Titre: Title:	Résolution de l'équation du transport des neutrons avec la méthode aux éléments finis isogéométrique
Auteur: Author:	Tengxiang Ren
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ren, T. (2023). Résolution de l'équation du transport des neutrons avec la méthode aux éléments finis isogéométrique [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/53404/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Résolution de l'équation du transport des neutrons avec la méthode aux éléments finis isogéométrique

TENGXIANG REN
Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie énergétique

Mai 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Résolution de l'équation du transport des neutrons avec la méthode aux éléments finis isogéométrique

présenté par **Tengxiang REN**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

André GARON, président
Alain HÉBERT, membre et directeur de recherche
Guy MARLEAU, membre

DÉDICACE

À mon arrière grand-mère et ma grand-mère.

REMERCIEMENTS

Mes remerciements vont tout d'abord à mon directeur de recherche, Alain Hébert, pour sa disponibilité et ses conseils. Il a toujours su me donner rapidement des idées à explorer lorsque je me retrouvais en situation d'impasse.

Je remercie également Michaël et Ivan pour leur compagnies au bureau d'étude. Les échanges et les débats que nous avons eus furent brefs mais intenses.

Je ne compte plus combien de fois je te l'ai dit, mais une fois de plus, merci Atyab ! Tu m'as aidé tout au long de mon passage à l'Institut et je te remercie pour plus que ça. Je pense à notre entretien de physiciens, au lendemain des fêtes, assis chez Premières Moissons ; à la technique de "Grande frite, svp" ; à ta recette de coco-tofu frit !

J'adresse également mes remerciements à ma famille. Merci pour le soutien et l'encouragement au quotidien.

Et pour terminer, mes remerciements vont à ma fiancée. Merci de t'être lancée dans cette aventure avec moi, elle n'aurait jamais été possible sans toi.

RÉSUMÉ

L'objectif de ce mémoire est de réaliser une veille technologique sur la méthode d'analyse isogéométrique (IGA, Iso-Geometric Analysis) appliquée à la résolution de l'équation de transport des neutrons (BTE, Boltzmann Transport Equation). Il s'agit d'une méthode de résolution similaire à la méthode des éléments finis classique FEM (Finite Elements Method). Cependant, la méthode IGA présente deux avantages. D'une part elle retranscrit exactement la géométrie lors de l'analyse numérique et d'autre part elle facilite la transition de la phase de conception à la phase d'analyse.

Pour atteindre cet objectif, un prototype MATLAB est conçu à partir de la littérature. Les différentes étapes de la conception sont détaillées afin de rendre l'algorithme reproductible.

Sur le plan théorique, l'équation de transport est discrétisée selon les méthodes usuelles, à savoir, la discrétisation en énergie avec l'approche multigroupe et la discrétisation en angle avec la méthode S_N (ordonnées discrètes). Quant à la discrétisation spatiale, elle constitue la différence entre la FEM classique et la méthode IGA.

Sur le plan pratique, la génération des géométries NURBS (Non-Uniform Rational B-Spline) ainsi que le calcul des fonctions NURBS correspondantes sont réalisés en utilisant le package GeoPDE. Leur incorporation dans la FEM avec ordonnées discrètes constitue la tâche principale. La logique de conception de l'algorithme demeure similaire à celle de la FEM, impliquant la méthode d'itération de puissance pour calculer le facteur de multiplication effectif K_{eff} et la méthode de source itérée pour découpler les équations et calculer le flux.

Pour mener à bien la veille technologique, une démarche progressive est adoptée.

Le premier sous-objectif est de construire correctement l'algorithme. Pour ce faire, les équations sont explicitées pour chacun des éléments constitutifs de l'algorithme. Le bon fonctionnement du package principal - GeoPDE - est vérifié. Les matrices de résolutions produites par l'algorithme sont également confrontées avec un second calcul indépendant.

Le deuxième sous-objectif est de tester la performance de l'algorithme. Il est soumis à des problèmes de complexités croissantes, testant différents aspects de sa conception. Les problèmes résolus vont d'un problème à source externe monogroupe isotrope (méthode dite de solution manufacturée, MMS) à un problème à fission multigroupe anisotrope. Et les géométries impliquées incluent aussi bien des formes cartésiennes que des géométries en Pincell et Colorset. Enfin, le dernier sous-objectif est de connaître les limites de l'algorithme et de réfléchir à des améliorations futures.

Les résultats obtenus sont mitigés. Si l'algorithme est capable de résoudre la plupart des problèmes qui lui sont posés, ses résultats manquent parfois de précision comparés à d'autres

outils comme DRAGON, qui servent de référence. Par exemple, l'équation multigroupe anisotrope sur une géométrie Pincell est résolue avec précision jusqu'à ce qu'une gaine vienne entourer le crayon de combustible central.
Une faille de conception semble avoir été identifiée, mais elle demeure non résolue à l'heure de la rédaction. Cependant, un cas d'étude simple et reproductible permet de la mettre en avant et de l'étudier par une approche unitaire. Les démarches d'étude entreprises sont clairement documentées.

ABSTRACT

The main goal of this work consists in a technological watch over the application of the Iso-Geometrique Analysis (IGA) method in solving the Boltzmann Transport Equation (BTE) of neutrons. It is a solving method similar to the standard finite elements method (FEM). Meanwhile, IGA presents two advantages. On one hand, it is capable of exactly transcribing the geometry in order to use it in analysis. On the other hand, the transition from the designing stage to the analysis stage would become smoother.

To fulfill the goal, an algorithm is conceived with the information gathered from the litterature. Every step of the conception is detailed in order to make the algorithm reproducible.

On the theoretical ground , the BTE is discretized following the standard procedure, i.e, the energy discretization with multigroup approach, and the angular discretization with the S_N method (Discrete Ordinates). When it comes to the spatial discretization, it is where the IGA method introduces the main changes.

From the practical point of view, the creation of NURBS (Non-Uniform Rational B-Spline) geometry and the calculus of NURBS functions are performed using the GeoPDE package. Their incorporation into the standard FEM (with discrete ordinates) is the main task to deal with. However the solving logic remains the same, requiring the power iteration to compute Keff and the source iteration method to decouple equations and calculate the flux.

A progressive approach is adopted to ensure the accuracy of every step along the path.

The first sub-goal is to correctly build the algorithm. For that purpose, every term involved in the conception of the algorithm has its equation explicited. The main package, GeoPDE is checked. The matrices computed by this package are compared to another version of them, which are independantly computed.

The second sub-goal is to test the perfomances of the algorithm. It is challenged with a series of problems with growing complexity. The problems range from isotrope monogroup source problem (the Method of Manufactured Solution, MMS, is used) to anisotrope multigroup fission problem in order to shed light on differents aspects. Also the geometries varies from cartesian to the standard Pincell geometry. And finally the last sub-goal is to find out the algorithm's limits and propose futur improvements.

The performances observed are mixed. The algorithm is capable of solving most of the problems but some of the results lack precision, when compared to other solving tools like DRAGON. For example, the anisotropic multigroup fission problem over a Pincell is well managed until the fuel pin is surrounded by a protective material. The flaw seems to be identified but it remains unsolved. Meanwhile, a simple and reproductible test-case is found

and allow to diagnotise the flaw. All the procedures are documented.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

NURBS	Non-Uniform Rational B-Splines
IGA	Analyse Iso-Géométrique (Iso-Geometric Analysis)
Keff	Facteur de multiplication effectif
FEM	Méthode des éléments finis (Finite Elements Method)
PDE	Équation aux dérivées partielles (Partial Differential Equation)
BTE	Équation de Transport de Boltzmann (Boltzmann Transport Equation)
CAD	Design Assisté par l'ordinateur (Computer Assisted Design)
CAE	Ingénierie Assisté par l'ordinateur (Computer Assisted Engineering)
CAM	Fabrication Assisté par l'ordinateur (Computer Assisted Manufacturing)
S_N	Discrétisation aux ordonnées discrètes
MMS	Méthode de la solution manufacturée (Method of the Manufactured Solution)

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte et motivation

Les courbes de Bézier ont été inventées dans les années 60s par l'ingénieur qui leur donna son nom. Il était à la quête d'outils mathématiques pour modéliser efficacement des courbures lisses et naturelles. Ces courbes définies à partir de fonctions polynomiales par morceaux et de points de contrôle n'ont été alors uniquement appliquées qu'au service des compagnies automobile. De nos jours, les Non-Uniform Rational B-Splines, ou NURBS - successeurs des courbes de Bézier - jouent désormais un rôle important en CAD/CAM/CAE, et ce dans tous les domaines de l'industrie et d'arts visuels. Leur succès a commencé dans les années 90s grâce à la simplicité de leur définition qui permet toutefois une description précise des géométries complexes (objets coniques, paraboliques...)
Quant à la FEM, il s'agit d'une des méthodes courantes de résolution d'équations aux dérivées partielles (Partial Differential Equation, PDE) qui a vu ses débuts dans les années 40s. Une grande partie des PDEs en Physique demeurent insolubles analytiquement avec les outils mathématiques actuels. La FEM permet d'y apporter une solution approchée en discrétisant le domaine d'étude en un nombre fini de sous régions(l'ensemble des sous région est nommé maillage, chaque sous région est nommée élément). Sur chaque élément une solution locale est calculée grâce à une base de fonctions de décomposition. La solution numérique globale est obtenue en combinant les solutions locales.
L'objectif du mémoire consiste à amener une veille technologique sur l'utilisation des NURBS dans la FEM, en construisant un algorithme fonctionnel. Pour la version classique de la FEM, les éléments ont des formes triangulaires ou bien quadrilatérales et les bases de décompositions courantes sont polynomiales par morceaux. La FEM a fait preuve de son efficacité au cours du temps, néanmoins ce mémoire tente d'investiguer les améliorations que la combinaison avec les NURBS pourrait apporter :

— Dès lors que la géométrie possède, par exemple, un objet conique comme un disque, le maillage de la FEM ne peut être qu'une approximation de celle-ci. En effet, de nombreuses courbures ne peuvent pas être décrites par les formes courantes des éléments en FEM et de ce fait, une erreur est systématiquement introduite. Pour réduire cette erreur, la FEM requiert souvent un maillage fin dans le but d'approximer correctement la géométrie. L'utilisation des géométries NURBS en tant qu'éléments pour la FEM est une réponse à cette contrainte. Comme les géométries NURBS interviennent en amont de l'analyse, pendant la phase de conception de la géométrie, leur réutilisation

permet de décrire parfaitement les courbures. Il n'y a pas d'écart entre la géométrie conçue et la géométrie analysée, ainsi cette erreur est éliminée. De plus le maillage peut être plus grossier comparé à la FEM classique. Ainsi les NURBS permettent d'améliorer la précision de la solution numérique avec un coût de calcul moindre.

— La transition entre la phase de conception et la phase d'analyse devient naturelle avec les NURBS. Le maillage n'a pas besoin d'être calculé séparément, alors que trouver un maillage adéquat est une tâche à part entière pour la FEM classique. Les données de la géométrie conçue peuvent servir directement en tant que le maillage le plus grossier. Cependant quelques règles de conception propre à l'algorithme de ce mémoire sont à observer, sinon les données devront être réarrangées. Néanmoins, le coût calculatoire reste bien moindre.

Dans le cadre du mémoire, l'algorithme est utilisé pour résoudre l'équation de transport de Boltzmann en régime stationnaire (Boltzmann Transport Equation, BTE). L'équation est simplifiée avec les discrétisations usuelles. Le schéma de calcul est similaire à la FEM classique. Les géométries principalement étudiées sont celles du Pincell et du Colorset. Le Pincell représente un crayon de combustible cylindrique entouré par de l'eau. L'étude est en 2D, la géométrie revient alors à un disque de combustible encadré par un carré d'eau. Et le Colorset est un assemblage de Pincell. Ces géométries peuvent être étudiées par DRAGON avec la méthode des caractéristiques [1]. Cela permet la comparaison des résultats, en prenant DRAGON5 comme référence. D'autre part, il existe aussi un algorithme Discontinuous Galerkin Finite Element Method (DGFEM) dans DRAGON5, une autre référence pour des géométries cartésiennes [2]. Le prototype fonctionnel pourrait être adapté pour DRAGON, ainsi il permettrait d'enrichir son inventaire d'outils et apporter plus de flexibilité à l'étude de géométries complexes.

L'application de l'algorithme ne se limite pas au domaine nucléaire. Il pourrait être intéressant dès lors qu'il y a une PDE à résoudre, et donc dans toutes les branches de la physique. Le mémoire est très fourni en détails techniques et en équation, dans l'espoir de permettre au lecteur intéressé de pouvoir adapter la méthode à la résolution de ses propres équations.

1.2 Plan du mémoire

Le mémoire est composé d'une partie théorique suivie d'une partie résultat.

Pour la partie théorique, la théorie des NURBS est présentée dans un premier temps. Quelques unes de leurs propriétés, intéressantes et essentielles à leur utilisations en analyse sont explicitées. Les deux méthodes de raffinement d'une géométrie NURBS (h et p) sont expliquées. Dans un deuxième temps, la BTE et ses discrétisations sont détaillées. Enfin, la conception

de l'algorithme est montrée en deux sous-sections, avec une sous-section pour la construction des matrices et une sous-section pour le schéma de calcul.
En ce qui concerne la partie résultat, le package GeoPDE et le bon assemblage des matrices sont vérifiés sur un patch typique du Pincell, via un calcul indépendant. Ensuite, la performance de l'algorithme à traiter divers problème est présentée. Cela commence par la MMS, sur un problème à source, monogroupe et isotrope. Un problème à fission, monogroupe et anisotrope s'en suit. Puis, un problème à fission multigroupe anisotrope est étudiée sur une géométrie Pincell, où les valeurs de k_{eff} et du flux sont comparés à celles calculées par DRAGON. Enfin, la géométrie est davantage complexifiée avec un Colorset. Et pour terminer, une dernière section traite de la faille présente dans l'algorithme et laisse des indices à un quelconque successeur.

CHAPITRE 2 REVUE DE LITTÉRATURE

Si les NURBS ont vu leur succès commencer dans les années 90s, M.Piegl y a contribué son effort. Les informations sur les NURBS, encore dispersés à l'époque, ont été rassemblées pour la première fois dans son livre "The NURBS Book" [3], publié en 1995 et devenu un incontournable pour les amateurs de NURBS. Les premières utilisations de NURBS en analyse datent de 2005 [4]. Elles étaient alors utilisées pour résoudre des problèmes de mécanique de solide. Leur application en neutronique commence en 2012 [5], et les pionniers sont aujourd'hui seulement constitués de deux groupes de chercheurs, un anglais et l'autre japonais. Leurs travaux montrent que la méthode IGA est efficace pour résoudre les problèmes courants de la neutronique et montre une précision suffisante pour un temps de calcul adéquat. Cependant ces travaux sont récents, et il n'y a que très peu de littérature disponible. Les informations clés ne sont pas dévoilées, et de ce fait les résultats ne sont pas reproductibles. Ce mémoire tente de construire un prototype d'algorithme en s'appuyant sur leurs travaux [6, 7]. Cela permettrait d'enrichir la littérature et apporter une validation à leurs résultats.
Par ailleurs, les détails sur les méthodes de résolution telles que l'itération de puissance ou la méthode de source itérée n'ont pas été présentés dans les travaux anglais et japonais. Ce mémoire met en lumière les détails d'implémentation permettant la reproductibilité des résultats. L'adaptation à la méthode IGA s'inspire alors de la littérature disponible sur les autres méthodes de FEM [2, 5].

