB

L'occupation mémoire présentée dans le Tableau 5.16 est celle d'une UCG pour les coefficients et les sections alors qu'elle est celle d'un UCE pour le flux. Ainsi pour chaque processus MPI, une seule UCG et 4 UCE sont allouées par processus dans les

sous domaines combustibles. De ce fait, près de 8 GB de mémoire sont épargnés par processus en évitant les copies multiples des UCG.

Tableau 5.17. Occupation mémoire totale.

	Occupation mémoire (GB)
Flux + intégrale de fission	513
Sections efficaces	1.1
Coefficients transport (MOSC)	124
Total	**638**

L'occupation mémoire totale du calcul est présentée dans le Tableau 5.17. Grâce au partage des UCG entre les sous domaines, l'occupation mémoire due aux coefficients est nettement réduite. Il faut alors 638 GB pour le calcul complet. Nous pouvons aussi estimer les besoins en mémoire pour un calcul Direct : plus de 150 GB sont nécessaires pour le stockage du flux, et 20 GB pour les coefficients transport. Un tel calcul n'est alors pas réalisable sur les machines standard disponibles (cf. Annexe C).

Notons que l'occupation mémoire du CMFD n'est pas présentée puisqu'elle est faible (~0.5 GB par processus) et qu'elle doit être améliorée lors de la parallélisation du CMFD. La condensation à 26 groupes est utile pour réduire son impact sur la mémoire : on aurait 38 GB avec un maillage à 281 groupes.

COUT EN TEMPS DE CALCUL

Le nombre d'itérations externes ainsi que la répartition du temps de calcul du processus qui résout le CMFD (processus 1) est présenté dans le Tableau 5.18. Le calcul converge en 25 itérations externes en un peu moins de 8 heures. Le problème à résoudre est nettement plus important que pour le calcul du C5G7, cependant le nombre d'itérations externes reste stable, ce qui renforce notre conviction quant aux capacités de l'algorithme PMBJ accéléré par le CMFD à traiter des problèmes de grandes tailles.

Les calculs transport représentent 86% du temps de calcul total. Le calcul CMFD représente 6% du temps total. Le temps de communication est de moins de 6% en comptabilisant les communications intra et extra nœud. Pour ce processus, en charge du calcul des sous domaines de l'assemblage combustible #1, le temps de synchronisation est de 6%. Cependant, cette valeur n'est pas représentative du déséquilibrage de charge. Parmi tous les processus, le temps moyen du calcul transport est de l'ordre de 80% du temps total, le temps maximal représente 90% du temps total et le temps minimal 70% du temps total. Le déséquilibrage de charge est assez important mais son optimisation n'est pas recherchée ici. Un travail ultérieur devra être effectué de manière à mieux le gérer.

Tableau 5.18. Colorset 3D : nombre d'itérations et répartition des temps de calcul du processus qui résout le CMFD.

	Maquette DDM P3
Nombre d'itérations externes	25
Machine	Eris
# de threads – # de nœuds	200-50
Temps de calcul	7h 42 minutes
% solveur transport	86
% CMFD	6
% communications MPI CMFD	2
% échanges flux	<0.5
% communications MPI flux	5
% synchronisation	6

5.4.6 Conclusion

Nous avons démontré dans cette section la capacité de la Maquette à réaliser un calcul en 3D raffiné en espace et en énergie. Le temps de calcul, de quelques heures avec 200 cœurs, est pénalisé par un déséquilibrage de charge dû à la répartition des sous domaines sur les processus MPI. Cependant, un tel calcul n'est pas accessible avec une machine de bureau standard compte-tenu de la quantité de mémoire requise.

La Maquette offre ainsi de nouvelles capacités pour la réalisation de calculs transport en 3D avec un maillage raffiné à la fois en espace et en énergie. Elle permettra d'étudier des effets 3D en cœur liés à des hétérogénéités de matières (interface UOx – MOX, présence de barre de contrôle) mais également l'impact de certaines hypothèses de modélisation dans les schémas de calcul actuels (mode fondamental notamment). La validation du calcul du colorset 3 x 3 d'assemblages 3D devra être réalisée par comparaison aux résultats de calcul de la référence Monte-Carlo.

5.5 Calcul d'un grand cœur de REL avec réflecteur lourd en 3D à 26 groupes

Afin de démontrer la scalabilité de la méthode vis-à-vis de la taille du problème à traiter, nous avons réalisé un dernier exercice qui concerne le calcul d'un grand cœur de REL avec baffle lourd en 3D et sans aucune hypothèse de symétrie pour le calcul. Le modèle mis en œuvre dans cette section s'appuie sur la description radiale du cœur en 2D et la description axiale du colorset en 3D. Ainsi, le plan de chargement est le même que pour le calcul de cœur en 2D et les assemblages ont une hauteur active de 400 cm identique à celle utilisée pour le colorset. De même pour les réflecteur : radialement, ce sont ceux du calcul 2D qui ont été utilisés et en partie haute et basse du cœur, ceux du colorset 3D.

Le nombre de régions total pour décrire le cœur complet en 3D est de $45x10^6$. La bibliothèque de sections efficaces utilisée pour les calculs a été produite à partir du calcul des assemblages en milieu infini (mode fondamental) à 281 groupes avec le code APOLLO2 et condensée à 26 groupes. La quadrature angulaire utilisée est S_8, soit 80 directions. Le nombre de sous domaines est de 7942 : radialement un sous domaine est défini par assemblage combustible et axialement chaque sous domaine est composé de 5 plans de 4 cm d'épaisseur. Il y a ainsi 361 sous domaines radialement et 22 sous domaines axialement, en incluant les réflecteurs haut et bas.

Nous avons réalisé ce calcul avec 36 processus MPI sur la partition Eris du serveur de calcul de notre département (Annexe C). Le nombre maximal de cœurs disponibles sur cette partition est de 288 ; chaque cœur est alors en charge d'environ 27 sous domaines. 36 processus sont alors définis pour réaliser le calcul, avec 8 threads par processus. La distribution radiale des sous domaines sur les processus est présentée Figure 5.35, elle est la même dans tous les plans.

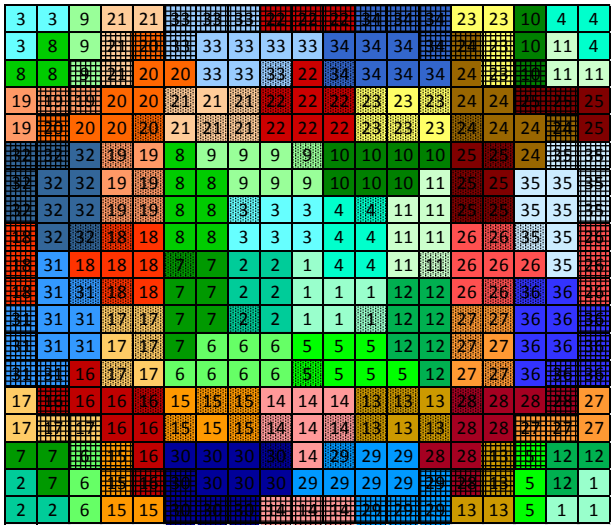


Figure 5.35. Distribution des sous domaines sur les processus MPI.

Chaque processus a en charge de 120 à 140 sous domaines d'assemblages combustible et de 68 à 102 sous domaines de réflecteur à calculer avec 8 threads. Le nombre total de sous domaines dans un processus varie entre 198 et 242 sous domaines. Le déséquilibrage de charge est alors important, d'où la nécessité d'un outil permettant de distribuer automatiquement les sous domaines sur les processus lorsque leur nombre est élevé.

Le temps de calcul est de 27 heures et le nombre d'itérations externes pour converger est de 33. Pour ce calcul, près de 47% du temps est dédié au calcul CMFD démontrant là encore la nécessité de paralléliser cette partie de la méthode efficacement.