CHAPITRE 3 THÉORIES

Les trois concepts fondateurs de ce mémoire sont les NURBS, la BTE et l'IGA. Ce chapitre présente leurs éléments théoriques respectifs, en commençant par les NUBRS et ses propriétés, puis la BTE et ses simplifications, et enfin l'IGA et son algorithme.

3.1 Les NURBS - Non Uniform Rational Base Splines

Les NURBS sont l'élément innovateur de la méthode IGA par rapport à la FEM. Utilisées uniquement pour le tracé dans le passé, elles commencent tout juste à montrer leurs avantages en analyse. Cette section se veut avant tout explicative : elle commence par définir les NURBS, puis elle montre leurs propriétés intéressantes pour l'analyse et enfin elle illustre les méthodes de raffinement des géométries NURBS via la modification de leurs paramètres.

3.1.1 Définition

Cette sous-section est très académique et la majorité des informations provient de [3].

Les NURBS sont un ensemble fini de fonctions rationnelles par morceaux. Cet ensemble est entièrement caractérisé par 3 objets mathématiques : le degré, le vecteur nodal et les poids. Comme elles ont principalement été utilisées pour le tracé, il est coutume de définir d'abord une géométrie NURBS qui a besoin d'un quatrième objet mathématique : les points de contrôle. Une attention doit être apportée à l'appellation. Si les NURBS sont des fonctions, une géométrie NURBS est une courbe, une surface ou un volume paramétré. Il est tout à fait possible d'avoir des géométries NURBS différentes avec les mêmes fonctions NURBS, en ne modifiant que les points de contrôle. Une explication sur ces objets mathématiques est fournie ci-dessous.

`Degré` :	`Entier` Degré des fonctions polynomiales intervenant au numérateur et au dénominateur des fonctions rationnelles(voir l'équation (3.1)). C'est le paramètre de raffinement p.(cf 3.1.4)
`Vecteur nodal` :	`Vecteur` Vecteur de réels positifs non décroissants. Les réels (appelés "noeuds") sont compris entre 0 et 1 dans le cadre de l'algorithme développé ici. Son rôle est multiple. Il contrôle notamment la densité locale des fonctions, et c'est le paramètre responsable du raffinement h. (cf 3.1.4) Le descriptif "Non Uniform" dans NURBS indique que les nœuds ne sont pas nécessairement équidistants.
`Points de contrôle` :	`Ensemble de points de l'espace` Aussi appelé polygone de contrôle(1D)/réseau de contrôle(2D). Une image serait de voir ces points comme des ancrages. Ces points sont utilisés pour le tracé des géométries, et n'affectent pas les fonctions en soi ; mais ils ont un impact sur la projection dans l'espace physique, via le Jacobien. La modification d'un point de contrôle engendre un changement local de la courbure autour de ce point. Á chaque fonction de l'ensemble est associée un point de contrôle.
`Poids` :	`Ensemble de réels` A chaque fonction est associée un poids. Les poids jouent un rôle de pondération.

Les équations vont être développées d'abord pour les B-splines puis pour les NURBS. Pour simplifier la compréhension, les équations et les illustrations qui suivent se limitent au cas 1D. Le passage aux dimensions supérieures est réalisé avec un produit tensoriel.

Les notations ci-dessous sont adoptées :

- u, la variable de paramétrisation
- p, le degrée
- $U = \{u_i, i \in [\![0; m]\!]\}$, le vecteur nodal, où $m = n + p + 1$
- $\{\mathbf{P}_i, i \in [\![0; n]\!]\}$, les points de contrôle, où n est le nombre de fonctions
- $\{\omega_i, i \in [\![0; n]\!]\}$, les poids

— $N_{i,p}, i \in [\![0; n]\!]$, la ième fonction B-splines de degrée p
— $R_{i,p}, i \in [\![0; n]\!]$, la ième fonction rationnelles de degrée p

Les fonctions B-splines sont les précurseurs des NURBS, elles s'obtiennent avec la formule de récurrence de De Boor, Cox et Mansfields [8] :

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{si } u_i \leq u < u_{i+1} \\ 0 & \text{sinon} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{3.1}$$

Les fonctions B-splines sont polynomiales par morceaux. Le cas de forme indéterminée $\frac{0}{0}$ est résolu en le prenant égal à 0. La formule est intrinsèquement récurrente. Le calcul d'une fonction de degré p en u fait appel à deux fonctions de degré p-1 en u. Il a une structure d'appel pyramidale, il est donc important d'éviter de répéter ces calculs lors de la conception d'algorithme.

Les fonctions NURBS sont alors définies comme :

$$R_{i,p}(u) = \frac{N_{i,p}(u)\omega_i}{\sum_{j=0}^{n} N_{j,p}(u)\omega_j} \tag{3.2}$$

Remarque :
Ces fonctions sont bien rationnelles par morceaux, d'où le qualificatif "Rational". Cependant, elles reviennent à des fonctions polynomiales par morceaux lorsque les poids ω_i sont égaux.

La courbe NURBS correspondante est définie par :

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{P}_i \tag{3.3}$$

En pratique, le tracé de la courbe à l'aide de logiciels de design impose les 4 paramètres définis ci-dessus. Néanmoins, il est possible d'insérer des nœuds au vecteur nodal ou d'augmenter le degré, et cela sans altérer la géométrie [4]. Le nombre de fonctions de l'ensemble croît avec ces deux manipulations (il faut alors aussi ajouter des points de contrôles et des poids), ce qui est très intéressant pour l'analyse. Qualifié de raffinement, ce dernier point va être développé dans la sous-section 3.1.4.

L'exemple suivant illustre les équations. La figure 3.1 montre le tracé d'un arc de cercle de rayon 1, de centre (0,0), allant de 0 à π :

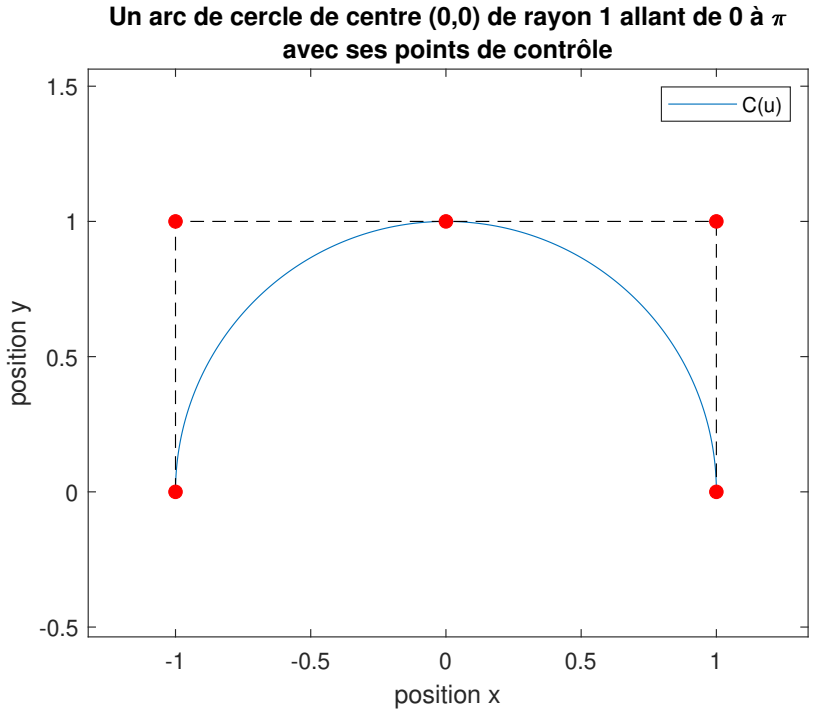


Figure 3.1 Un arc de cercle de centre (0,0) de rayon 1, allant de 0 à pi

Les paramètres correspondants sont :

Degré	$\mathrm{p} = 2$
Vecteur nodal	$U = \{0, 0, 0, 0.5, 0.5, 1, 1, 1\}$
Points de contrôle	$\{\mathbf{P}_i\} = \left\{ \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}$
Poids	$\{\omega_i\} = \left\{1, \frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}, 1\right\}$

Les fonctions B-splines sont obtenues avec l'équation (3.1), puis les NURBS sont calculées avec l'équation (3.2). Le détail des calculs sont laissés en annexe (A.1), les expressions des NURBS sont présentées dans le tableau 3.1

Tableau 3.1 Un exemple de fonctions NURBS 1D de degré 2

Nom	Expression
$R_{0,2}$	$\frac{(0.5-u)^2}{(0.5-u)^2+\sqrt{2}u(0.5-u)+u^2}$ $u \in [0, 0.5[$
$R_{1,2}$	$\frac{\sqrt{2}u(0.5-u)}{(0.5-u)^2+\sqrt{2}u(0.5-u)+u^2}$ $u \in [0, 0.5[$
$R_{2,2}$	$\frac{u^2}{(0.5-u)^2+\sqrt{2}u(0.5-u)+u^2}$ $u \in [0, 0.5[$ $\frac{(1-u)^2}{(1-u)^2+\sqrt{2}(1-u)(u-0.5)+(u-0.5)^2}$ $u \in [0.5, 1]$
$R_{3,2}$	$\frac{\sqrt{2}(1-u)(u-0.5)}{(1-u)^2+\sqrt{2}(1-u)(u-0.5)+(u-0.5)^2}$ $u \in [0.5, 1]$
$R_{4,2}$	$\frac{(u-0.5)^2}{(1-u)^2+\sqrt{2}(1-u)(u-0.5)+(u-0.5)^2}$ $u \in [0.5, 1]$

Remarque :

— le $\sqrt{2}$ au dénominateur et au numérateur dans le tableau 3.1 est dû au poids $1/\sqrt{2}$. Si les poids avaient été égaux, le dénominateur serait simplifié et les fonctions seraient bien seulement polynomiales [3].

— l'équation paramétrique de la courbe peut être calculée avec (3.3). En notant $\mathbf{C}(u) = \begin{pmatrix} x(u) \\ y(u) \end{pmatrix}$, il est vérifié que $x(u)^2 + y(u)^2 = 1 \ \forall \ u \in [0, 1[$. L'arc est exactement décrit par $\mathbf{C}$. Si les poids étaient uniformes, la courbe paramétrée serait une parabole et vérifierait $y(u) = (x(u) + 1)\,(x(u) - 1) \ \forall \ u \in [0, 1[$

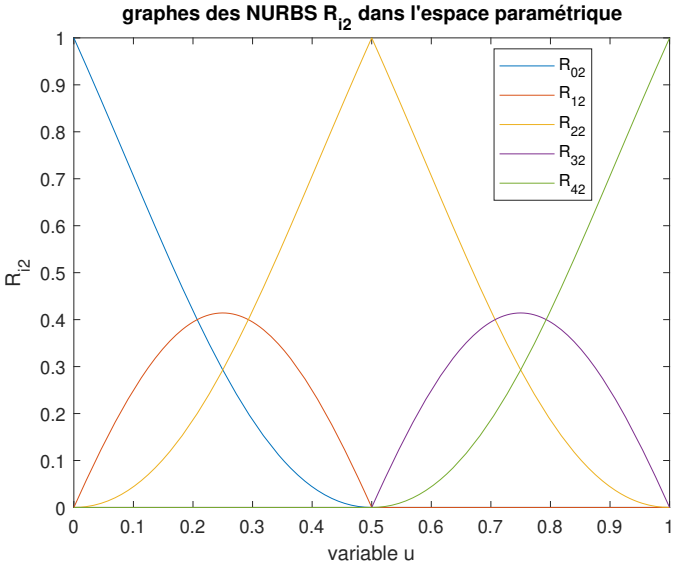


Figure 3.2 Les graphes des NURBS du tableau 3.1 dans l'espace paramétrique

Si les NURBS sont définies dans l'espace paramétrique $\hat{\Omega}$, il est possible de définir leur projections dans l'espace physique Ω. La fonction de passage $\mathcal{F} : u \mapsto \mathbf{C}(u)$, ou push-

forward, est bijective. Il est possible de définir $\mathcal{F}^{-1}$ et de construire $R_{i,p} \circ \mathcal{F}^{-1}$, qui est définie dans Ω. Ainsi, il est possible de tracer les graphes des NURBS en parcourant la courbe $\mathbf{C}$. De fait, les calculs de l'IGA sont réalisés dans l'espace paramétrique puis la solution est visualisée dans le domaine physique. Cependant une alternative est de se dire que la solution est écrite comme une somme de $R_{i,p} \circ \mathcal{F}^{-1}$ [9], ce qui pourrait paraître plus naturel et intuitif. (et de comprendre pourquoi l'IGA n'introduit aucune erreur liée à la géométrie, puisque les fonctions de la base de décomposition sont définies sur la courbe elle-même)

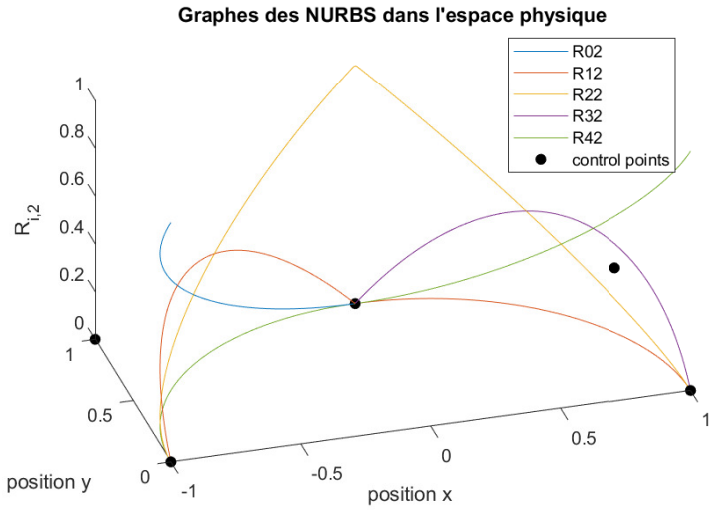


Figure 3.3 Les graphes des NURBS du tableau 3.1 dans l'espace physique

Pour une surface l'ensemble des fonctions est construit par produit tensoriel de deux ensembles de fonctions NURBS indépendants, chacun pour une direction de l'espace paramétrique.

Ainsi pour des NURBS de degré p dans la direction u et des NURBS de degré q dans la direction v, les NURBS 2D et l'équation paramétrique de la surface sont respectivement données par [3] :

$$R_{i,j}(u,v) = \frac{N_{i,p}(u) N_{j,q}(v) \omega_{i,j}}{\sum_{k=0}^{n} \sum_{l=0}^{m} N_{k,p}(u) N_{l,q}(v) \omega_{k,l}} \tag{3.4}$$

$$\mathbf{S}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}(u,v) \mathbf{P}_{i,j} \tag{3.5}$$

Il est possible de généraliser la formule pour un volume. Ce mémoire se limitera à des géométries 2D.

3.1.2 Organisation de la géométrie

Avant d'aborder les propriétés des fonctions NURBS et les raffinements, il est nécessaire de clarifier l'organisation de la géométrie.
Pour répondre aux besoins et aux contraintes de l'analyse, le domaine d'étude est généralement représenté par plusieurs géométries NURBS, appelées patches. Le patch est l'équivalent de l'élément de la FEM classique. La division en patches est fonction de la variation spatiale des propriétés du milieu, et elle doit répondre à des contraintes de conception imposées par l'analyse. Comparativement à la FEM classique, le nombre de patches avec l'IGA reste faible. Alors que la FEM classique requiert souvent un grand nombre d'éléments pour assurer la précision de la solution, l'IGA augmente le degré de sa discrétisation spatiale à une sous-échelle de patch, appelée, malencontreusement, élément. La division d'un patch en éléments est réalisée en fonction du vecteur nodal. Le raffinement h de la méthode IGA consiste donc à ajouter des nœuds au vecteur, tandis qu'un équivalent strict du raffinement h à la FEM serait de subdiviser un patch en de plus petits patches. Cela nécessiterait de recalculer une division du domaine d'étude, plus coûteuse que l'ajout de nœuds et ce n'est pas la méthode courante.
Par ailleurs, il faut noter que les fonctions NURBS sont définies à partir du patch qui constitue leur domaine de définition dans l'espace physique. La solution numérique globale obtenue est donc discontinue à travers deux patches. Elle est cependant au moins continue à travers deux éléments.

3.1.3 Propriétés des NURBS

Cette section ne se veut pas une liste exhaustive des propriétés des NURBS, elle a pour but de mettre en lumière celles qui sont utiles à l'IGA. Le lecteur pourrait se référer aux graphes des figures 3.2 et 3.3 pour avoir une illustration. Une liste complète des propriétés peut être trouvée dans [3].

1. Non-négativité : $R_{i,p}(u) \geq 0 \ \forall i, p$ `et` $\forall u \in [0,1]$
2. Partion d'unité : $\sum_{i=0}^{n} R_{i,p}(u) = 1 \ \forall p$ `et` $\forall u \in [0,1]$
   Il est trivial d'ajouter un offset et aisé d'imposer une condition de Dirichlet homogène sur le bord. (bien que non pertinent pour l'application en génie nucléaire)
3. Localité :$R_{i,p}(u) = 0$ `si` $u \notin [u_i, u_{i+p+1}[$. De plus $\forall k \in [\![0; m-1]\!]$, il y a au plus p+1 fonctions non identiquement nulles sur l'intervalle $[u_k, u_{k+1}[$.
   Cette propriété est essentielle à l'analyse pourvu que le problème présente quelques singularités ou hétérogénéités. Chaque fonction est non nulle seulement sur au plus $p+1$ éléments.

4. Régularité : Les NURBS sont C^∞ entre deux noeuds successifs. Sur un noeud, elles sont C^{p-l} où l est la multiplicité du noeud (i.e le nombre de répétition de ce noeud). Pour avoir une meilleure solution numérique, les fonctions devraient être aussi régulières que possibles. (par exemple la fonction R_{22} est C^0 en $u = 0.5$ ($l_{0.5} = 2$), ce qui est peu souhaitable)
5. L'espace paramétrique est simple, ce qui facilite les calculs.
   Dans ce mémoire l'espace paramétrique est le segment [0,1] en 1D, et le carré $[0, 1]^2$ en 2D. En général, l'espace paramétrique est au plus polynomial. L'usage du package NURBS Toolbox inclus dans GeoPDE restreint l'algorithme à des vecteurs nodaux dit ouverts (commençant par des 0 et finissant par des 1, chacun de multiplicité p+1).

3.1.4 Raffinement - Influence du vecteur nodal et du degré

Pour la FEM traditionnelle, deux moyens de raffinement existent pour augmenter la précision de la solution numérique. Il en est de même pour les NURBS, d'une part via l'insertion de noeud et d'autre part via l'augmentation de degré [4]. Pour les NURBS, ces deux approches ont l'avantage de ne pas altérer la géométrie, et donc ne font pas appel de nouveau aux logiciels de CAD.
Les illustrations dans cette sous-section sont en 1D ; les lecteurs intéressés pourraient voir les figures 3.16, 3.17 et 3.18 pour des illustrations avec des NURBS en 2D.

L'insertion de noeuds est analogue au **raffinement h**. La division du maillage en éléments est faite en fonction du vecteur nodal, de sorte qu'il y ait le même nombre de fonctions non nulles sur chaque élément. L'ajout de nouveaux noeuds au vecteur nodal a donc quatre conséquences :

1. Le nombre de NURBS augmente, i.e, le nombre de degré de liberté augmente (l'appellation degré de liberté est définie plus tard, c.f 3.2.4).
2. Le nombre d'éléments augmente. Les éléments autour du nœud inséré délimitent chacun une part plus petite du domaine physique. Avec l'augmentation du nombre de NURBS, la densité (pas au sens mathématique) locale des fonctions augmente.
3. Les nouvelles NURBS sont non nulles sur une plus petite partie de l'espace paramétrique. (cf la propriété "Localité" de la sous-section 3.1.3) Les variations fines de la solution peuvent être mieux décrites.
4. Les poids et les points de contrôles sont ajustés pour compenser l'altération apportée par l'insertion de nœuds. L'expression paramétrique de la géométrie reste inchangée.

Les théories mathématiques pour savoir quels ajustements sont à apporter ne sont pas étudiées dans ce mémoire.

Pour illustration, la courbe de la figure 3.1 est réutilisée. Le noeud $\{0.25\}$ est ajouté au vecteur. Le tableau de comparaison suivant permet de visualiser les modifications apportées :

Tableau 3.2 Tableau de comparaison avant et après insertion d'un noeud

	Avant insertion	après insertion
U	$\{0,0,0,0.5,0.5,1,1,1\}$	$\{0,0,0,0.25,0.5,0.5,1,1,1\}$
Degré de liberté	5	6
Éléments	[0 ;0.5[,[0.5,1]	[0 ;0.25[,[0.25 ;0.5[,[0.5 ;0.1]
Fontions non nulles (leurs indices)	$\{0,1,2\}$,$\{2,3,4\}$	$\{0,1,2\}$,$\{1,2,3\}$,$\{3,4,5\}$

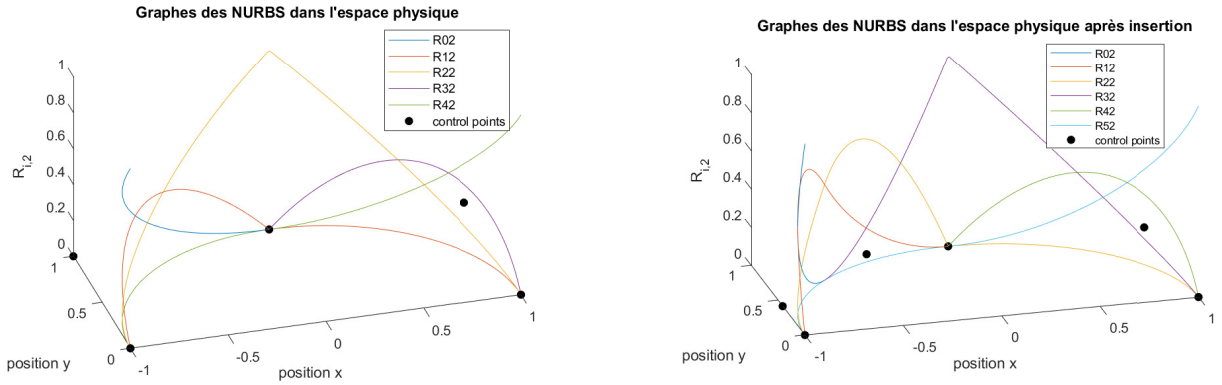


Figure 3.4 Comparaison des graphes avant et après insertion

Il est possible de remarquer [1] que le **nœud inséré affecte les fonctions qui étaient non nulles autour de ce nœud**. Les nouvelles fonctions sont notées avec un '. $R'_{0,2}$ s'annule désormais à partir de $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ au lieu de $(0.5, 1)$. $R_{1,2}$ n'existe plus, elle est remplacée par $R'_{1,2}$ et $R'_{2,2}$ qui sont symétriques (pour la partie non nulle) par rapport à $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ où $R_{1,2}$ prend son maximum. Ceci s'explique par le fait que 0.25 soit inséré à équidistance de 0 et 0.5. $R_{2,2}$ n'est affectée que pour la partie définie sur [0,0.5[. Les fonctions $R_{3,2}$ et $R_{4,2}$ ne sont pas affectées par le noeud inséré et correspondent exactement aux fonctions $R'_{4,2}$ et $R'_{5,2}$ après insertion. Il est donc possible d'**insérer stratégiquement des noeuds** pour étudier des zones plus intéressantes que d'autres.

Lorsque le nœud ajouté est un nœud déjà existant dans le vecteur nodal, il n'y a pas de

1. Pour faciliter la lecture du nouveau graphe, il faut savoir que les NURBS atteignent exactement un maximum

raffinement h (pas de nouvelles fonctions, pas de nouvelles subdivisions en éléments). En revanche, la multiplicité du nœud augmente et la régularité de la base en est affectée. Pour un nœud interne (différent de 0 et 1), la multiplicité est au plus égale au degré p.

L'augmentation du degré est analogue au **raffinement p**. Contrairement à l'approche précédente, cette approche est globale. Elle a les conséquences suivantes :

- Le degré et le nombre des NURBS augmentent.
- Les éléments ne sont pas affectés, en revanche le nombre de fonctions non nulles par élément augmente. (cf la propriété "Localité" de la sous-section 3.1.3) La densité globale (pas au sens mathématique) de fonctions augmente, ce qui contribue à la précision de la solution.
- La multiplicité de chacun des noeuds augmente, de sorte que la régularité de la base soit conservée.
- Les poids et les points de contrôles sont ajustés pour compenser l'augmentation de degré. L'expression paramétrique de la géométrie reste inchangée. De nouveau, les théories mathématiques pour savoir quels ajustements sont à apporter ne sont pas étudiées dans ce mémoire.

Pour illustrer, le degré de la courbe 3.1 est augmenté de 1. Le même tableau de comparaison peut être dressé :

Tableau 3.3 Tableau de comparaison avant et après élévation du degré

	Degré 2	Degré 3
U	$\{0,0,0,0.5,0.5,1,1,1\}$	$\{0,0,0,0,0.5,0.5,0.5,1,1,1,1\}$
Degré de liberté	5	7
Éléments	[0 ;0.5[,[0.5,1]	[0 ;0.5[,[0.5 ;0.1]
Fontions non nulles (indices)	$\{0,1,2\}$,$\{2,3,4\}$	$\{0,1,2,3\}$,$\{3,4,5,6\}$

Il n'est pas possible de baisser le degré, du moins avec le package NURBS Toolbox utilisé pour l'algorithme de ce mémoire. Il faut noter que certaines courbures nécessitent un degré minimum pour être représentées - les objets coniques ont besoin d'un degré supérieur ou égal à 2. En revanche, NURBS Toolbox assure que le degré minimal est utilisé lors de la conception des géométries.

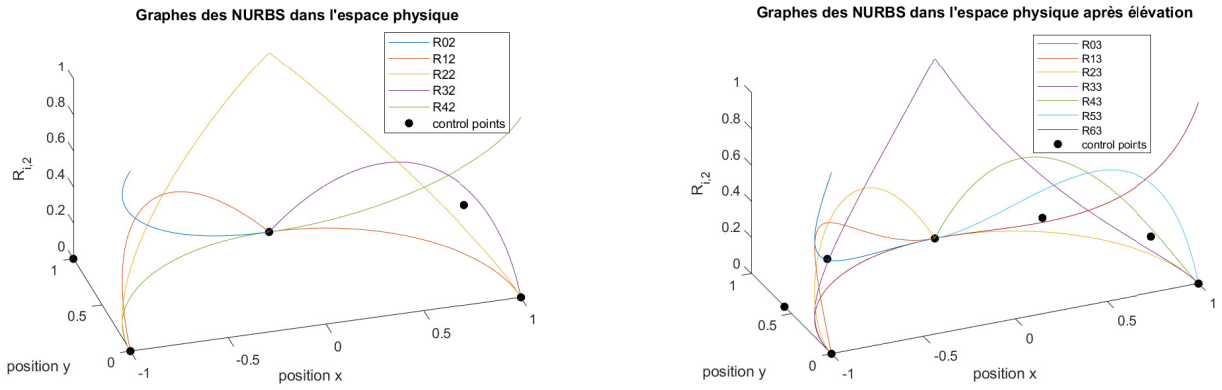


Figure 3.5 Comparaison des graphes avant et après élévation de degré

Les deux approches peuvent être simultanément utilisées Cependant les deux approches ne sont **pas commutatives** [4], par le simple fait que l'augmentation de degré conserve la régularité de la base. Il est conseillé d'insérer des nœuds après l'élévation de degré pour avoir une meilleure régularité de la base. C'est-à-dire privilégier un raffinement p-h à un raffinement h-p. L'exemple de la figure 3.6 correspond à l'insertion des nœuds {0.25 0.25} et l'élévation du degré de 1. La multiplicité 2 du nœud est choisie volontairement parce que la différence C^0(pour h-p) et C^1(pour p-h) à la traversée du noeud 0.25 est plus visible à l'œil.

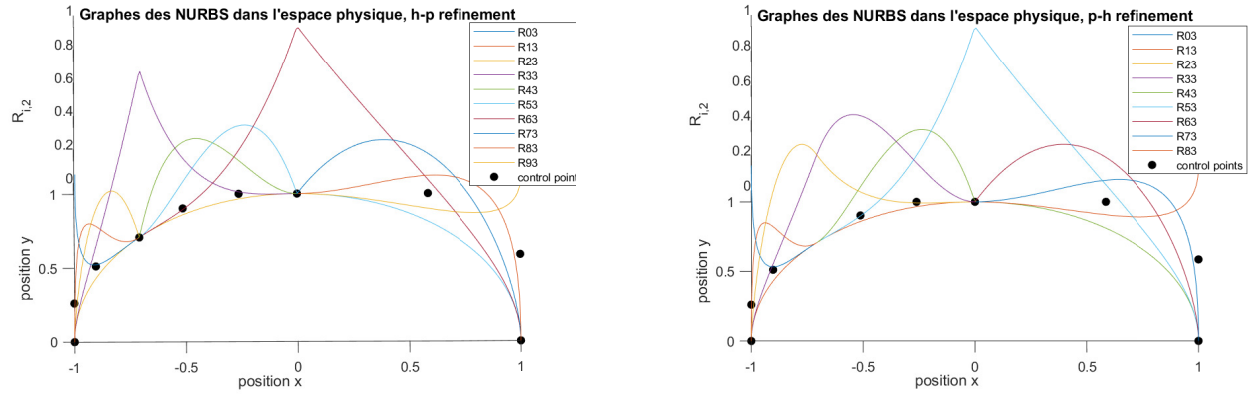


Figure 3.6 Comparaison des graphes pour raffinement h-p et p-h

3.2 L'équation de transport de Boltzmann

La BTE est l'équation fondamentale de la neutronique. Son établissement adopte la logique de la mécanique statistique : l'évolution des populations de neutrons est mise en équation via le comptage de ceux qui entrent et de ceux qui sortent, de ceux qui sont crées et de ceux qui ont été consommés. Dans un premier temps, cette section présentera brièvement la dérivation classique de la BTE. Ensuite la lumière est mise sur les différentes discrétisations impliquées dans la résolution numérique de l'équation.

3.2.1 Mise en équation

Cette mise en équation peut être retrouvée avec plus de détail dans [10] , où une liste exhaustive des hypothèse est explicitée. Une version concise est présentée dans ce mémoire.