Conclusion

Cette dernière application démontre qu'un calcul de cœur avec une discrétisation spatiale très fine et un nombre de groupes d'énergie fin pour un calcul de cœur est réalisable avec l'algorithme PMBJ accéléré par le CMFD implémenté dans la Maquette. Elle met aussi particulièrement en évidence les limites de la réalisation :

nécessité d'une gestion automatique de la distribution des sous domaines sur les processus MPI de manière à réduire le déséquilibrage de charge ainsi que la parallélisation du CMFD.

5.6 Conclusion

Tout d'abord, le calcul des configurations 2D et 3D du benchmark C5G7 a été réalisé, permettant de vérifier la qualité des résultats fournis par le solveur de la Maquette. Les écarts avec le calcul de référence sont inférieurs au pourcent pour le calcul 2D et de moins de 2 pourcents pour les calculs 3D pour la distribution spatiale du taux de fission à l'échelle du crayon. En termes de réactivité, les écarts sont de 6 pcm pour le calcul 2D et de l'ordre de 25 pcm pour les configurations 3D. Ces résultats offrent un bon compromis entre précision et coût de calcul. En effet, nous avons obtenu cette précision avec une quadrature angulaire S_8 et un nombre minimum de mailles spatiales, notamment grâce au développement spatial linéaire du flux, permettant de réaliser ces calculs en quelques secondes pour le 2D et une dizaine de minutes pour les configurations 3D.

La seconde application est le calcul de cœur en 2D avec une discrétisation spatiale et énergétique fine. Elle nous a permis de démontrer la faisabilité d'un calcul de cœur sur une machine standard avec une grande précision. En effet, les écarts sur la distribution du taux de fission à l'échelle du crayon sont de l'ordre du pourcent par rapport au calcul de référence Monte Carlo. Ensuite nous avons démontré la faisabilité d'un calcul d'évolution à l'échelle du cœur en décrivant un nombre de matériaux représentatif d'une telle simulation. Ce calcul est réalisé en distribuant la mémoire sur plusieurs nœuds de calcul. Ainsi en utilisant 200 cœurs, le calcul est réalisé en une vingtaine de minutes.

Ensuite, nous avons réalisé le calcul d'un colorset d'assemblages en 3D, lui aussi raffiné en espace et en énergie. Le temps de calcul est de quelques heures avec 200 cœurs. Cependant ces résultats souffrent d'un déséquilibrage de charge important, que nous n'avons pas cherché à améliorer. Un tel calcul n'est pas réalisable sur une machine standard avec le solveur IDT. La Maquette offre alors de nouvelles possibilités pour la réalisation de modélisations 3D permettant de prendre en compte les discontinuités axiales présentes dans un réacteur telles que les barres de contrôle.

Finalement, un calcul de cœur en 3D, avec un nombre de groupes d'énergie réduit à 26 et une discrétisation spatiale fine, a été réalisé. Le temps de calcul est d'un peu plus d'une journée, parallélisé sur 288 cœurs. Ce calcul est grandement pénalisé par le calcul séquentiel de l'accélération par le CMFD (~50% du temps), d'autant que le nombre de sous domaines est de l'ordre de 8000 ce qui permet une parallélisation nettement plus élevée que celle réalisée.

Ainsi, nous avons présenté quelques exemples de modélisations réalistes accessibles par la Maquette. Ces applications ont renforcé nos convictions quant à la possibilité de réaliser un calcul de cœur en une étape avec une modélisation fine.

Cependant elles ont aussi montré les limites de la réalisation et en particulier la gestion de l'équilibrage de charge et le besoin de paralléliser le CMFD.

Conclusions et perspectives

BILAN DU TRAVAIL DE THESE

L'objectif de cette thèse est de proposer une démarche de modélisation neutronique des cœurs de réacteurs permettant d'intégrer directement les spécificités physiques locales dans une description détaillée en énergie et en espace (3D). Notre démarche a été orientée vers la parallélisation du calcul transport en optimisant l'utilisation de la mémoire et en limitant l'overhead du parallélisme pour répondre au besoin de portabilité du code APOLLO3.

Nous proposons un algorithme basé sur la Décomposition de Domaine (DD) spatiale qui minimise le nombre de communications entre les sous domaines. Pour cela, la DD est appliquée au problème multigroupe, contrairement aux méthodes de DD classiques qui sont appliquées au niveau du problème monocinétique. L'algorithme de résolution consiste alors à réaliser le balayage multigroupe complet avant de communiquer les flux d'interface. Une seule communication est requise par itération externe et les calculs des sous domaines sont indépendants : le coût de la parallélisation est alors minimisé. Il s'agit de l'algorithme de Jacobi Parallèle par Bloc Multigroupe (PMBJ) présenté dans cette thèse.

L'algorithme PMBJ a cependant plusieurs défauts importants. Tout d'abord l'ordre de convergence est dégradé par rapport à l'algorithme de résolution classique. De plus le rayon spectral dépend du nombre de sous domaines. De ce fait, l'augmentation du nombre d'itérations nécessaires pour converger ne permet pas d'atteindre une scalabilité idéale par rapport à l'algorithme de résolution classique. Néanmoins, nous avons implémenté une méthode d'accélération non linéaire par la diffusion : le CMFD. Le rayon spectral de l'algorithme accéléré est alors proche de celui de l'algorithme classique accéléré par la même méthode, comme nous l'avons vu dans le cas du calcul d'un motif d'assemblages de REP. De plus, il est pratiquement insensible au nombre de sous domaines. Finalement, l'algorithme accéléré est performant pour des sous domaines de la taille du quart d'un assemblage combustible, permettant alors de paralléliser efficacement le calcul de cœur de réacteur.

La méthode mise en œuvre permet d'optimiser la mémoire en partageant les données communes à plusieurs sous domaines. Cet avantage est conservé lors de phase de parallélisation en s'appuyant sur le parallélisme à mémoire partagée. Ce paradigme permet aussi de ne pas avoir à échanger de données entre processeurs et facilite donc une transmission rapide du flux d'interface entre les sous domaines. Pour traiter des problèmes de grande taille, il est nécessaire d'avoir accès à un nombre de cœurs plus élevé et à une quantité de mémoire plus importante. C'est pourquoi nous devons nous tourner vers le parallélisme à mémoire distribuée. L'optimisation mémoire est alors

partiellement conservée en utilisant une méthode hybride à mémoire partagée et distribuée. Cette implémentation a été réalisée dans une maquette, offrant la flexibilité nécessaire pour ce travail de recherche. Le choix de l'implémentation préliminaire dans une maquette a permis d'évaluer les diverses méthodes et de démontrer le potentiel de la méthode retenue dans notre cas.

Dans la seconde phase, nous avons cherché à valider la méthodologie mise en œuvre en retenant plusieurs configurations d'études à 2D et 3D. Tout d'abord, nous avons retenu le benchmark C5G7 (2D et à 3D). Celui-ci a permis d'évaluer la précision et les performances de la maquette sur une configuration de benchmarking disposant d'un très grand nombre de résultats dans la littérature. Dans une seconde étape, nous avons évalué les performances de la maquette sur des configurations de cœurs de REL représentatives de cas d'études et présentant une complexité de modélisation (discrétisation en énergie et en espace) supérieure à celle retenue dans le benchmark C5G7. Ainsi, le calcul d'un grand cœur de REL avec un baffle lourd en 2D a été réalisé avec une discrétisation spatiale (description hétérogène de la cellule et du crayon combustible) et énergétique fine (281 gr). Les résultats ont été comparés à ceux d'une simulation Monte Carlo de référence obtenue avec le code TRIPOLI-4. La précision des résultats obtenus avec la maquette est de l'ordre du pourcent sur la distribution du taux de fission à l'échelle du crayon combustible ce qui constitue un résultat très satisfaisant. Ensuite, nous avons réalisé le calcul d'un colorset d'assemblages (3x3) en 3D avec une barre de contrôle dans l'assemblage central, là encore avec une discrétisation spatiale et énergétique fine. Cette application démontre la capacité de la maquette mise en œuvre pour des calculs 3D complexes avec une description énergétique et spatiale beaucoup plus fine que les « standards » de calcul actuels. Finalement la dernière application est le calcul d'un grand cœur de REL en 3D avec une discrétisation spatiale fine et un nombre de groupes d'énergie réduit (26 groupes en énergie).