Les notations adoptées :

- $\mathbf{r}$, la variable position
- E, la variable énergie définie par $E = \frac{1}{2} m_{neutron} V^2$ avec $m_{neutron}$ la masse du neutron et V le module de la vitesse des neutrons
- $\mathbf{\Omega}$ la variable angulaire, désignant la direction des neutrons
- t, la variable de temps
- $n(\mathbf{r}, E, \mathbf{\Omega}, t)$, la densité de neutrons en $\mathbf{r}$, avec l'énergie E, dans la direction $\mathbf{\Omega}$ à l'instant t
- $\Phi(\mathbf{r}, E, \mathbf{\Omega}, t) = Vn(\mathbf{r}, E, \mathbf{\Omega}, t)$, le flux angulaire. C'est une construction mathématique et c'est la variable de choix en neutronique.

En faisant un bilan de neutrons pour un volume de contrôle arbitraire, à une petite variation près pour chaque variable, l'équation suivante est obtenue :

$$\frac{\partial n(\mathbf{r}, E, \mathbf{\Omega}, t)}{\partial t} = -\nabla \cdot (\mathbf{\Omega}\Phi(\mathbf{r}, E, \mathbf{\Omega}, t)) - \Sigma(\mathbf{r}, E)\Phi(\mathbf{r}, E, \mathbf{\Omega}, t) + Q(\mathbf{r}, E, \mathbf{\Omega}, t) \tag{3.6}$$

Où $\Sigma(\mathbf{r}, E)$ est la section efficace totale et $Q(\mathbf{r}, E, \mathbf{\Omega}, t)$ est le terme source.
Avec l'hypothèse de régime stationnaire, et quelques manipulations mathématiques, la BTE sous sa forme différentielle est obtenue :

$$\mathbf{\Omega} \cdot \nabla\Phi(\mathbf{r}, E, \mathbf{\Omega}) + \Sigma(\mathbf{r}, E)\Phi(\mathbf{r}, E, \mathbf{\Omega}) = Q(\mathbf{r}, E, \mathbf{\Omega}) \tag{3.7}$$

Le terme source peut être développé en 3 composants :

$$Q(\mathbf{r}, E, \mathbf{\Omega}) = \underbrace{\int_{4\pi} d^2\Omega' \int_0^\infty dE' \, \Sigma_s(\mathbf{r}, E \leftarrow E', \Omega \leftarrow \Omega')\Phi(\mathbf{r}, E', \mathbf{\Omega'})}_{\texttt{terme A}} + \underbrace{\frac{1}{4\pi K_{eff}} Q^{fis}(\mathbf{r}, E)}_{\texttt{terme B}} + Q_{fixe}(\mathbf{r}, E, \mathbf{\Omega}) \tag{3.8}$$

A est le terme de **diffusion**.
$\Sigma_s(\mathbf{r}, E \leftarrow E', \mathbf{\Omega} \leftarrow \mathbf{\Omega'})$ est la section efficace macroscopique de diffusion et de réaction (n,xn) en $\mathbf{r}$ pour les neutrons d'énergie initiale E' et de direction initiale $\mathbf{\Omega'}$ et qui ont l'énergie E et la direction $\mathbf{\Omega}$ après diffusion.

B est le terme de **fission**. La fission est supposée isotrope.
K_{eff} est le facteur de multiplication effectif introduit pour équilibrer le nombre de neutrons produits et le nombre de neutrons consommés afin de maintenir la condition de régime stationnaire.

Q^{fis} désigne la source de fission isotrope, voir (3.10)

Q_{fixe} est le terme **source externe** imposé par l'utilisateur. En pratique, il intervient seulement dans les problèmes à source et il n'apparaît pas dans les problèmes à fission.

Remarque :
Avec l'hypothèse d'un milieu isotrope, seul l'angle entre la direction initiale $\mathbf{\Omega'}$ et la direction diffusée $\mathbf{\Omega}$ compte. Le `terme A` peut être réécrit comme :

$$\texttt{terme A} = \frac{1}{2\pi} \int_{4\pi} d^2\mathbf{\Omega'} \int_0^\infty dE' \, \Sigma_s(\mathbf{r}, E \leftarrow E', \mathbf{\Omega} \cdot \mathbf{\Omega'})\Phi(\mathbf{r}, E', \mathbf{\Omega'}) \tag{3.9}$$

La source de fission a pour expression :

$$Q^{fis}(\mathbf{r}, E) = \sum_{j=1}^{J_{fis}} \chi_j(E) \int_0^\infty dE' \, \nu\Sigma_{f,j}(\mathbf{r}, E')\Phi(\mathbf{r}, E') \tag{3.10}$$

J^{fis}	est le nombre total d'isotopes fissiles
$\chi_j(E)$	est le spectre de fission de l'isotope j. Il s'agit d'une densité de probabilité. Le neutron à l'issu d'une fission d'isotope j a une probabilité $\chi_j(E)$ d'avoir l'énergie E à dE près.
$\nu\Sigma_{f,j}$	est un produit couramment présenté ensemble. ν est le nombre moyen de protons émis par fission d'isotope j, et $\Sigma_{f,j}$ est la section efficace macroscopique de fission de l'isotope j.
$\Phi(\mathbf{r}, E')$	est le flux intégré d'énergie E'. Il est défini par : $\Phi(\mathbf{r}, E') = \int_{4\pi} d^2\mathbf{\Omega}\ \Phi(\mathbf{r}, E', \mathbf{\Omega})$ La fission ne dépend pas de l'angle incident du neutron.

En remplaçant (3.10) dans (3.8), puis (3.8) dans (3.7), la BTE est entièrement développée. Avec les outils mathématiques disponibles, cette équation n'admet que rarement une solution analytique. C'est pourquoi une solution numérique est nécessaire. La méthode IGA utilisée dans ce mémoire est déterministe (versus les méthodes stochastiques). Elle est fortement similaire à la méthode DGFEM, elles suivent la même logique de résolution. Dans la suite de ce chapitre, la discrétisation des différentes variables de la BTE sera présentée, afin de montrer étape après étape, les modifications apportées à cette équation. Sa résolution sera présentée dans le chapitre suivant, avec le schéma de calcul de la méthode IGA.
Pour alléger les notations, un seul isotope fissile sera considéré.

3.2.2 La discrétisation multigroupe

Cette approche consiste à remplacer la variable continue E par un ensemble discret d'énergie $\{E_0 > E_1 > .. > E_G\}$ [10]. Ce qui donne lieu à G intervalles d'énergie $]E_{g-1}, E_g]$. Chaque intervalle constitue un groupe d'énergie (d'où le nom multigroupe), sur lequel les neutrons sont supposés avoir une seule vitesse (donc une seule énergie moyenne). Cela permet de simplifier les calculs, car la BTE peut être résolue sur chaque groupe ; une grandeur dépendante de E peut être remplacée par une grandeur moyennée sur ce groupe par rapport à E et qui en devient alors indépendante. L'intégrale sur l'énergie peut aussi être remplacée par :

$$\int_0^\infty dE' = \sum_{g=1}^{G} \int_{E_g}^{E_{g-1}} dE' \tag{3.11}$$

Définitions des grandeurs moyennées par rapport à E :

$$\Phi^g(\mathbf{r}, \mathbf{\Omega}) = \int_{E_g}^{E_{g-1}} dE'\ \Phi^g(\mathbf{r}, \mathbf{\Omega}, E') \tag{3.12}$$

$$\Phi^g(\mathbf{r}) = \int_{E_g}^{E_{g-1}} dE' \; \Phi^g(\mathbf{r}, E') = \int_{4\pi} d^2\mathbf{\Omega} \; \Phi^g(\mathbf{r}, \mathbf{\Omega}) \tag{3.13}$$

$$\Sigma^g(\mathbf{r}) = \frac{\int_{E_g}^{E_{g-1}} dE' \; \Sigma(\mathbf{r}, E')\Phi(\mathbf{r}, \mathbf{\Omega}, E')}{\int_{E_g}^{E_{g-1}} dE' \; \Phi(\mathbf{r}, \mathbf{\Omega}, E')} \tag{3.14}$$

$$Q^g_{fixe}(\mathbf{r}, \mathbf{\Omega}) = \int_{E_g}^{E_{g-1}} dE' \; Q_{fixe}(\mathbf{r}, \mathbf{\Omega}, E') \tag{3.15}$$

$$\chi^g = \int_{E_g}^{E_{g-1}} dE' \; \chi(E') \tag{3.16}$$

$$\nu\Sigma^g_f(\mathbf{r}) = \frac{\int_{E_g}^{E_{g-1}} dE' \; \nu\Sigma_f(\mathbf{r}, E')\Phi(\mathbf{r}, E')}{\int_{E_g}^{E_{g-1}} dE' \; \Phi(\mathbf{r}, E')} \tag{3.17}$$

Quant à la section efficace de diffusion, elle est usuellement développée en polynômes de Legendre jusqu'à un ordre L pour exprimer facilement sa dépendance vis-à-vis de $\mathbf{\Omega} \cdot \mathbf{\Omega}'$:

$$\Sigma_s(\mathbf{r}, E \leftarrow E', \mathbf{\Omega} \cdot \mathbf{\Omega}') = \sum_{l=0}^{L} \frac{2l+1}{2} \Sigma_{s,l}(\mathbf{r}, E \leftarrow E') P_l(\mathbf{\Omega} \cdot \mathbf{\Omega}') \tag{3.18}$$

Les grandeurs à moyenner pour la diffusion sont donc les coefficients $\Sigma_{s,l}(\mathbf{r}, E \leftarrow E')$:

$$\Sigma^{g \leftarrow g'}_{s,l}(\mathbf{r}) = \frac{\int_{E_g}^{E_{g-1}} dE' \; \int_{E_{g'}}^{E_{g'-1}} dE'' \; \Sigma_{s,l}(\mathbf{r}, E' \leftarrow E'')\Phi(\mathbf{r}, \mathbf{\Omega}', E'')}{\int_{E_{g'}}^{E_{g'-1}} dE'' \; \Phi(\mathbf{r}, \mathbf{\Omega}', E'')} \tag{3.19}$$

En plus, ce développement permet de simplifier l'intégrale par rapport à l'angle $\mathbf{\Omega}$ dans le terme de diffusion en utilisant les moments du flux angulaire en harmoniques sphériques et le théorème d'addition. Cette démarche traditionnelle n'est pas adoptée dans ce mémoire, car le développement est limité à l'ordre $L = 1$ (anisotropie d'ordre 1, $L = 0$ correspondant à une diffusion isotrope). Sachant que $P_0(\mathbf{\Omega} \cdot \mathbf{\Omega}') = 1$ et $P_1(\mathbf{\Omega} \cdot \mathbf{\Omega}') = \mathbf{\Omega} \cdot \mathbf{\Omega}'$, les deux intégrales demeurent faciles à calculer.
Il est possible de remarquer au passage que les définitions de $\nu\Sigma^g_f$ et des $\Sigma^{g \leftarrow g'}_{s,l}$ permettent de conserver le taux de réaction.

Autant $\Phi^g(\mathbf{r}, \mathbf{\Omega})$ est la solution numérique que l'algorithme cherche à calculer, autant les sections efficaces moyennées sont fournies par des librairies de DRAGON. Leurs calculs constituent une tâche en soi et ne sont pas investigués dans ce mémoire.

Ainsi à partir de la BTE continue, G équations moyennées sont obtenues :

$$(\mathbf{\Omega}\cdot\nabla+\Sigma^g(\mathbf{r}))\Phi^g(\mathbf{r},\mathbf{\Omega})=\sum_{g'=1}^{G}\sum_{l=0}^{L}\frac{2l+1}{4\pi}\Sigma_{s,l}^{g\leftarrow g'}(\mathbf{r})\int_{4\pi}d^2\mathbf{\Omega}'\ P_l(\mathbf{\Omega}\cdot\mathbf{\Omega}')\Phi^{g'}(\mathbf{r},\mathbf{\Omega}') + Q^g(\mathbf{r},\mathbf{\Omega}) \tag{3.20}$$

avec

$$Q^g(\mathbf{r},\mathbf{\Omega})=\begin{cases} Q^g_{fixe}(\mathbf{r},\mathbf{\Omega}) & \texttt{pour un problème à source fixe} \\ \frac{\chi^g}{4\pi K_{eff}}\sum\limits_{g'=1}^{G}\nu\Sigma_f^{g'}(\mathbf{r})\Phi^{g'}(\mathbf{r}) & \texttt{pour un problème à fission} \end{cases} \tag{3.21}$$

3.2.3 La discrétisation en angle - la méthode des ordonnées discrètes

La deuxième variable à discrétiser est l'angle $\mathbf{\Omega}$. La méthode des ordonnées discrètes, dite méthode S_N consiste à discrétiser la variable continue $\mathbf{\Omega}$ en un ensemble fini de directions (ordonnées) $\{\mathbf{\Omega_k}, k\in[\![1;M]\!]\}$. Pour les cas 3D, 2D et 1D, le nombre d'ordonnées M est respectivement égal à $N(N+2)$, $N(N+2)/2$ et N.
Sous cette discrétisation, les neutrons prennent nécessairement une des directions appartenant à $\{\mathbf{\Omega_k}\}$. Mathématiquement parlant, l'intégrale sur $\mathbf{\Omega}$ devient nulle. Pour approximer correctement l'intégrale, une quadrature est introduite. Un poids ω_k est associé à chaque ordonnée $\mathbf{\Omega}_k$, de sorte que :

$$\int_{4\pi}d^2\mathbf{\Omega}\approx\sum_{k=1}^{M}\omega_k \tag{3.22}$$

La distribution des ordonnées sur la sphère unité et la quadrature associée sont calculées par DRAGON. Ce mémoire se contente de les réutiliser. Plus de renseignements sur ces calculs sont fournis dans [10]. Il faut souligner que cette discrétisation a été optimisée par DRAGON pour des intégrands polynomiaux. Leur utilisation avec les NURBS est expérimentale. L'ensemble d'ordonnées utilisé est qualifié de Level-Symétric (d'autres ensembles d'ordonnées existent [11]), c'est-à-dire qu'il présente les propriétés suivantes :

- Si $(\mu,\eta,\xi)\in\{\mathbf{\Omega_k}\}$, alors $(\pm\mu,\pm\eta,\pm\xi)\in\{\mathbf{\Omega_k}\}$
- Si $(\mu,\eta,\xi)\in\{\mathbf{\Omega_k}\}$, alors toutes permutation de μ,η et ξ sont également dans $\{\mathbf{\Omega_k}\}$

Il en découle une invariance par rotation de $\pi/2$ autour des axes cartésiens pour les ordonnées. Pour illustration, la figure 3.7 représente les ordonnées S_4 et S_{16} utilisées dans ce mémoire. La figure 3.8 représente les ordonnées S_{16} en vues planes pour mieux souligner l'invariance par rotation et son aspect Level-symétrique.