Ce travail de thèse constitue une première étape vers l'objectif d'un calcul de cœur en une seule étape permettant de se passer de l'étape de calcul de l'assemblage en milieu infini et en mode fondamental tout en utilisant des sections autoprotégées sur un maillage multigroupe standard (environ 300 groupes en énergie). Ces travaux de développement ont été intégrés dans une maquette et permettent d'ores-et-déjà de calculer directement des configurations 2D et 3D en transport avec la précision du calcul de l'étape réseau, sans introduction d'approximations et d'hypothèses de modélisation.

PERSPECTIVES

A l'issue de ce travail, nous identifions diverses pistes susceptibles d'améliorer les performances de la maquette mise en œuvre :

- Nous avons vu que le calcul séquentiel du CMFD pénalise extrêmement la scalabilité de l'algorithme. De ce fait, sa parallélisation devient la priorité actuelle pour améliorer les performances. Une possibilité envisagée est d'appliquer une DDM au CMFD comme présenté en Annexe F. Cette solution permettrait de se

passer des communications collectives couteuses en profitant de la DD appliquée au transport pour définir les sous domaines du CMFD. Un travail bibliographique est à réaliser afin de choisir un algorithme de résolution du CMFD efficace avec la décomposition de domaine.

- Il sera également nécessaire d'améliorer l'implémentation du parallélisme. La version actuelle répond au besoin élémentaire de pouvoir accéder au parallélisme à mémoire distribuée. Cependant, l'algorithme de communication des flux d'interface et son implémentation peuvent être améliorés. Un travail d'optimisation est à réaliser en profitant de l'expérience acquise au cours de cette thèse mais aussi de celle de l'équipe d'APOLLO3. Par ailleurs, il pourrait être intéressant d'étudier des algorithmes asynchrones permettant de minimiser les effets de déséquilibrage de charge.
- L'intégration de la maquette dans APOLLO3 doit s'accompagner du développement d'outils permettant de gérer automatiquement la décomposition de domaine. La démarche d'intégration doit aussi être réalisée en prenant en compte les nombreuses opportunités de recherche qu'offre cette maquette.

En l'état de la maquette, celle-ci reproduit la solution du calcul transport sans décomposition de domaine. Néanmoins, il serait possible d'introduire des hypothèses de modélisation permettant de gagner en performance. Tout d'abord, l'utilisation d'un maillage non conforme à l'interface entre sous domaines permettrait de réduire le nombre de mailles en n'imposant la conformité du maillage qu'à l'intérieur du sous domaine. Une motivation pour un tel développement est que la méthode des caractéristiques courtes d'IDT, très peu coûteuse pour des régions cartésiennes homogènes, est particulièrement adaptée pour les calculs transport 3D des petits réacteurs à plaques (réacteurs expérimentaux ou autres). Cependant, le maillage cartésien conforme engendre un nombre important de régions pour le calcul. Les interfaces non conformes permettraient d'optimiser le maillage avec un gain considérable en temps de calcul et en ressources tout en gardant la même précision.

Cette hypothèse de maillage non-conforme pourrait être généralisée à d'autres variables : il serait intéressant de pouvoir profiter de discrétisations spatiales, angulaires et/ou énergétiques différentes d'un sous domaine à l'autre. De cette manière, il serait alors possible de prendre en compte très précisément certaines hétérogénéités locales sans avoir à utiliser le même degré de précision pour les autres parties du cœur. Cette approche a déjà fait l'objet d'un travail de thèse au CEA [64] où différentes méthodes de discrétisations spatiales (MOC, méthodes nodales, MOSC) et angulaires (S_N, P_N) sont définies dans des sous domaines. Des opérateurs de couplages spatiaux et angulaires sont alors définis pour communiquer les flux d'interface entre les sous domaines. Cette méthode peut alors être très utile pour épargner à la fois du temps de calcul et de la mémoire dans un schéma permettant de prendre en compte précisément des effets locaux d'intérêt. L'architecture de la maquette mise en œuvre au cours de cette thèse permet d'ores-et-déjà d'envisager de tels travaux de R&D.

Un autre moyen de réduire le coût d'un calcul transport 3D serait l'exploration de la méthode de fusion 2D/1D [65] [66]. Cette méthode consiste à transformer le problème 3D en une combinaison de problèmes 2D et 1D. La solution des problèmes 1D permet de calculer les fuites axiales des problèmes 2D et inversement. L'approximation de la solution provient de la manière dont sont représentées les fuites couplant les calculs 2D et 1D. L'architecture de la réalisation actuelle permettrait d'expérimenter la fusion avec un effort de développement relativement peu important. Selon les hypothèses faites sur les fuites axiales et radiales, cette méthode permettrait d'ouvrir la voie à de nouveaux schémas de calcul.

Par ailleurs, nous avons expérimenté l'algorithme PMBJ accéléré par le CMFD uniquement dans le cadre du calcul des REL (spectre thermique). Il serait alors pertinent d'étudier le comportement de l'algorithme pour des réacteurs dont les propriétés neutroniques sont différentes (spectre, épaisseur optique des matériaux, etc.) comme par exemple les réacteurs à neutrons rapides. Une étude théorique des propriétés de convergence de l'algorithme PMBJ accéléré par le CMFD permettrait d'identifier les limites de son champ d'application.

Finalement, notre travail a été orienté vers l'utilisation d'un nombre de processeurs restreint pour répondre aux besoins des utilisateurs d'APOLLO3. Il serait cependant intéressant d'aller plus loin et d'utiliser l'algorithme PMBJ dans une application ayant le HPC comme objectif. Un parallélisme hiérarchisé avec une décomposition de domaine angulaire en mémoire partagée serait une solution pour utiliser un plus grand nombre de processeurs.

Bibliographie

[1] H. Golfier et al., «APOLLO3: a common project of CEA, AREVA and EDF for the development of a new deterministic multi-purpose code for core physics analysis,» chez *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009)* , New York, US, 2009.

[2] I. Zmijarevic, «Multidimensional Discrete Ordinates Nodal and Characteristics Methods for APOLLO2 Code,» chez *Proc. Int. Conf. on Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications (M&C'99),* Madrid, Spain, 1999.

[3] E. Masiello, R. Sanchez et I. Zmijarevic, «New Numerical Solution with the Method of Short Characteristics for 2-D Heterogeneous Cartesian Cells in the APOLLO2 Code: Numerical Analysis and Tests,» *Nucl. Sci. Eng.*, vol. 161, pp. 257-278, 2009.

[4] AEN/NEA, «Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation, A 2-D/3-D MOX Fuel Assembly Benchmark,» 2003.

[5] AEN/NEA, «Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation, MOX Fuel Assembly 3-D Extension Case,» 2005.

[6] T.-4. P. Team, «TRIPOLI-4®, CEA, EDF and AREVA Reference Monte Carlo Code,» chez *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013)*, Paris, FRANCE, 2013.

[7] J. Bussac et P. Reuss, Traité de neutronique : phisique et calcul des réacteurs nucléaires : avec application aux réacteurs à eau pressurisée et aux réacteurs à neutrons rapides, Paris: Herman, 1985.

[8] A. Santamarina et al., «JEFF3.1.1 Nuclear Data Library - Validation results from JEFF2.2 to JEFF3.1.1,» *JEFF Report 22, OECD 2009 NEA No 6807,* 2009.

[9] M. Chadwick et al., «ENDF/B-VII.0: Next Generation of Evaluated Nuclear Data Library for Nuclear Science and Technilogy,» *Nuclear Data Sheets,* vol. 107, pp. 2931-3060, 2006.

[10] H. Bateman, «Solution of a System of Differential Equations Occurring in the Theory of Radioactive Transformations,» *Proc. of Cambridge Philos. Soc.*, vol. 15, 1910.

[11] R. Sanchez et A. Chetaine, «A Synthetic Acceleration for a Two-Dimensional Characteristic Method in Unstructured Meshes,» *Nucl. Sci. and Engineering,* vol. 136, pp. 122-139, 2000.

[12] R. Sanchez, «Module d'autoprotection du code APOLLO2,» *Note CEA N-2765,* 1994.

[13] M. Coste-Delclaux, «Modélisation du phénomène d'autoprotection dans le code de transport multigroupe APOLLO2,» *Rapport CEA, R6114,* 2006.

[14] G. Rimpault et M. Grimstone, «Validation of New Subgroup Algorithm for

Resonnance Self-Shielding in Heterogeneous Structures,» chez *Proc. Int. Top. Meet. on Advances in Nuclear Engineering Computation and Radiation Shielding,*, Santa Fe, New Mexico, US, 1989.

[15] R. Sanchez, I. Zmijarevic, M. Coste-Delclaux, M. Masiello, S. Santandrea, E. Martinolli, L. Villate, N. Schwartz et N. Guler, «APOLLO2 year 2010,» *Nucl. Eng. and Technology,* vol. 42, pp. 474-499, 2010.

[16] S. Santandrea et R. Sanchez, «Acceleration techniques for the characteristic method in unstructured meshes,» *Annals of Nuclear Energy,* vol. 29, pp. 323-352, 2002.

[17] C. Calvin et D. Nowak, High Performance Computing in Nuclear Engineering, Chapter in Handbook of Nuclear engineering, Springer, 2010.

[18] V. Eijkout, Introduction to high performance scientific computing, 2015.

[19] «Le calcul par le GPU,» Nvidia, [En ligne]. Available: http://www.nvidia.fr/object/gpu-computing-fr.html.

[20] «Intel Many Integrated Core Architecture,» Intel, [En ligne]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html.

[21] «OpenACC Directives for Accelerators,» [En ligne]. Available: http://www.openacc-standard.org/.

[22] J. L. Hennessy et D. A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann Publishers, 2012.

[23] «The OpenMP® API specification for parallel programming,» [En ligne]. Available: http://openmp.org/wp/.

[24] «Threading Building Blocks (Intel® TBB),» [En ligne]. Available: https://www.threadingbuildingblocks.org/documentation.

[25] «POSIX® 1003.1,» [En ligne]. Available: http://www.opengroup.org/austin/papers/posix_faq.html.

[26] «Message Passing Interface Forum,» [En ligne]. Available: http://www.mpi-forum.org/index.html.

[27] T. Evans, A. S. Stafford, R. N. Slaybaugh et K. T. Clarno, «DENOVO: a New Three-Dimensional Parallel Discrete Ordinates Code in Scale,» *Nuclear Technology,* vol. 171, 2010.

[28] R. S. Baker et K. R. Koch, «An Sn Algorithm for the Massively Parallel CM-200 Computer,» *Nucl. Sci. Eng.*, vol. 128, pp. 312-320, 1998.

[29] G. G. Davidson, T. M. Evans, J. J. Jarrell, S. P. Hamilton et T. M. Pandya, «Massively Parallel, Three-Dimensional Transport Solutions for the k-Eigenvalue Problem,» vol. 177, pp. 111-125, 2014.

[30] W. S. Yang, M. A. Smith, C. H. Lee, A. Wollaber, D. Kaushik et A. S. Mohamed, «Neutronics modeling and simulation of sharp for fast reactir

analysis,» *Nucl. Eng. and technology,* vol. 42, 2010.

[31] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knipley, L. C. McInnes, B. F. Smith et H. Zhang, «PETSc Web page,» 2013. [En ligne]. Available: http://www.mcs.anl.gov/petsc/.

[32] M. A. Smith, A. Marin-Lafleche, W. S. Yang, D. Kaushik et A. Siegel, «Method of Characteristics Development Targeting the High Performance Blue Gene/P Computer at Argonne National Laboratory,» chez *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, RJ, Brazil, 2011.

[33] B. Kochunas et T. J. Downar, «Parallel 3-D Method of Characteristics in MPACT,» chez *M&C 2013*, Sun Valley, Idaho, 2013.

[34] B. Kochunas, «Hybrid Parallel Algorithm for the 3-D Method of Characteristics Solution of the Boltzmann Transport Equation on High Performance Compute Clusters,» Nuclear Engineering and Radiological Sciences, University of Michigan, 2013.

[35] T. Takeda et H. Ikeda, «Final Report on the 3-D Neutron Transport Benchmarks,» 1991.

[36] W. Boyd, K. Smith et B. Forget, «Parallel Performances results for the OPENMOC Methdod of Characteristics Code on Multi-Core Platforms,» chez *Physor 2014 - The Role of Reactor Physics toward a Sustainable Future*, Kyoto, Japan, 2014.

[37] H. Schwarz, «Uber einige Abbildungsaufgaben,» *Ges. Math. Abh.,* vol. 11, pp. 65-83, 1869.

[38] M. Adams et E. Larsen, «Fast iterative methods for discrete-ordinates particle transport calculations,» *Progress in Nuclear Energy*, vol. 40, n° %1I, pp. 3-159, 2002.

[39] K. S. Smith, «Nodal Method Storage Reduction by Nonlinear Iteration,» *Trans. Am. Nucl. Soc.,* vol. 44, pp. 265-266, 1983.

[40] N. Z. Cho, G. S. Lee et C. J. Park, «Partial Current-Based CMFD Acceleration of the 2D/1D Fusion method for 3D Whole-Core Transport Calculations,» *Trans. Am. Nucl. Soc,* vol. 88, p. 594, 2003.

[41] Y. Saad, Iterative Methods for Sparse Linear Systems, New York: Dover Publication.

[42] K. S. Moon, N. Z. Cho, J. M. Noh et S. G. Hong, «Acceleration of the Analytic Function Expansion Nodal Method by Two-Factor Two-Node Nonlinear Iteration,» *Nucl. Sci. Eng.,* vol. 132, p. 194, 1999.

[43] J. M. Aragones et C. Ahnert, «A Linear Discontinuous Finite Difference Formulation for Synthetic Coarse-Mesh Few-Group Diffusion Calculations,» *Nucl. Sci. Eng.,* vol. 94, pp. 309-324, 1986.

[44] Y. A. Chao, «Coarse Mesh Finite Difference Methods and Applications,» chez *Proc. Int. Topl. Mtg. Advances in Reactor Physics and Mathematics and*

Computation into the Next Millennium (PHYSOR2000), Pittsburgh, Pennsylvania, 2000.

[45] R. Sanchez, C. Rabiti et Y. Wang, «Nonlinear Acceleration of a Continuous Finite Element Discretization of the Self-Adjoint Angular Flux Form of the Transport Equation,» *Nuclear Science and Engineering,* vol. 175, n° %13, pp. 213-226, 2013.

[46] N. Z. Cho et C. J. Park, «A Comparison of Coarse Mesh Rebalance (CMR) and Coarse Mesh Finite Difference (CMFD) Acceleration Methods for the Neutron Transport Calculations,» 2003.

[47] A. Yamamoto, «Convergence Property of Response Matrix Method for Various Finite-Difference Formulations Used in the Nonlinear Acceleration Method,» *Nucl. Sci. Eng.,* vol. 149, pp. 259-269, 2005.

[48] E. W. Larsen et B. W. Kelley, «CMFD and Coarse-Mesh DSA,» chez *Proc. PHYSOR2012,* Knoxwille, Tennessee, USA, 2012.

[49] D. Lee, T. Downar et Y. Kim, «Convergence Analysis of the Nonlinear Coarse-Mesh Finite Difference Method for One-Dimensional Fixed-Source Neutron Diffusion Problem,» *Nucl. Sci. Eng.,* vol. 147, p. 127, 2004.

[50] E. Masiello, «Analytical stability analysis of Coarse-Mesh Finite Difference method,» chez *International Conference on the Physics of Reactors "Nuclear Power: A Sustainable Resource,* Interlaken, Switzerland, 2008.