Seules les ordonnées de la moitié supérieure de la sphère sont représentées, car les études

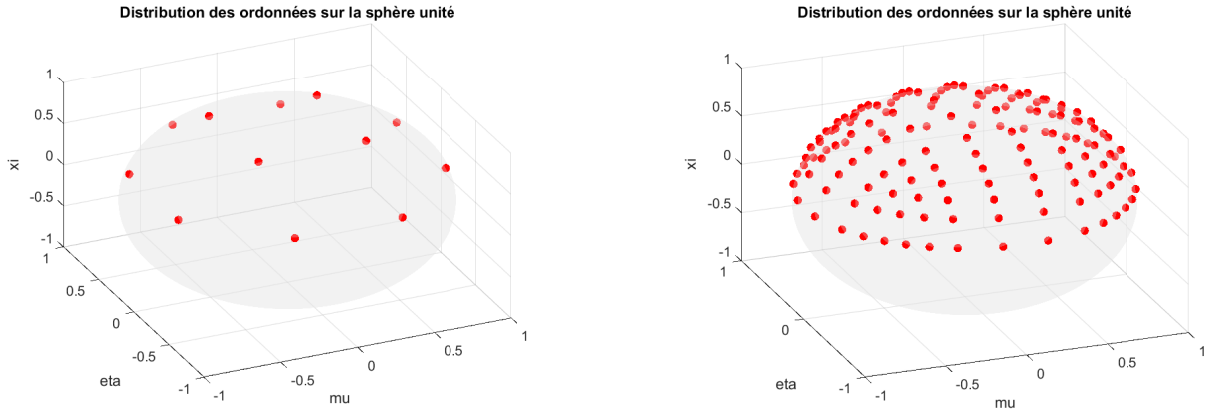


Figure 3.7 Distribution des ordonnées S_4 et S_{16} sur la sphère unité. Seules les ordonnées avec $\xi > 0$ sont représentées

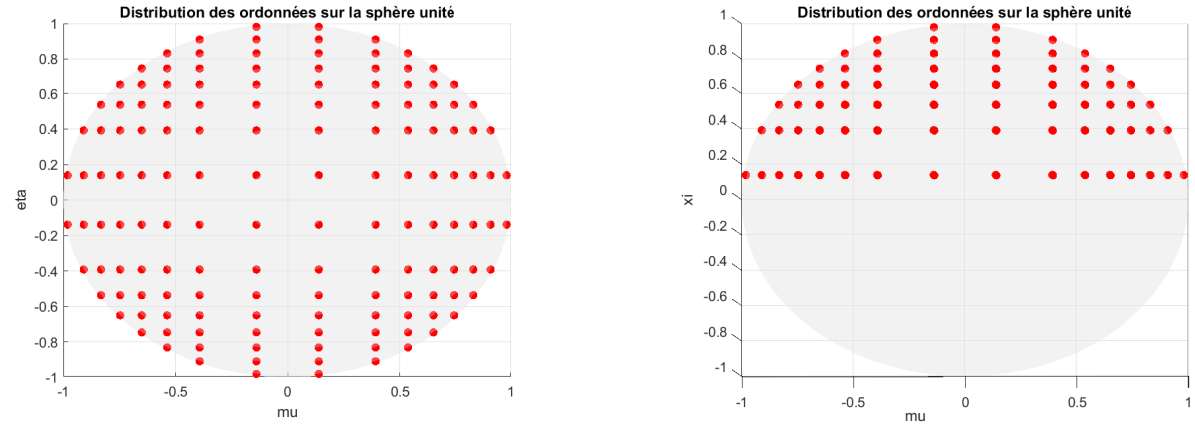


Figure 3.8 Distribution des ordonnées S_{16} sur la sphère unité : vue plane (μ,η) et vue plane (μ,ξ). Seules les ordonnées avec $\xi > 0$ sont représentées

seront restreintes à des géométries 2D. Par soucis de concision, il est coutume de représenter seulement un des huit octants de la sphère, les autres ordonnées pouvant être retrouvées par symétrie.

Enfin, il est possible de réécrire les G équations multigroupe de (3.20). Pour un groupe d'énergie g, et une ordonnée $\mathbf{\Omega}_k$, l'équation se réécrit :

$$(\mathbf{\Omega}_k \cdot \nabla + \Sigma^g(\mathbf{r}))\Phi_k^g(\mathbf{r}) = \sum_{g'=1}^{G}\sum_{l=0}^{L}\frac{2l+1}{4\pi}\Sigma_{s,l}^{g\leftarrow g'}(\mathbf{r})\sum_{k'=1}^{M}\omega_{k'}P_l(\mathbf{\Omega}_k\cdot\mathbf{\Omega}_{k'})\Phi_{k'}^{g'}(\mathbf{r}) + Q_k^g(\mathbf{r}) \tag{3.23}$$

où $\Phi_{k'}^{g'}$ est le flux angulaire de groupe d'énergie g' et de direction $\mathbf{\Omega_{k'}}$ et

$$Q_k^g(\mathbf{r}) = \begin{cases} Q_{fixe,k}^g(\mathbf{r}) & \texttt{pour un problème à source fixe} \\ \frac{\chi^g}{4\pi K_{eff}} \sum\limits_{g'=1}^{G} \nu\Sigma_f^{g'}(\mathbf{r})\Phi^{g'}(\mathbf{r}) & \texttt{pour un problème à fission} \end{cases} \tag{3.24}$$

Remarques :

- Ces $G \times M$ équations ne dépendent plus que de $\mathbf{r}$.
- Elles sont couplées via le terme de diffusions. La méthode de source itérée sera utilisée pour traiter ce couplage. Ce couplage faible laisse la possibilité de paralléliser la résolution.
- L'opérateur de convection $\mathbf{\Omega}_k \cdot \nabla$ suggère d'introduire un sens de balayage de $\mathbf{r}$ en fonction de $\mathbf{\Omega}_k$.

3.2.4 La discrétisation spatiale

La discrétisation spatiale intervient à deux niveaux : d'abord lors de la phase de conception de la géométrie puis lors de la résolution.

Lors de la conception, la géométrie est divisée en patches comme le montre la figure 3.9. Chaque patch est un objet NURBS et joue le rôle équivalent d'un élément en FEM. Sur chaque patch, les paramètres physiques tels que les sections efficaces sont constants. En génie nucléaire, la division est plutôt intuitive, chaque patch correspondant souvent à un matériau différent. La résolution de l'équation se fait alors de patch en patch. La solution peut être discontinue entre deux patches, mais elle est au moins continue à l'intérieur d'un patch.

Pour un patch donné, l'équation pour un groupe g, une ordonnée $\mathbf{\Omega}_k$ s'écrit :

$$(\mathbf{\Omega}_k \cdot \nabla + \Sigma^g)\Phi_k^g(\mathbf{r}) = \sum_{g'=1}^{G}\sum_{l=0}^{L} \frac{2l+1}{4\pi}\Sigma_{s,l}^{g\leftarrow g'} \sum_{k'=1}^{M} \omega_{k'} P_l(\mathbf{\Omega}_k \cdot \mathbf{\Omega}_{k'})\Phi_{k'}^{g'}(\mathbf{r}) + Q_k^g(\mathbf{r}) \tag{3.25}$$

avec

$$Q_k^g(\mathbf{r}) = \begin{cases} Q_{fixe,k}^g(\mathbf{r}) & \texttt{pour un problème à source fixe} \\ \frac{\chi^g}{4\pi K_{eff}} \sum\limits_{g'=1}^{G} \nu\Sigma_f^{g'}\Phi^{g'}(\mathbf{r}) & \texttt{pour un problème à fission} \end{cases} \tag{3.26}$$

La résolution porte non pas directement sur cette équation, mais sur sa formulation faible. Cela consiste à multiplier cette équation par une fonction test $v \in L_2(V)$ et l'intégrer par

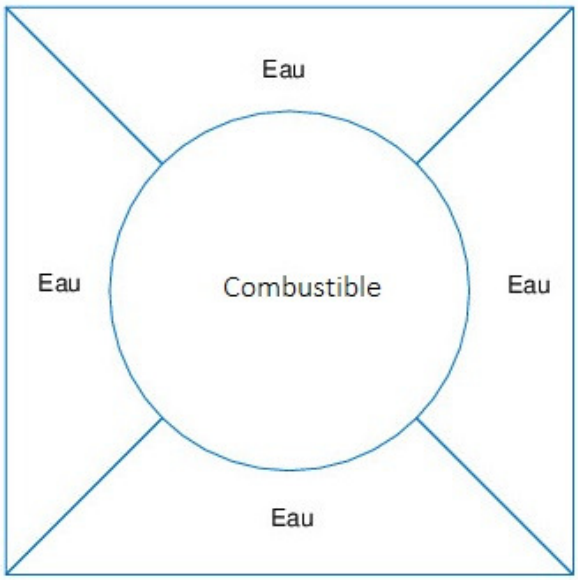


Figure 3.9 Exemple de la géométrie Pincell divisée en 5 patchs

rapport à $\mathbf{r}$ sur le patch V :

$$\int_V d\mathbf{r}\ v(\mathbf{r})\mathbf{\Omega}_k \cdot \nabla\Phi_k^g(\mathbf{r}) + \Sigma^g \int_V d\mathbf{r}\ v(\mathbf{r})\Phi_k^g(\mathbf{r}) = \sum_{g'=1}^{G}\sum_{l=0}^{L}\frac{2l+1}{4\pi}\Sigma_{s,l}^{g\leftarrow g'}\sum_{k'=1}^{M}\omega_{k'}P_l(\mathbf{\Omega}_k\cdot\mathbf{\Omega}_{k'})\int_V d\mathbf{r}\ v(\mathbf{r})\Phi_{k'}^{g'}(\mathbf{r}) + \int_V d\mathbf{r}\ v(\mathbf{r})Q_k^g(\mathbf{r}) \tag{3.27}$$

Il est évident que toute solution de l'équation (3.25) est aussi solution de l'équation (3.27). Mais pour que la solution de l'équation (3.27) puisse être aussi solution de l'équation (3.25), il faut qu'elle soit vérifiée pour toute fonction intégrable sur V. En pratique, elle est vérifiée pour un nombre limité de fonctions test, d'où son appellation formulation faible.

En appliquant le théorème de la divergence à $\int_V d\mathbf{r}v(\mathbf{r})\mathbf{\Omega}_k \cdot \nabla\Phi_k^g(\mathbf{r})$ et en notant l'intégrale volumique $\int_V d\mathbf{r}f(\mathbf{r})g(\mathbf{r}) = (f,g)$, l'intégrale surfacique $\int_{dV} d\mathbf{r}f(\mathbf{r})g(\mathbf{r}) =< f,g >$, l'équation (3.27) devient :

$$-\mathbf{\Omega}_k\cdot(\Phi_k^g,\nabla v)+\Sigma^g(\Phi_k^g,v)+\mathbf{\Omega}_k\cdot<\Phi_k^g,v\mathbf{n}> = \sum_{g'=1}^{G}\sum_{l=0}^{L}\frac{2l+1}{4\pi}\Sigma_{s,l}^{g\leftarrow g'}\sum_{k'=1}^{M}\omega_{k'}P_l(\mathbf{\Omega}_k\cdot\mathbf{\Omega}_{k'})(\Phi_{k'}^{g'},v)+(Q_k^g,v) \tag{3.28}$$

Où $\mathbf{n}$ est la normale à la surface.

Pour l'instant, l'équation est résolue sur chaque patch indépendamment des autres patches. Il faut que l'information puisse être transmise d'un patch à l'autre. Un terme d'upwinding est introduit [2, 5–7]. Pour cela, le terme $\mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >$ est divisé en deux parties, une partie sur la surface entrante du patch pour la direction donnée et l'autre sur la surface sortante :

$$\mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >= \mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >_+ + \mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >_- \tag{3.29}$$

Où $\mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >_+$ est l'intégrale sur les surfaces sortantes telle que $\mathbf{\Omega}_k \cdot \mathbf{n} > 0$ et $\mathbf{\Omega}_k \cdot < \Phi_k^g, v\mathbf{n} >_-$ est celle sur la surface entrante telle que $\mathbf{\Omega}_k \cdot \mathbf{n} < 0$.
C'est $< \Phi_k^g, v\mathbf{n} >_-$ qui est remplacé par un terme d'upwinding $< \Phi_{adj}, v\mathbf{n} >_-$ (cf fig. 3.10). Le flux adjacent Φ_{adj} intervenant peut être défini de plusieurs façons selon la situation du patch. Il est égal au flux angulaire du patch voisin lorsque la surface entrante en question est une interface entre les deux patches. La direction angulaire est la même, parce que ce sont les neutrons sortants du patch voisin qui entrent avec la même direction angulaire dans le patch. Le flux adjacent Φ_{adj} est imposé par les conditions limites lorsque la surface est sur le bord de l'espace. Il est égal à 0 si la surface est soumise à une condition aux limites vide, donc pas de neutrons entrants. Si une condition aux limites réflective est imposée à la surface, Φ_{adj} est égal au flux angulaire du même patch, mais provenant d'une direction angulaire telle que les neutrons entrants soient ceux réfléchis par la surface (la surface est donc sortante pour cette direction en question).

$$\Phi_{adj}(\mathbf{r}) = \begin{cases} \Phi^g_{k,patch\ voisin}(\mathbf{r}) & \texttt{si } \mathbf{r} \in \partial V_{interface} \\ 0 & \texttt{si } \mathbf{r} \in \partial V_{vide} \\ \Phi^g_{k'}(\mathbf{r}) & \texttt{si } \mathbf{r} \in \partial V_{refl} \texttt{ où } \mathbf{\Omega}_{k'} = \mathbf{\Omega}_k - 2(\mathbf{\Omega}_{k'} \cdot \mathbf{n})\mathbf{n} \end{cases} \tag{3.30}$$

Remarque :
Grâce au caractère Level-Symétric, $\mathbf{\Omega}_{k'}$ est bien une ordonnée disponible, si la surface en question est parallèle aux axes cartésiennes.

Quant à la résolution de l'équation, c'est à cette étape que les NURBS interviennent. En effet, les fonctions test sont celles de la base utilisée pour construire la géométrie du patch en question. Et la solution recherchée Φ_k^g est aussi décomposée sur cette même base. C'est pour cette raison que la méthode IGA est qualifiée d'isogéométrique. Par abus de notation, on notera $R_i(\mathbf{r}) = R_{i,p} \circ \mathcal{F}^{-1}(\mathbf{r})$:

$$v(\mathbf{r}) = R_i(\mathbf{r}) \text{ où } i \in [\![0; n]\!] \tag{3.31}$$

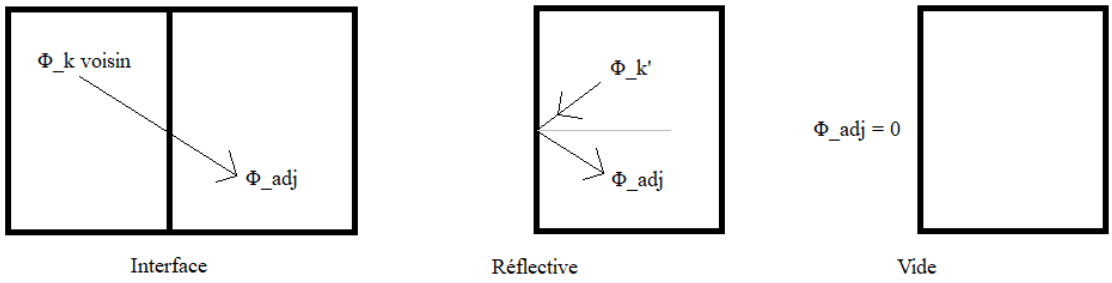


Figure 3.10 Illustration des 3 cas possibles pour le flux adjacent, de gauche à droite, respectivement interface, réflective et vide

$$\Phi_k^g(\mathbf{r}) = \sum_{j=0}^{n} \Phi_k^{g,j} R_j(\mathbf{r}) \tag{3.32}$$