[51] K. P. Keady et E. W. Larsen, «Stability of Sn k-Eigenvalue Iterations Using CMFD Acceleration,» chez *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, Nashville, Tennessee, 2015.

[52] L. Li, K. Smith et B. Forget, «Techniques for Stabilizing Coarse Mesh Finite Difference (CMFD) in Methods Of Characteristics (MOC),» chez *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, Nashville, Tennessee, 2015.

[53] M. Jarret, B. Kelley, B. Kochunas, T. Downar et E. Larsen, «Stabilization Methods for CMFD Acceleration,» chez *Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, Nashville, Tennessee, 2015.

[54] H. Hiruta et D. Anistratov, «Homogenization Method for the Two-Dimensional Low-Order Quasi-Diffusion Equations for Reactor Core Calculations,» *Nucl. Sci. Eng.,* vol. 154, p. 328, 2006.

[55] E. Masiello, *Notes personnelles.*

[56] E. Masiello, B. Martin et J. M. Do, «Domaine Decomposition and CMFD Acceleration Applied to Discrete-Ordinate Methods for the Solution of the Neutron Transport Equations in XYZ Geometries,» chez *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*, Rio de Janeiro, RJ, Brazil, 2011.

[57] B. W. Kelley et E. W. Larsen, «CMFD Acceleration of Spatial Domain-Decomposed Neutron Transport Problems,» chez *Proceedings of PHYSOR 2012 - Advances in Reactor Physics - Linking Research, Industry, and Education*, Knoxville, TN, USA, 2012.

[58] S. G. Stimpson, B. S. Collins, B. M. Kochunas et T. J. Downar, «Boundary Acceleration Techniques for CMFD-Accelerated 2D-MOC,» chez *PHYSOR 2014*, Kyoto, Japan, 2014.

[59] B. M. Kochunas, B. W. Kelley, S. G. Stimpson, E. Larsen et T. J. Downar, «Application of the SDD-CMFD acceleration technique to parallel 3-D method of characteristic transport,» chez *PHYSOR 2014*, Kyoto, Japan, 2014.

[60] R. Lenain, E. Masiello, F. Damian et R. Sanchez, «Domain decomposition method for 2D and 3D transport calculations using hybrid MPI/OPENMP parallelism,» chez *ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, Nashville, TN, US, 2015.

[61] A. Santamarina et N. Hfaiedh, «Determination of the Optimized SHEM Mesh For Neutron Transport Calculations,» chez *M and C 2005: international topical meeting on mathematics and computation, supercomputing, reactor physics and nuclear and biological applications*, Avignon, France, 2005.

[62] C. Guénaut et H. Golfier, «2D standard and tight lattice core calculations using APOLLO2: Validation with TRIPOLI4,» SERMA/LPEC/RT/08-4597/A.

[63] R. Lenain, E. Masiello, F. Damian et R. Sanchez, «Coarse-grained parallelism for full-core transport calculations,» chez *PHYSOR 2014 - The Role of Reactor Physics toward a Sustainable Future*, Kyoto, Japan, 2014.

[64] E. Girardi, «Couplage de méthodes et décomposition de domaine pour la résolution de l'équation du transport des neutrons,» Université d'Evry Val d'Essonne, 2004.

[65] G. S. Lee et N. Z. Cho, «2D/1D fusion method solutions of the three-dimensional transport OECD benchmark problem C5G7 MOX,» *Progress in Nuclear Energy*, vol. 48, n° %15, pp. 410-423, 2006.

[66] S. Yuk et N. Z. Cho, «p-CMFD acceleration and nonoverlapping local/global iterative transport methods with 2-D/1-D fusion kernel,» chez *PHYSOR 2014 – The Role of Reactor Physics Toward a Sustainable Future*, Kyoto, Japan, 2014.

[67] J. Y. Cho, K. S. Kim, H. J. Shim, J. S. Song, C. Lee et H. G. Joo, «Whole Core Transport Calculation Employing Hexagonal Modular Ray Tracing and CMFD Formulation,» *Nucl. Sci. and Technology*, vol. 45, n° %18, pp. 740-751, 2008.

[68] K. R. Koch, R. S. Baker and R. E. Alcouffe, "Solution of the First-Order Form of Three-Dimensional Discrete Ordinates Equations on a Massively Parallel Machine," *Nucl. Sci. Eng.*, vol. 111, no. 46, 1992.

[69] G. Palmiotti, M. Smith and C. Rabiti, "UNIC: Ultimate Neutronic Investigation Code," in *Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA)*, 2007.

Annexes

Annexe A. Colorset d'assemblages 3 x 3 homogène 2D

Ce cas test est un colorset d'assemblages de type Réacteur à Eau Pressurisé (REP), en milieu infini. La Figure - A.1 représente ce colorset modélisé en 2 dimensions. Il est composé de 9 assemblages UOx neufs avec une grappe d'absorbant B4C insérée dans l'assemblage central. La dimension totale de la géométrie est de 64.83 x 64.83 cm. Chaque assemblage est un réseau de 17 x 17 cellules d'un pas de 1.262 cm. Il est entouré d'une lame d'eau de 0.078 cm. Parmi les 289 cellules, 264 sont des crayons combustibles et 25 sont des tubes guides. Les grappes d'absorbant sont insérées dans 24 tubes guides de l'assemblage central. Le combustible, la gaine et le modérateur sont homogénéisés par cellule.

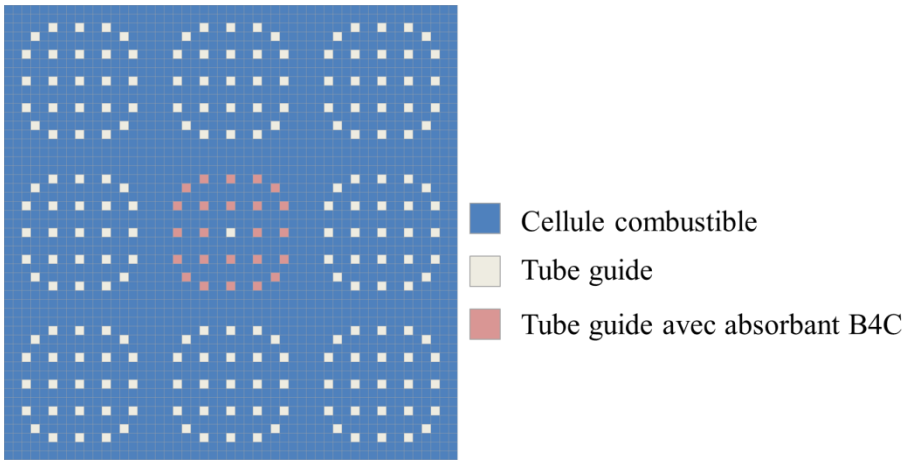


Figure - A.1. Colorset d'assemblages avec absorbant B4C dans l'assemblage central.

Les sections efficaces pour décrire les milieux des cellules homogènes ont été générées en deux étapes par le code APOLLO2. Dans un premier temps les sections efficaces ont été autoprotégées à l'échelle de l'assemblage et discrétisées en 281 groupes d'énergie basé sur le maillage SHEM [61] jusqu'à l'ordre P3 pour le scattering. Dans un second temps, le calcul de flux à l'échelle du colorset a été réalisé à l'aide du code de transport TDT (méthode des caractéristiques) d'APOLLO2 [15]. Les crayons y sont décrits avec 4 couronnes dans le combustible en plus de la gaine et du modérateur. Le flux obtenu par ce calcul permet d'effectuer une homogénéisation flux-volume à l'échelle de la cellule, sans condensation énergétique. Les sections efficaces macroscopiques obtenues de cette manière ont été stockées dans 9 bibliothèques correspondant à chacun des 9 assemblages. Chacune de ces bibliothèques contient un milieu par cellule soit un total de 2601 milieux pour le colorset complet. La taille de la bibliothèque des sections efficaces pour un assemblage est de 110MB, soit un total de 990MB pour le colorset 3 x 3.