Où les $\Phi_k^{g,j}$ sont les dégrées de liberté qui sont recherchés.
L'équation doit être vérifiée pour les $n+1$ fonctions test, un système matriciel est donc obtenu. Pour une géométrie 2D, un groupe g, une ordonnée $\mathbf{\Omega}_k = (\mu_k, \eta_k, \xi_k)$, sur un patch, le système matriciel s'écrit :

$$\begin{aligned}\left(-\mu_k \underline{\underline{S_x}} - \eta_k \underline{\underline{S_y}} + \Sigma^g \underline{\underline{M}} + \underline{\underline{P_k}}\right) \mathbf{\Phi}_k^g = \sum_{g'=1}^{G} \sum_{l=0}^{L} \frac{2l+1}{4\pi} \Sigma_{s,l}^{g \leftarrow g'} \sum_{k'=1}^{M} \omega_{k'} P_l(\mathbf{\Omega}_k \cdot \mathbf{\Omega}_{k'}) \underline{\underline{M}} \mathbf{\Phi}_{k'}^{g'} \\ + \mathbf{Q}_k^g - \mathbf{\Phi}_{adj}\end{aligned} \tag{3.33}$$

avec

$$\mathbf{\Phi}_k^g = \begin{bmatrix} \Phi_k^{g,0} \\ \dots \\ \Phi_k^{g,j} \\ \dots \\ \Phi_k^{g,n} \end{bmatrix} \quad \mathbf{\Phi}^g = \begin{bmatrix} \sum_{k=1}^{M} \omega_k \Phi_k^{g,0} \\ \dots \\ \sum_{k=1}^{M} \omega_k \Phi_k^{g,j} \\ \dots \\ \sum_{k=1}^{M} \omega_k \Phi_k^{g,n} \end{bmatrix} \tag{3.34}$$

et

$$\mathbf{Q}_{k_{\{i\}}}^g = \begin{cases} \left(Q_{fixe,k}^g, R_i\right) & \texttt{source fixe} \\ \frac{\chi^g}{4\pi K_{eff}} \sum\limits_{g'=1}^{G} \nu \Sigma_f^{g'} \underline{\underline{M}} \mathbf{\Phi}^{\mathbf{g'}} & \texttt{fission} \end{cases} \tag{3.35}$$

$$\mathbf{\Phi}_{adj_{\{i\}}} = \langle \Phi_{adj}, R_i \mathbf{\Omega}_k \cdot \mathbf{n} \rangle_- \tag{3.36a}$$

$$\underline{\underline{S_x}}_{\{i,j\}} = \left(\frac{\partial R_i}{\partial x}, R_j \right) \tag{3.36b}$$

$$\underline{\underline{S_y}}_{\{i,j\}} = \left(\frac{\partial R_i}{\partial y}, R_j \right) \tag{3.36c}$$

$$\underline{\underline{M}}_{\{i,j\}} = (R_i, R_j) \tag{3.36d}$$

$$\underline{\underline{P_k}}_{\{i,j\}} = \langle R_j, R_i \mathbf{\Omega}_k \cdot \mathbf{n} \rangle_+ \tag{3.36e}$$

Pour le calcul des intégrales spatiales, une règle de quadrature de Gauss est utilisée. La quadrature est calculée par GeoPDE de sorte qu'il y ait une distribution uniforme de points de quadratures sur chaque élément(sous-division d'un patch, cf 3.1.3). La quadrature prend aussi en compte le degrée des NURBS pour que l'intégrale calculée soit suffisamment précise.

Pour les cas d'études qui vont intervenir dans la suite du mémoire, les systèmes matriciels correspondants sont les suivants.

Pour un problème à source 2D, monoénergétique, isotrope, il y a M équations par patch. Elles s'écrivent :

$$\left(-\mu_k \underline{\underline{S_x}} - \eta_k \underline{\underline{S_y}} + \Sigma \underline{\underline{M}} + \underline{\underline{P_k}} \right) \mathbf{\Phi}_k = \frac{1}{4\pi} \Sigma_s \underline{\underline{M}} \mathbf{\Phi} + \mathbf{Q}_{fixe,k} - \mathbf{\Phi}_{adj} \tag{3.37}$$

Pour un problème à fission 2D, monoénergétique, isotrope, il y a M équations par patch. Elles s'écrivent :

$$\left(-\mu_k \underline{\underline{S_x}} - \eta_k \underline{\underline{S_y}} + \Sigma \underline{\underline{M}} + \underline{\underline{P_k}} \right) \mathbf{\Phi}_k = \frac{1}{4\pi} \Sigma_s \underline{\underline{M}} \mathbf{\Phi} + \frac{\chi}{4\pi K_{eff}} \nu \Sigma_f \underline{\underline{M}} \mathbf{\Phi} - \mathbf{\Phi}_{adj} \tag{3.38}$$

Pour un problème à fission 2D, multigroupe, anisotrope(d'ordre 1), il y a $M \times G$ équations par patch. Elles s'écrivent :

$$\begin{aligned} \left(-\mu_k \underline{\underline{S_x}} - \eta_k \underline{\underline{S_y}} + \Sigma^g \underline{\underline{M}} + \underline{\underline{P_k}} \right) \mathbf{\Phi}_k^g = & \sum_{g'=1}^{G} \sum_{k'=1}^{M} \frac{1}{4\pi} \left(\Sigma_{s,0}^{g \leftarrow g'} + 3\Sigma_{s,1}^{g \leftarrow g'} \mu_k \right) \omega_{k'} \underline{\underline{M}} \mathbf{\Phi}_{k'}^{g'} \\ & + \frac{\chi^g}{4\pi K_{eff}} \sum_{g'=1}^{G} \nu \Sigma_f^{g'} \underline{\underline{M}} \mathbf{\Phi}^{\mathbf{g'}} - \mathbf{\Phi}_{adj} \end{aligned} \tag{3.39}$$

Ici se termine la mise en équation de la BTE. La prochaine section présente les détails du développement de l'algorithme.

3.3 L'algorithme de la méthode IGA

Dans cette section, les grandes lignes de l'algorithme sont présentées. L'algorithme peut être divisé en 3 étapes majeures : conception et préparation de la géométrie, assemblage des matrices et résolution des équations. Dans un premier temps, cette section explicite les règles de conceptions de la géométrie, les choix effectués pour le raffinement et la construction de maillages/espaces de fonctions à partir de la géométrie. Puis, elle aborde les calculs effectués pour assembler les matrices, notamment les intégrales de surface qui présentent quelques subtilités. Enfin, cette section explique les méthodes de résolution des équations, comprenant la méthode de puissance pour calculer K_{eff} et la méthode de source itérée pour découpler les équations.
Les fonctions MATLAB avec le préfix "PF_" sont écrites pour le projet. Par conséquent, leurs arguments et leurs sorties seront précisés lors de leur première apparition. L'exemple de géométrie Pincell 2D (figure 3.11) sert d'illustration.

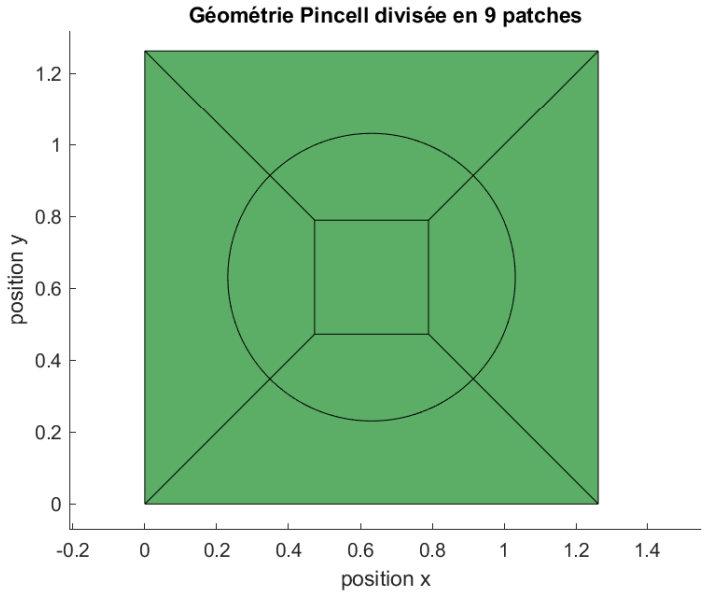


Figure 3.11 La géométrie Pincell divisée en 9 patches, chaque patch contenant qu'un élément

3.3.1 Conception de la géométrie, raffinement et constructions des maillages /espaces de fonctions

Conception de la géométrie L'algorithme impose quelques règles de conception pour la géométrie. Il faut noter que ces règles sont propres à l'algorithme conçu ici et non intrinsèquement requises pour la méthode IGA. Il pourrait être utile de rappeler qu'un patch 2D est

une surface NURBS définie par produit tensoriel (voir les équations (3.4) et (3.5)).

— Un patch est nécessairement délimité par 4 faces. En notant u et v les deux paramètres, GeoPDE adopte la convention suivante pour les numéroter :

Tableau 3.4 Numérotation conventionnelle des faces dans GeoPDE

Numéro de la face	Expression
1	$\{(0, v),\ v \in [0, 1]\}$
2	$\{(1, v),\ v \in [0, 1]\}$
3	$\{(u, 0),\ u \in [0, 1]\}$
4	$\{(u, 1),\ u \in [0, 1]\}$

— Un patch doit être orienté de sorte que la face 1 est à gauche de la face 2 et la face 3 est en dessous de la face 4.(cf 3.12) NURBS Toolbox dispose d'une fonction nrbreverse pour inverser l'orientation d'une direction paramétrique.

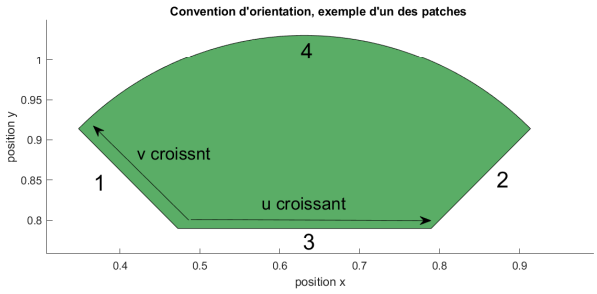


Figure 3.12 Convention d'orientation illustré par un des patches du Pincell

— Chaque face d'un patch ne peut pas être à l'interface avec plus d'un patch voisin ou être à la fois interface et frontière.

— Les degrés des NURBS de part et d'autre de l'interface doivent être égaux. Il est possible de modifier les degrés via la fonction nrbdegelve de GeoPDE. De même, le nombre de nœuds doit être égal.

Remarque :

Avec ces règles de conception en place, les patches résultants sont généralement assez réguliers pour ne pas présenter de points de singularité. Par ailleurs, les points de contrôle des patches voisins coïncident sur les interfaces (cf figure 3.13). La transmission d'information de patch en patch est facilitée.

La contrainte sur le degré et les nœuds n'est pas obligatoire pour une méthode discontinue comme la nôtre, il faudrait alors adapter l'algorithme pour calculer correctement l'information transmise. Mais il existe une méthode continue pour laquelle cette contrainte devient une contrainte forte [12].

Les fonctions de GeoPDE disposent d'une documentation dans MATLAB, et le package est en libre accès.

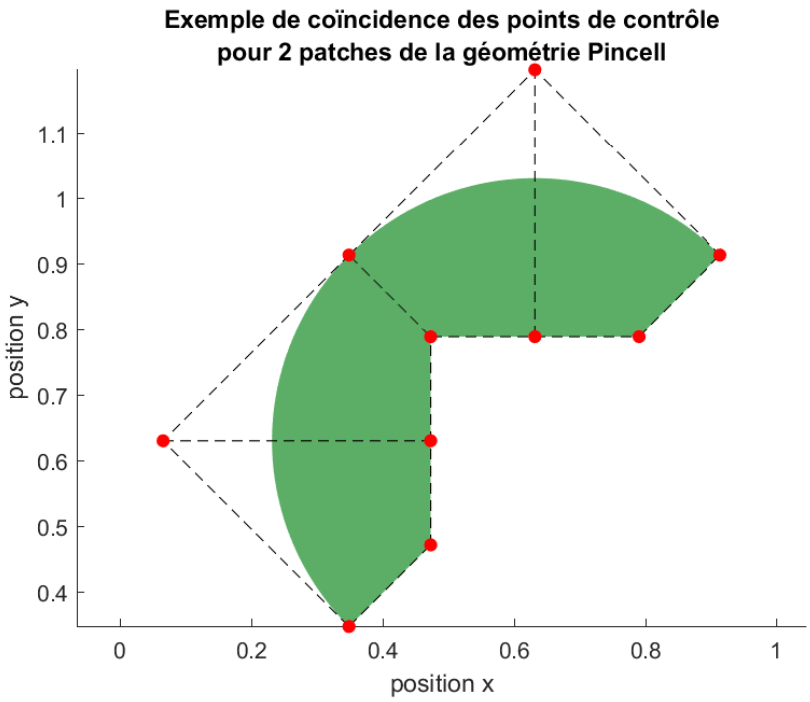


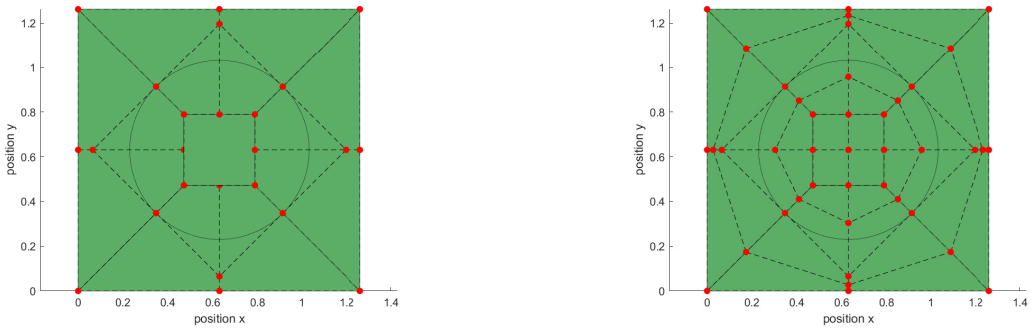
Figure 3.13 Exemple de deux patches de la géométrie Pincell avec les points de contrôle coïncidents à l'interface

Raffinement de la géométrie Avant de procéder au raffinement, les degrés des différents patches sont d'abord uniformisés afin qu'ils respectent la règle énoncée ci-dessus. Le but du mémoire étant de construire un prototype de la méthode IGA, une méthode brutale est appliquée. L'uniformisation consiste alors à prendre le degré le plus élevé de tous les patches dans les deux directions paramétriques et de l'imposer à chaque patch dans chacune des directions paramétriques.

Quant au raffinement, il en existe deux types : le raffinement h et le raffinement p (cf la sous-section 3.1.4). Pour la même raison de prototypage, les insertions de nœuds et les élévations de degré sont réalisées de manière uniforme sur toute la géométrie (c'est-à-dire identique pour les deux directions, dans chaque patch). Si des méthodes plus raffinées existent, cette approche grossière est suffisante lorsque la géométrie est hautement symétrique. L'ensemble uniformisation et raffinement est pris en charge par la fonction PF_knotins_degelev (cf Annexe E.1 pour le script). Le raffinement p précède systématiquement le raffinement h.

PF_knotins_degelev	
Inputs	
nrb	Array de structs NURBS Les patches NURBS à modifier
knot_ins	Entier >=0 Nombre de noeuds à insérer. Les nœuds insérés seront alors ceux uniformément répartis dans le segment $[0, 1]$
deg_elev	Entier >=0 Élévation souhaitée par rapport au degré actuel
Output	
nrb_m	Array de structs NURBS Les patches NURBS modifiées

La figure 3.14 montre l'effet de l'uniformisation. Avant uniformisation, il y a un problème de coïncidence des degrés entre le patch central et les 4 patches l'entourant. En effet, pour ces interfaces, il y a seulement 2 points de contrôle (donc des NURBS de degré 1, comme il n'y pas de nœud inséré) du côté du patch carré et 3 points de contrôle du côté de ses patches voisins (pas de nœuds insérés, NURBS de degré 2). Comme les points de contrôle sont superposés, la lecture n'est pas intuitive. Si le patch central avait 3 points de contrôle sur l'interface, il posséderait aussi toute une rangée de points supplémentaires à l'intérieur du patch puisqu'il est construit par produit tensoriel (comme c'est le cas de la figure 3.14b). En revanche, il faut souligner le caractère brutal de l'uniformisation. Il aurait été suffisant d'élever seulement les degrés du patch central à 2. Et les autres patches pouvaient garder leurs degrés $[2, 1]$ ou $[1, 2]$ comme leurs interfaces coïncident déjà.