Annexe B. Colorset d'assemblages 3 x 3 2D dissymétrique, crayons hétérogène

Ce cas test est un colorset en 2 dimensions de 3 x 3 assemblages, en milieu infini. Il est composé de 9 assemblages UOx qui diffèrent par leurs taux de combustion. Le motif ne présente ainsi aucune symétrie. La dimension totale de la géométrie est de 60.58 x 60.58 cm. Chaque assemblage est composé par un réseau de 16x16 cellules d'un pas de 1.262 cm. Parmi les 256 cellules de l'assemblage, 240 sont des crayons combustibles et 16 sont des tubes guides. Les assemblages pour ce cas test sont issus des assemblages utilisés pour le calcul de cœur, décrits au paragraphe 5.3.3. Si les assemblages de type REP sont formés d'un réseau de 17 x 17 crayons, dans notre cas et pour faciliter l'étude numérique de la décomposition de domaine, le nombre de crayons a été réduit à 16 x 16 en supprimant les rangées centrales de crayons, de manière à garder la symétrie de l'assemblage. Cette simplification a été réalisée de manière à faciliter les études dans lesquelles nous faisons varier le nombre de sous domaines. Malgré le fait que la cohérence entre les sections efficaces et la géométrie ne soit plus scrupuleusement respectée (l'autoprotection a été réalisée sur le motif 17 x 17 et non sur le motif 16 x 16), ce système simplifié permet de réaliser des études numériques, sans accorder d'importance à la précision physique des résultats obtenus.

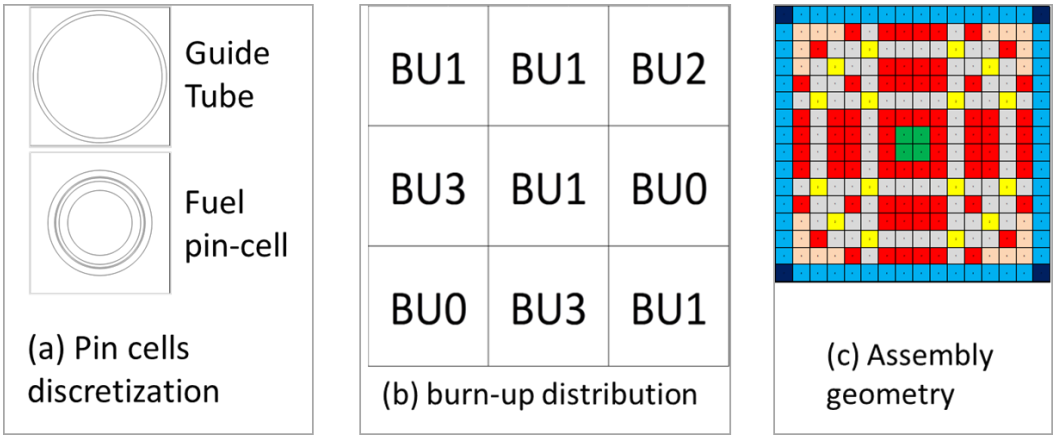


Figure - B.1 Description du colorset hétérogène asymétrique.

Les sections efficaces pour décrire les milieux sont discrétisées en 281 groupes d'énergie jusqu'à l'ordre P1 pour le scattering. Plus de détails sont donnés au paragraphe 5.3.4. Le nombre total de régions est de 13392 pour la géométrie complète du cluster. La géométrie du colorset est donnée Figure - B.1. Les différents taux de combustion associés aux assemblages rendent le colorset asymétrique. Les crayons sont décrits avec 4 couronnes dans le combustible, la gaine et le modérateur. Les différentes couleurs sur la Figure - B.1(c) caractérisent des crayons ayant les mêmes sections efficaces. Les tubes guides sont indiqués en jaune.

Annexe C. Machines de calcul utilisées

Le Tableau - C.1 présente les machines de calcul utilisées pour la production des résultats de ce travail de thèse.

Tableau - C.1. Machines de calcul utilisées.

Nom	CPU	# CPU / nœud	# cœur / CPU	Mem. / nœud (GB)	# nœuds	# cœurs	Mem. cache / cœur (MB)
Machine standard	Intel® Xeon E5-2620 2.00GHz	1	6	64	1	6	15.36
Callisto Slim	Intel® Xeon E5-2680 V2 2.8GHz	2	10	128	40	800	25
Callisto Large	Intel® Xeon E5-2680 V2 2.8 GHz	2	10	256	16	16	25
Callisto Fat	Intel® Xeon E5-4617 2.9GHz	4	6	512	2	48	15.3
Callisto Eris	Intel® Xeon X5667 3.06 GHz	2	4	48	36	288	12

Annexe D. Algorithme de communication des flux d'interface pour une topographie de processus MPI quelconque

La distribution des sous domaines sur les différents processus MPI peut être définie d'une manière quelconque par l'utilisateur. La topographie des processus MPI n'a pas de propriété géométrique particulière. Elle n'est pas nécessairement cartésienne et peut même être discontinue, c'est-à-dire que les sous domaines d'un processus ne sont pas en contact les uns avec les autres. De ce fait il est alors difficile de trouver l'algorithme de communication optimal. Dans cet objectif, une première étape a été réalisée, permettant d'effectuer plusieurs communications simultanément. Cet algorithme est présenté dans l'Algorithme - D.1. Nous définissons la notation Γ_p utilisée dans cet algorithme comme la surface des sous domaines d'un processus p en contact avec des sous domaines d'un autre processus :

$$\Gamma_p = \bigcup_{\alpha \in A_p} \Gamma_\alpha \ : \ \Gamma_\alpha \cap \Gamma_\beta \notin \emptyset, \beta \notin N_p.$$

Pour cet algorithme, deux tableaux auxiliaires sont définis. Le premier, $Status_{comm}(p, p')$, permet d'identifier si les communications entre les processus p et p' ont été effectuées ou non. Le statut est alors respectivement « $TRUE$ » ou « $FALSE$ ». Le second tableau est $Status_{work}(p)$. Il permet de savoir si un processus est en train de communiquer ou non. De même si $Status_{work}(p)$ est « $TRUE$ », le processus est en train de communiquer. Pour alléger les notations, les instructions ont été numérotées par les indices situés à gauche de l'algorithme. Les indices sont indiqués par *(x)* dans la suite.

Au début, aucune communication n'est encore réalisée, *(1)*. Les communications sont effectuées par cycle jusqu'à ce qu'elles soient toutes réalisées, *(2)*. A chaque cycle, on commence par définir qu'aucun processus n'est en train de communiquer, *(3)*. Ensuite on balaye l'ensemble des *P* processus, *(4)*. Si le processus p est déjà en train de communiquer dans ce cycle, il ne peut pas engager de nouvelle communication. On cherche alors un autre processus, *(5)*. Dans le cas contraire, on parcourt la liste des processus devant entrer en communication avec p, *(6)*. Elle est notée $Neighboors(p)$. Dans le cas où le processus p' est déjà en train de communiquer, *(7)*, ou bien que la communication entre p et p' est déjà réalisée, *(8)*, on recherche un nouveau p'dans la liste $Neighboors(p)$. Dans le cas contraire, on indique que les processus p et p' sont en train de communiquer, *(9)*, et que les communications entre ces deux processus vont être effectuées, *(10)*.

Jusque-là, tous les processus ont effectué l'ensemble de ces opérations. Chacun des *P* processus connait alors le statut de tous les processus engagés dans les communications. Maintenant, les communications doivent être réalisées. Dans l'algorithme, PID indique l'identifiant porté par chaque processus. Nous nous mettons maintenant dans la situation où le processus PID exécute l'algorithme. Si PID est p, il est alors engagé dans la communication avec p', *(11)*. De la même manière, si PID est

p', il est engagé dans la communication avec p, *(16)*. Sinon, il n'a pas de tâche à effectuer et il passe au couples (p, p') suivants en poursuivant l'algorithme, jusqu'à être engagé dans une communication. S'il ne trouve pas de communication, par exemple dans la situation où tous ses voisins sont déjà en communication avec d'autres processus, un nouveau cycle est commencé, *(2)*.

Algorithme - D.1. Algorithme de communication des flux entre les processus MPI.

1 $Status_{comm}(p,p') = FALSE \ , \quad \forall p \in P \ , \quad \forall p' \in Neighboors(p)$

2 ***Tant que*** $\exists\, p \in P \ , p' \in Neighboors(p) : Status_{comm}(p,p') == FALSE$, ***faire***

3 $Status_{work}(p) = FALSE \ , \quad \forall p \in P$

4 ***Pour chaque*** $p \in P$, ***faire***

5 ***Si*** $Status_{work}(p) == TRUE$, ***cycler***

6 ***Pour chaque*** $p' \in Neighboors(p)$, ***faire***

7 ***Si*** $Status_{work}(p') == TRUE$, ***cycler***

8 ***Si*** $Status_{comm}(p,p') == TRUE$, ***cycler***

9 $Status_{work}(p) = TRUE \quad , \quad Status_{work}(p') = TRUE$

10 $Status_{comm}(p,p') = TRUE, \quad Status_{comm}(p',p) = TRUE$

11 ***Si*** $p == PID$, ***faire***

12 ***Pour chaque*** $\alpha \in A_p : \Gamma_\alpha \cap \Gamma_{p'} \notin \emptyset$, ***faire***

13 ***Pour chaque*** $\beta \in A_{p'} : \left(x \in \Gamma_\alpha^- \cap \Gamma_\beta^+ \vee (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T\right)$, ***faire***

14 `CALL MPI_SENDRECV` : $\begin{cases} receive\ \psi_\beta^{+(e)}(x) : & \psi_\alpha^{-(e+1)}(x) = \psi_\beta^{+(e)}(x) \\ send\ \psi_\alpha^{+(e)}(x) : & \psi_\beta^{-(e+1)}(x) = \psi_\alpha^{+(e)}(x) \end{cases}$

fin

fin

15 ***Aller à (2)***

fin

16 ***Si*** $p' == PID$, ***faire***

17 ***Pour chaque*** $\alpha \in A_p, : \Gamma_\alpha \cap \Gamma_{p'} \notin \emptyset$, ***faire***

18 ***Pour chaque*** $\beta \in A_{p'} : \left(x \in \Gamma_\alpha^- \cap \Gamma_\beta^+ \vee (T^{-1}x) \in \Gamma_\beta^+ \cap \Gamma_T\right)$, ***faire***

19 `CALL MPI_SENDRECV` : $\begin{cases} send\ \psi_\beta^{+(e)}(x) : & \psi_\alpha^{-(e+1)}(x) = \psi_\beta^{+(e)}(x) \\ receive\ \psi_\alpha^{+(e)}(x) : & \psi_\beta^{-(e+1)}(x) = \psi_\alpha^{+(e)}(x) \end{cases}$

fin

fin

20 ***Aller à (2)***

fin

fin

fin

fin

Les instructions *(14)* et *(19)* sont les communications MPI. Pour les réaliser, il faut communiquer les flux de chaque sous domaine au contact de l'interface $\Gamma_{p'} \cap \Gamma_p$, qui est l'ensemble des interfaces entre les sous domaines du processus p et les sous domaines du processus p'. Chaque sous domaine α du processus p en contact de l'interface $\Gamma_{p'} \cap \Gamma_p$, *(12 ou 17)*, communique avec les sous domaines β du processus p' en contact avec α, *(13 et 18)*. L'échange des flux d'interface est réalisé par un appel à la directive MPI `MPI_SENDRECV`, *(14 et 19)*, qui permet l'envoi et la réception d'un message en une seule instruction. Le processus p envoie le flux du sous domaine α et reçoit celui du sous domaine β, alors que p' envoie le flux du sous domaine β et reçoit celui du sous domaine α. Une fois la communication réalisée, un nouveau cycle recommence, *(15 et 20)*.

Cet algorithme permet de faire communiquer simultanément plusieurs processus sans avoir à se soucier de la manière dont ils sont ordonnés. Par cet algorithme, les situations de blocage (par exemple le processus 1 attend le processus 2 qui attend le processus 3 qui attend le processus 1) ne sont pas possibles. L'enchainement des communications est réalisé de telle sorte qu'un processus communique dès que l'un de ses voisins est disponible.

De plus en utilisant cet algorithme, il est possible d'optimiser l'enchainement des communications. En ordonnant habilement les listes « ***Pour chaque*** $p \in P$, ***faire*** » et « ***Pour chaque*** $p' \in Neighboors(p)$, ***faire*** », il est possible de minimiser le nombre de cycles pour effectuer l'ensemble des communications. Le nombre total de communications étant constant, la minimisation du nombre de cycle entraine la maximisation du nombre de communications simultanées. Dans la réalisation actuelle, aucune stratégie n'a été mise en place pour optimiser l'ordonnancement des listes à l'exception des situations où la topographie des processus MPI est cartésienne et conforme.

Dans l'implémentation présente, la topologie complète est stockée dans chaque processus. Cette structure a une taille proportionnelle au nombre de processus, ce qui peut être problématique lorsque ce nombre augmente. En ordonnant habilement les listes, il est aussi possible de contrôler l'ordre des communications de sorte que les situations de blocage soient évitées. De cette manière, la topologie complète des processus MPI n'a pas besoin d'être stockée dans chaque processus et les 2 listes peuvent alors être réduites à une seule du type : « ***Pour chaque*** $p' \in Neighboors_{PID}$, ***faire*** ».

Un second facteur est important pour optimiser les communications : il s'agit du volume de données par communication. Contrairement aux méthodes de DDM classiques, plusieurs sous domaines sont présents sur chaque processus MPI. De ce fait, le nombre de sous domaines situés sur les frontières Γ_p d'un processus n'est pas le même pour tous les processus. Il en résulte que la quantité de données à communiquer n'est pas également répartie même si l'équilibrage de charge est réalisé au niveau des sous domaines. L'ordonnancement optimal des communications doit aussi prendre en compte cet effet.

L'utilisation des communications désynchronisées `MPI_ISEND` et `MPI_IRECV` n'a pas pu être réalisée. Ces directives permettent d'envoyer un message ou de le recevoir sans avoir besoin de se synchroniser avec l'autre processus de la communication. La raison est que les flux angulaires d'interface sont stockés par groupe et par axe (x,y,z) dans IDT. De ce fait, il est nécessaire de créer un tampon pour regrouper le flux multigroupe dans un seul tableau. Afin de désynchroniser les communications, il faut soit allouer un tableau pour chaque communication, entrainant un surcoût mémoire, soit modifier la manière dont sont stockés les flux d'interface d'IDT.

Nous pouvons aussi noter que les directives OpenMP peuvent être utilisées pour paralléliser l'Algorithme - D.1. Les communications MPI sont alors réalisées par plusieurs threads. Cette amélioration n'a pas été implémentée dans la réalisation.

L'algorithme de communication développé permet de faire face à n'importe quelle topologie de processus MPI, permettant à l'utilisateur d'avoir le choix d'optimiser d'autres paramètres que les communications, par exemple la charge de travail ou la distribution de la mémoire. Cependant, l'algorithme présent n'est pas optimisé et plusieurs voies sont proposées pour l'améliorer.

Annexe E. Résultats complémentaire du C5G7 : configuration non barrée et configuration barrée A

Cette annexe présente les résultats des configurations non-barrée (Tableau - E.1) et barrée A (Tableau - E.2) du Benchmark C5G7 obtenus avec la Maquette DDM.

Tableau - E.1. Comparaison référence - calcul Maquette, configuration non barrée.