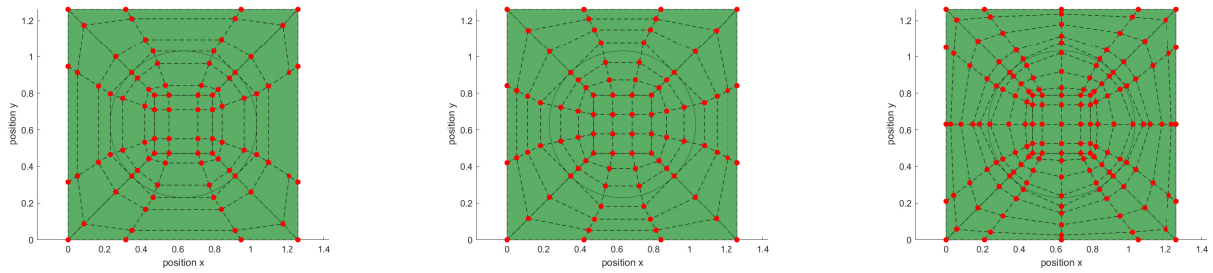


(a) Avant uniformisation

(b) Patches uniformisés

Figure 3.14 Comparaison des patches NURBS avec leurs points de contrôle avant et après uniformisation

Comme les raffinements sont uniformes, les patches restes uniformisés.



(a) raffinement h,{0.5} inséré (b) raffinement p, (+1,+1) (c) raffinement p-h

Figure 3.15 Effet des raffinements sur la géométrie

Remarque :

La discussion ici concerne uniquement la géométrie. En ce qui concerne les effets sur les fonctions NURBS, ils ont été discutés dans la sous-section 3.1.4. Néanmoins, une illustration des fonctions NURBS est fournie figures 3.16, 3.18 et 3.17 avec le patch de la figure 3.12. Ces fonctions sont représentées dans l'espace paramétrique :

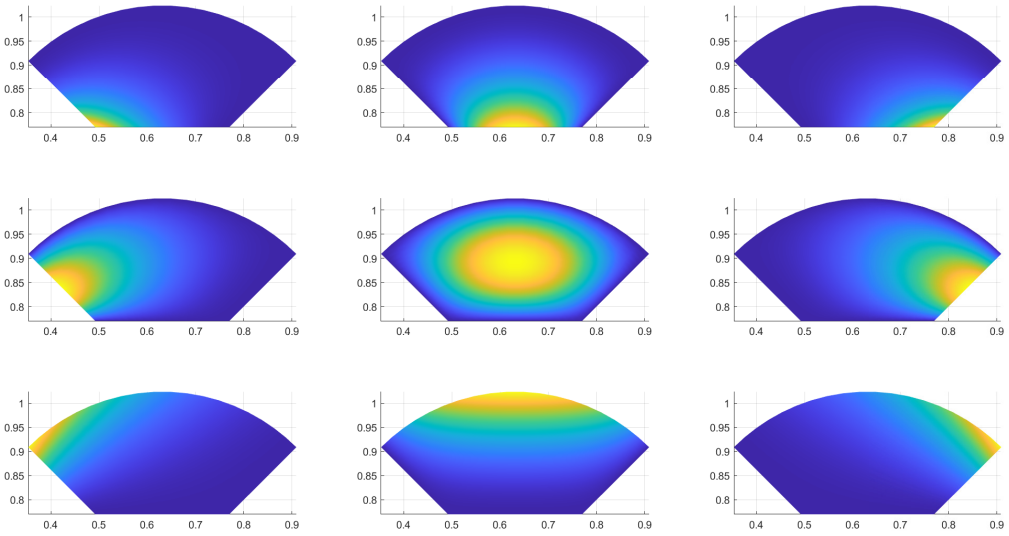


Figure 3.16 Les fonctions NURBS du patch après uniformisation

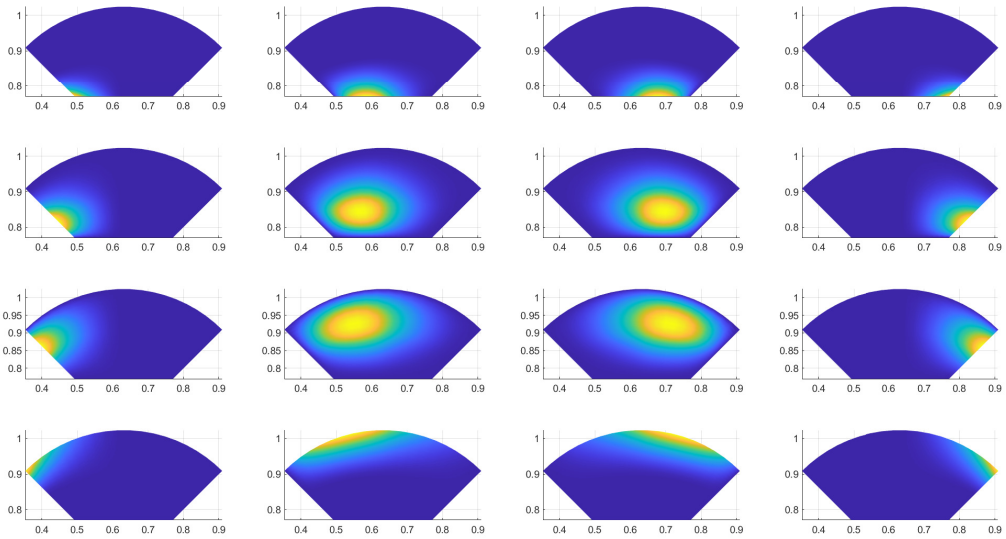


Figure 3.17 Les fonctions NURBS du patch après uniformisation et insertion du nœud {0.5} dans les deux directions paramétriques

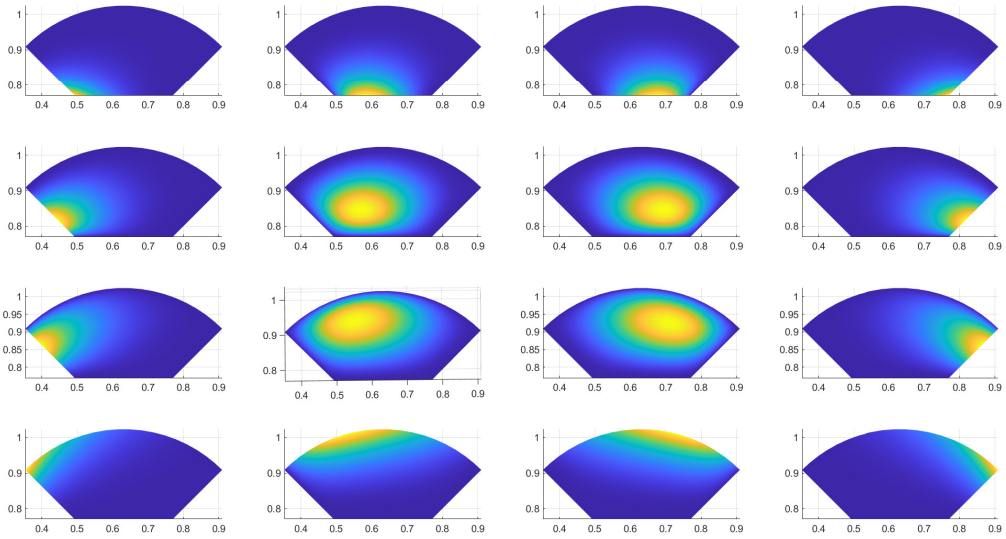


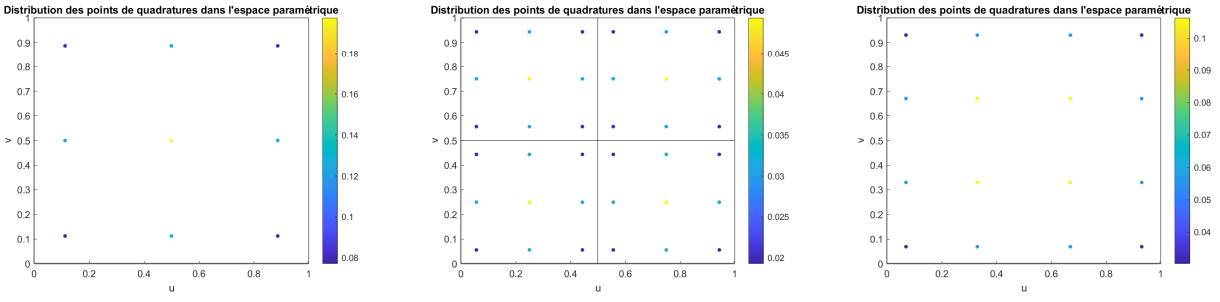
Figure 3.18 Les fonctions NURBS du patch après uniformisation et élévation de degré de 1 pour les deux directions paramétriques

Il est possible de remarquer au passage que les fonctions NURBS sont plus étalées spatialement à la suite d'un raffinement p figure (3.18) que celles obtenues à la suite d'un raffinement h (figure 3.17). Cela s'explique par le fait que le nombre d'éléments reste invariant pour le raffinement p. Par ailleurs, si le nœud inséré était, par exemple, {0.1}, le caractère densité locale pour le raffinement h versus densité globale pour le raffinement p serait plus prononcé.

Construction des maillages et des espaces de fonction NURBS Les fonctions MATLAB intervenant dans cette partie proviennent du package GeoPDE. Leur utilisation est bien documentée dans [12].

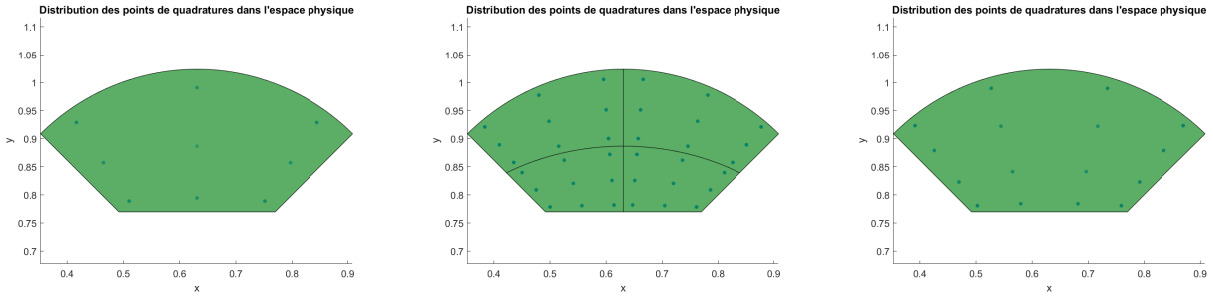
La construction d'un maillage à partir d'un patch consiste à calculer les points de quadrature en fonction des degrés des NURBS et de les appliquer sur les éléments en fonction du vecteur nodal. Un maillage constitue un quadrillage pour l'évaluation des NURBS et le calcul des intégrales. Le syntaxe utilisé est :
[qn,qw]=msh_set_quad_nodes (nrb.knots, msh_gauss_nodes(nrb.order)) ;
Où [qn, qw] est la quadrature, "nrb" est la `struct` MATLAB représéntant le patch ; les champs "knots" et "order" sont respectivement les vecteurs nodaux et les ordres du patch (ordre = degré+1).

Les informations sont regroupées dans un constructeur MATLAB de classe `msh_cartesian`, via la fonction msh_cartesian(nrb.knots,qn,qw,geo_load(nrb)). Le constructeur ne contient que les moyens de calculs et les données nécessaires mais ne contient pas de résultats de calcul. Pour avoir accès à certaines grandeurs comme le Jacobien ou la distribution des points de quadrature sur les éléments, il faut encore faire appel à msh_precompute(msh) qui réalise les calculs et renvoie les résultats sous format `struct`. Une illustration de maillage est fournie figures 3.19 et 3.20 pour un patch de la géométrie Pincell.



(a) après uniformisation (b) raffinement h, {0.5 } inséré (c) raffinement p, (+1, +1)

Figure 3.19 Les maillages du patch de la fig. 3.16, représentés dans l'espace paramétrique



(a) après uniformisation (b) raffinement h, {0.5 } inséré (c) raffinement p, (+1, +1)

Figure 3.20 Les maillages du patch de la fig. 3.16, représentation dans l'espace physique

La distribution des points de quadrature ainsi que la répartition des poids sont effectivement homogènes d'un élément à un autre (fig 3.19b). Il y a p+1 points de quadrature par élément, et sachant qu'une fonction NURBS peut être non nulle sur au plus $(p+1)^2$ éléments (généralisation de la propriété "localité" à 2D), le nombre de points de quadrature pour une fonction NURBS 2D donnée est majorée par $(p+1)^3$. Il est préférable d'avoir au moins p+1 éléments par direction paramétrique, pour permettre plus de finesse dans la variation de la solution numérique. Cela correspond à l'insertion de p nœuds dans le vecteur nodal.

La construction des maillages est terminée. L'évaluation des fonctions NURBS a lieu sur ces maillages. La logique est similaire à celle de la construction des maillages, le constructeur et les résultats sont séparés. Le constructeur de l'espace des fonctions est généré via la fonction sp_nurbs(nrb,msh) ; où "msh" est le constructeur `msh_cartesian` construit plutôt. Les fonctions NURBS peuvent être évaluées sur chaque point de quadrature avec sp_precompute(sp, msh), sp étant le constructeur `sp_scalar` retourné par sp_nurbs. Plusieurs grandeurs peuvent être calculées. En ce qui concerne le mémoire, ce sont les fonctions NURBS et leurs gradients.

Le même processus est appliqué à tous les patches. Il ne reste plus qu'à préparer les informations de surface avant de procéder à l'assemblage des matrices. Pour ce faire, la syntaxe utilisée est :[interface, boundaries]=nrbmultipatch(nrb) ; où "interface" indique les faces des patches qui sont en interfaces, et "boundaries" indique celles qui sont sur la frontière. C'est cette fonction de GeoPDE qui requiert la coïncidence des degrés et le respect de l'orientation. Et finalement, plus par soucis de commodité que de nécessité, les constructeurs sont rassemblés dans des constructeurs multipatches `msh_multipatch` et `sp_multipatch` via msh_multipatch(msh,boundaries) et sp_multipatch(sp,mp_msh,interface) respectivement ; (où msh et sp sont désormais des `cell` contenant tous les "msh" et "sp"). Ces constructeurs seron notés mp_msh et mp_sp par la suite.

3.3.2 Assemblage des matrices

Les matrices à assembler sont celles de l'équation (3.33). Elles se divisent en deux sortes, celles contenant les intégrales de volume : $\underline{\underline{S_x}}$, $\underline{\underline{S_y}}$ et $\underline{\underline{M}}$; et celles avec des intégrales de surfaces : $\underline{\underline{P_k}}$ et une matrice calculant $\mathbf{\Phi}_{adj}$.