	Maquette	Référence
Valeur propre	1.14336	1.14308
Erreur en pcm (écart type associé)	24	2.6

	Partie Basse		Partie Centrale		Partie Haute	
	Maquette	Réf.	Maquette	Réf.	Maquette	Réf.
Ecarts crayons spécifiques et erreurs associées						
Taux maximum	1.108	1.108	0.881	0.882	0.489	0.491
Erreur en % (écart type associé)	-0.009	(±0.090)	-0.102	(±0.100)	-0.280	(±0.130)
Distribution des erreurs en %						
Erreur maximale	0.985	0.290	1.181	0.320	1.562	0.430
Erreur moyenne	0.240	0.164	0.244	0.183	0.328	0.245
RMS	0.300	0.171	0.312	0.190	0.418	0.255
Nombre de crayon dans l'intervalle à :						
1 écart type	37%		43%		43%	
3 écarts types	93%		94%		95%	
Taux de fission moyen par assemblage et erreurs en %						
UOx intérieur	218.74	219.04	174.01	174.24	97.99	97.93
MOX	94.57	94.53	75.27	75.25	43.04	42.92
UOx extérieur	62.16	62.12	49.47	49.45	27.85	27.82
Erreur UOx intérieur (écart type associé)	-0.136	(±0.082)	-0.131	(±0.073)	0.062	(±0.055)
Erreur MOX (écart type associé)	0.043	(±0.061)	0.034	(±0.054)	0.276	(±0.041)
Erreur UOx extérieur (écart type associé)	0.063	(±0.043)	0.045	(±0.038)	0.128	(±0.029)

Tableau - E.2. Comparaison référence - calcul Maquette, configuration barre A.

	Maquette	Référence
Valeur propre	1.12827	1.12806
Erreur en pcm (écart type associé)	19	2.7

	Partie Basse		Partie Centrale		Partie Haute	
	Maquette	Réf.	Maquette	Réf.	Maquette	Réf.
Ecarts crayons spécifiques associées						
Taux maximum	1.196	1.197	0.829	0.832	0.304	0.304
Erreur en % (écart type associé)	-0.114	(±0.080)	-0.288	(±0.100)	-0.048	(±0.200)
Distribution des erreurs en %						
Erreur maximale	1.233	0.210	1.222	0.250	1.211	0.420
Erreur moyenne	0.247	0.157	0.273	0.180	0.355	0.260
RMS	0.313	0.163	0.342	0.186	0.439	0.266
Nombre de crayon dans l'intervalle à :						
1 écart type	36%		38%		42%	
3 écarts types	90%		90%		91%	
Taux de fission moyen par assemblage et erreurs en %						
UOx intérieur	237.32	237.41	167.10	167.51	56.01	56.26
MOX	104.68	104.48	78.07	78.01	39.24	39.23
UOx extérieur	69.90	69.80	53.45	53.39	28.24	28.21
Erreur UOx intérieur (écart type associé)	-0.036	(±0.087)	-0.246	(±0.071)	-0.446	(±0.040)
Erreur MOX (écart type associé)	0.191	(±0.065)	0.081	(±0.056)	0.034	(±0.040)
Erreur UOx extérieur (écart type associé)	0.142	(±0.047)	0.126	(±0.040)	0.104	(±0.029)

Les résultats de la configuration barrée B sont présentés à la section 5.2.3.

Annexe F. Décomposition de domaine du CMFD

La parallélisation du CMFD est nécessaire afin d'obtenir une bonne scalabilité. Pour cela, nous proposons une décomposition de domaine du CMFD, que nous présentons maintenant, avec les notations des sections 3.3.2 et 3.3.3.

Je rappelle la forme discrète de l'équation (3.14) décrivant le système du CMFD :

$$\sum_{S\in\Gamma_R}\frac{A_S}{V_R}\left(a_S^G\,\phi_R^G - b_S^G\phi_{R'(S)}^G\right) + \left(\Sigma_{t,R}^G - \Sigma_{s,R}^{G\to G}\right)\phi_R^G = \sum_{\forall G'\neq G}\Sigma_{s,R}^{G'\to G}\phi_R^{G'} + \frac{1}{\lambda}\sum_{\forall G'}(\chi\nu\Sigma)_{f,R}^{G'\to G}\phi_R^{G'} \quad , \qquad \forall R, G. \tag{F.1}$$

$R'(S)$ est la région séparée de R par l'interface S. On définit alors le problème à valeur propre suivant :

$$\left(K - H_{\setminus D}\right)\phi = \frac{1}{\lambda}F\phi \ . \tag{F.2}$$

avec :

$$\begin{cases} (K\phi)_R^G &= \left(\sum_{S\in\Gamma_R}\frac{A_S}{V_R}a_S^G + \Sigma_{t,R}^G - \Sigma_{s,R}^{G\to G}\right)\phi_R^G - \sum_{S\in\Gamma_R}\frac{A_S}{V_R}b_S^G\phi_{R'(S)}^G\,, \\ \left(H_{\setminus D}\phi\right)_R^G &= \sum_{\forall G'\neq G}\Sigma_{s,R}^{G'\to G}\phi_R^{G'}\,, \\ (F\phi)_R^G &= \sum_{\forall G'\in N_G}(\chi\nu\Sigma)_{f,R}^{G'\to G}\phi_R^{G'}\ . \end{cases} \tag{F.3}$$

C'est sur la matrice *K* des coefficients que portera la décomposition de domaine du CMFD.

Pour cela, nous proposons de diviser le domaine de calcul grossier en sous domaines de même taille que ceux définis pour la DD du transport. Une telle décomposition permet d'éviter les communications collectives coûteuses évoquées dans cette thèse (§4.4, §5.3.7,…).

Ainsi, pour toute région du sous domaine α en contact avec une interface avec les sous domaines β, on a la formulation suivante :

$$(K\phi)_R^G = \left(\sum_{S\in\Gamma_R}\frac{A_S}{V_R}a_S^G + \Sigma_{t,R}^G - \Sigma_{s,R}^{G\to G}\right)\phi_{\alpha,R}^G - \sum_{\substack{S\in\Gamma_R\cap\Gamma_\beta\cap\Gamma_\alpha,\\ \forall\beta:\,\Gamma_\beta\cap\Gamma_\alpha\neq\emptyset}}\frac{A_S}{V_R}b_S^G\phi_{\beta,R'(S)}^G - \sum_{S\in\Gamma_R, S\notin\Gamma_R\cap\Gamma_\alpha}\frac{A_S}{V_R}b_S^G\phi_{\alpha,R'(S)}^G \tag{F.4}$$

Nous séparons ainsi la composante hors diagonale provenant des sous domaines voisins β qui est exprimée dans le second terme. $\phi^{G}_{\beta,R'(S)}$ est le flux dans la région adjacente R' du sous domaine β voisin du sous domaine α, séparés par la surface S.

De cette manière Cette formulation n'introduit pas d'approximation dans le problème à l'interface entre deux sous domaines. Les données communiquées entre les sous domaines sont les flux volumiques des régions adjacentes au sous domaine. Ce problème peut être résolu par un algorithme de Schwarz comme présenté à la section 2.2.3.a (Algorithme 2.1).

Pour établir l'opérateur du CMFD local, les données échangées sont alors les quantités surfaciques et les sections efficaces totales nécessaires pour le calcul des coefficients de diffusion et des coefficients d'équivalence aux interfaces entre les sous domaines. Ensuite les données échangées durant les itérations sont les flux volumiques des régions adjacentes au sous domaine. Ainsi la quantité de données à communiquer entre les processus est nettement réduite, les communications collectives n'étant plus présentes. De plus cette décomposition de domaine permet de profiter du même niveau de parallélisme que le calcul transport.

Nous n'avons cependant pas eu l'opportunité de tester cette décomposition qui est dérivée de manière assez intuitive. La décomposition de domaine entraine une dégradation des propriétés de convergence de l'algorithme de résolution du CMFD mais une multitude de méthodes connues permettent d'accélérer la convergence de ce type de problèmes. Un travail bibliographique et de plus amples recherches sont à réaliser afin d'en évaluer le coût ainsi que les possibles accélérations de cet algorithme de décomposition de domaine du CMFD.