La matrice $\underline{\underline{M}}$ se calcule comme suit pour un patch :

$$\begin{aligned}
\underline{\underline{M}}_{\{i,j\}} &= \int_V d\mathbf{r} R_i(\mathbf{r}) R_j(\mathbf{r}) \\
&= \int_{\hat{V}} d\hat{\mathbf{r}} R_{i,p}(u,v) R_{j,p}(u,v) \parallel \underline{\underline{J}}(u,v) \parallel \\
&= \sum_{elmt} \int_{\hat{V}_{elmt}} d\hat{\mathbf{r}} R_{i,p}(u,v) R_{j,p}(u,v) \parallel \underline{\underline{J}}(u,v) \parallel \\
&= \sum_{elmt} \sum_{q_p} W_{q_p} R_{i,p}(q_p) R_{j,p}(q_p) \parallel \underline{\underline{J}}(q_p) \parallel
\end{aligned} \tag{3.40}$$

Où q_p sont les points de quadrature, W_{q_p} les poids associés et $\parallel \underline{\underline{J}} \parallel$ le Jacobien du push-forward.

Les matrices $\underline{\underline{S_x}}$ et $\underline{\underline{S_y}}$ se calculent de la même manière que la matrice $\underline{\underline{M}}_{\{i,j\}}$:

$$\begin{aligned}
\underline{\underline{S_x}}_{\{i,j\}} &= \int_V d\mathbf{r} \frac{\partial R_i}{\partial x}(\mathbf{r}) R_j(\mathbf{r}) \\
&= \int_{\hat{V}} d\hat{\mathbf{r}} \frac{\partial R_{i,p}}{\partial x}(u,v) R_{j,p}(u,v) \parallel \underline{\underline{J}}(u,v) \parallel \\
&= \sum_{elmt} \int_{\hat{V}_{elmt}} d\hat{\mathbf{r}} \frac{\partial R_{i,p}}{\partial x}(u,v) R_{j,p}(u,v) \parallel \underline{\underline{J}}(u,v) \parallel \\
&= \sum_{elmt} \sum_{q_p} W_{q_p} \frac{\partial R_{i,p}}{\partial x}(q_p) R_{j,p}(q_p) \parallel \underline{\underline{J}}(q_p) \parallel
\end{aligned} \tag{3.41}$$

Ces calculs sont pris en charge par la fonction MATLAB PF_Matrix_v2 (cf Annexe E.2 pour le script), et ils sont réalisés pour tous les patches avec un seul appel de la fonction.

PF_Matrix_v2	
Inputs	
mp_sp	sp_multipatch Constructeur multipatches pour les espaces de fonctions
mp_msh	msh_multipatch Constructeur multipatches pour les maillages
Outputs	
M	$n_{patch} \times 1$ cell Dans chaque cell : $ndof \times ndof$ array de double L'ensemble des matrices $\underline{\underline{M}}$ pour chaque patch
S_1	$n_{patch} \times 1$ cell Dans chaque cell : $ndof \times ndof$ array de double L'ensemble des matrices $\underline{\underline{S_x}}$ pour chaque patch
S_2	$n_{patch} \times 1$ cell Dans chaque cell : $ndof \times ndof$ array de double L'ensemble des matrices $\underline{\underline{S_y}}$ pour chaque patch

Où n_{patch} est le nombre de patches et $ndof$ est le degré de liberté sur ce patch.
La logique de cette fonction est la suivante, elle s'appliquera à l'ensemble des fonctions d'assemblage des matrices. Pour un patch donné, l'ensemble des éléments est parcouru. Pour un élément donné, l'ensemble des fonctions non nulles sur cet élément est indexé avec une numérotation locale. La sommation est réalisée sur l'ensemble des points de quadrature de l'élément pour ces fonctions non nulles. Les résultats sont ensuite correctement enregistrés grâce au champ "connectivity" de l'espace des fonctions qui fait le lien entre la numérotation à l'échelle de l'élément et celle à l'échelle du patch. Le calcul est répété pour le patch suivant.

Le Jacobien $\| \underline{\underline{J}} \|$ est calculé par msh_precompute. L'équation donnant le Jacobien est :

$$\| \underline{\underline{J}}(u,v) \| = det \left(\begin{bmatrix} \frac{\partial x}{\partial u}(u,v) & \frac{\partial x}{\partial v}(u,v) \\ \frac{\partial y}{\partial u}(u,v) & \frac{\partial y}{\partial v}(u,v) \end{bmatrix} \right); \qquad \mathbf{S}(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \end{pmatrix} \tag{3.42}$$

Les fonctions $R_{i,p}$ et leurs gradients $\frac{\partial R_{i,p}}{\partial x}$ sont calculés par sp_precompute. Les algorithmes d'évaluation utilisés par GeoPDE sont similaires à ceux décrits dans [3]. Pour calculer $\frac{\partial R_i}{\partial x}$, il existe une formule pour la dérivée des fonctions B-splines, explicitée ci-dessous en 1D :

$$N'_{i,p}(u) = \frac{p}{u_{i+p} - u_i} N_{i,p-1}(u) - \frac{p}{u_{i+p+1} - ui + 1} N_{i+1,p-1}(u) \tag{3.43}$$

La dérivée des fonctions NURBS revient alors à utiliser l'équation (3.43), en appliquant la formule de dérivation pour un rapport de fonctions. Puis le calcul du gradient utilise la formule :

$$\nabla_{\mathbf{x},\mathbf{y}} = \underline{\underline{J}}^{-T} \nabla_{\mathbf{u},\mathbf{v}} \tag{3.44}$$

Intégrales de surface La différence fondamentale de la méthode IGA avec la FEM classique est que les faces peuvent désormais être complexes. Le vecteur normal n'est pas constant, et il est très probable d'avoir une face à la fois entrante et sortante pour une direction donnée. L'idée de face entrante/sortante peut être généralisée. L'intégrale sur les faces entrantes devient l'intégrale sur l'ensemble des points de la frontière où $\mathbf{\Omega} \cdot \mathbf{n} < 0$ (de même pour les faces sortantes). L'intégrale surfacique pour une fonction f, par exemple sur les faces entrantes, s'écrit donc [5, 6] :

$$\int_{dV^-} d\mathbf{r}\ f(\mathbf{r}) \mathbf{\Omega} \cdot \mathbf{n} = \int_{dV} d\mathbf{r}\ \tilde{f}(\mathbf{r}) \mathbf{\Omega} \cdot \mathbf{n} \quad ; \quad \tilde{f}(\mathbf{r}) = \begin{cases} f(\mathbf{r}) & \texttt{si} \quad \mathbf{\Omega} \cdot \mathbf{n} < 0 \\ 0 & \texttt{sinon} \end{cases} \tag{3.45}$$

La matrice $\underline{\underline{P_k}}$ se calcule comme suit :

$$\begin{aligned} \underline{\underline{P}}_{k_{\{i,j\}}} &= \int_{dV^+} d\mathbf{r}\ \mathbf{\Omega_k} \cdot \mathbf{n}(\mathbf{r})\ R_i(\mathbf{r}) R_j(\mathbf{r}) \\ &= \int_{d\hat{V}^+} d\hat{\mathbf{r}}\ \mathbf{\Omega_k} \cdot \mathbf{n}(u,v)\ R_{i,p}(u,v) R_{j,p}(u,v) \parallel \underline{\underline{J}}(u,v) \parallel \\ &= \sum_{face} \sum_{elmt} \sum_{{q_p}^+} W_{q_p}\ \mathbf{\Omega_k} \cdot \mathbf{n}(q_p)\ R_{i,p}(q_p) R_{j,p}(q_p) \parallel \underline{\underline{J}}(q_p) \parallel \end{aligned} \tag{3.46}$$

Où les ${q_p}^+$ sont les points de quadrature avec $\mathbf{\Omega_k} \cdot \mathbf{n}(q_p) > 0$. Les points de quadrature avec $\mathbf{\Omega_k} \cdot \mathbf{n}(q_p) < 0$ sont simplement ignorés. Ce calcul est pris en charge par la fonction MATLAB PF_MatrixP_v3 (cf Annexe E.3), réalisé sur tous les patches, pour tous les angles.

PF_MatrixP_v3	
Inputs	
mp_sp	sp_multipatch Constructeur multipatches pour les espaces de fonctions. Il ne sert ici qu'à donner le ndof pour chaque patch
bnds_s	$n_{patch} \times 4$ cell Dans chaque cell : struct Contient les espaces de fonctions ("pre_computed") pour les 4 faces de chaque patch.
bnds_m	$n_{patch} \times 4$ cell Dans chaque cell : struct Contient les maillages ("pre_computed") pour les 4 faces de chaque patch.
OMEGA	$1 \times n_{angle}$ cell Dans chaque cell : 1×4 array de double l'ensemble des angles utilisés au format $[\mu, \eta, \xi, \omega]$
Outputs	
Pk	$n_{patch} \times n_{angle}$ cell Dans chaque cell : $ndof \times ndof$ array de double L'ensemble des matrices $\underline{\underline{P_k}}$ pour chaque patch, chaque angle

Où n_{angle} est le nombre d'angles. Les angles sont pris depuis DRAGON et s'obtiennent avec PF_SN qui ne prend en argument que les nombres 4, 8 et 16. Elle ne contient aucun calcul, sinon juste un renvoi de données stockées. `bnds_s` et `bnds_m` sont calculés au préalable, car ils sont partagés avec le calcul de $\mathbf{\Phi}_{adj}$. Les fonctions msh_eval_boundary_side et sp_eval_boundary_side sont utilisées à cette fin (toujours du package GeoPDE).

Le vecteur $\mathbf{\Phi}_{adj}$ C'est le calcul le plus subtile du mémoire, il va être très détaillé dans ce paragraphe. On part de l'équation initiale :

$$\mathbf{\Phi}_{adj_{\{i\}}} = \int_{dV^-} d\mathbf{r}\ \mathbf{\Omega_k} \cdot \mathbf{n}(\mathbf{r})\ \Phi_{adj}(\mathbf{r}) R_i(\mathbf{r}) \tag{3.47}$$

La définition de Φ_{adj} est donnée plus tôt par l'équation (3.30). On réécrit l'équation (3.47) avec la décomposition de Φ_{adj}, :

$$\mathbf{\Phi}_{adj_{\{i\}}} = \int_{dV^-} d\mathbf{r}\ \mathbf{\Omega_k} \cdot \mathbf{n}(\mathbf{r})\ \sum_l \Phi_{adj}^{(l)} R_l(\mathbf{r}) R_i(\mathbf{r}) \tag{3.48}$$

Les coefficients de décomposition $\Phi^{(l)}_{adj}$ sont une écriture flexible que l'on adaptera en fonction des situations énoncées dans l'équation (3.30). Comme ces coefficients changent en fonction de la face, il faut alors calculer une matrice pour chaque face du patch. Après changement de variable et la subdivision de chaque face en ses éléments, on obtient la somme suivante :

$$\mathbf{\Phi}_{adj_{\{i\}}} = \sum_{face} \sum_{elmt} \sum_{q_p^-} \sum_{l} \mathbf{\Omega_k} \cdot \mathbf{n}(q_p)\ \Phi^{(l)}_{adj,face} R_{l,p}(q'_p) R_{i,p}(q_p) \parallel \underline{\underline{J}}(q_p) \parallel \tag{3.49}$$

Cette équation peut être réécrite sous forme matricielle :

$$\mathbf{\Phi}_{adj} = \sum_{face} \underline{\underline{M'_{face}}} \mathbf{\Phi_{adj,face}} \tag{3.50}$$

Où

$$\underline{\underline{M'_{face}}}_{\{i,l\}} = \sum_{elmt} \sum_{q_p^-} R_{i,p}(q_p) R_{l,p}(q'_p) \quad \text{et} \quad \mathbf{\Phi_{adj,face}} = \begin{bmatrix} \Phi^{(0)}_{adj,face} \\ \cdots \\ \Phi^{(l)}_{adj,face} \\ \cdots \\ \Phi^{(n)}_{adj,face} \end{bmatrix} \tag{3.51}$$

Subtilité :
Une nouvelle notation q'_p est introduite ici. Cette notation prend en compte le fait que l'indexation des points de quadrature pour un patch voisin peut être différente. En effet, pour la majorité des cas, $q'_p = q_p$ mais lorsqu'il s'agit d'une interface, $\Phi_{adj,face}$ est définie à partir du patch voisin. Il faut alors incorporer l'indexation du patch voisin pour les points de quadratures. La coïncidence des points de quadrature dans l'espace physique et dans l'espace paramétrique est assurée facilement par l'uniformisation imposée lors de la conception de la géométrie. Sans quoi, il faudra aussi prendre en compte le potentiel décalage des points de quadrature de part et d'autre de l'interface. Lorsque la direction paramétrique est la même de part et d'autre de l'interface, on a toujours $q'_p = q_p$, mais lorsque les directions paramétriques sont opposées, l'indexation de q'_p est décroissante lorsque celle de q_p croît :

$$q'_p = \begin{cases} q_p & \texttt{si}\ face\ \in\ \partial V_{refl} \\ q_p & \texttt{si}\ face\ \in\ \partial V_{interface} \text{ et même direction paramétrique} \\ q_{Nq_n - p + 1} & \texttt{si}\ face\ \in\ \partial V_{interface} \text{ et direction paramétrique opposée} \end{cases} \tag{3.52}$$

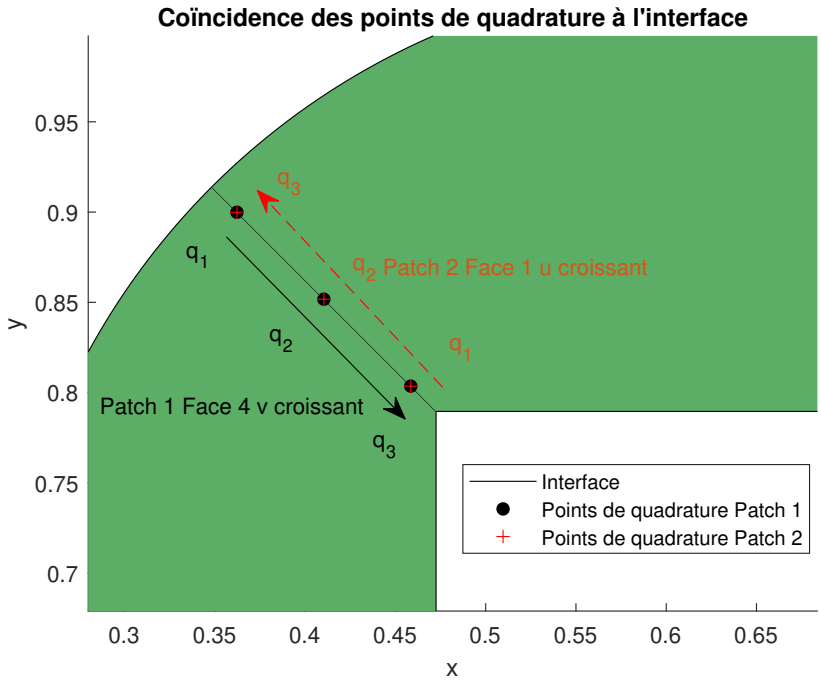


Figure 3.21 Un exemple d'interface où les points de quadrature sont superposés mais leurs indexation inversée

Remarque :
Les fonctions NURBS non nulles sur l'interface sont différentes de part et d'autre de l'interface. Il faudrait prendre la bonne connectivité et enregistrer les résultats dans la bonne position des matrices. Par exemple, pour la figure 3.21 les fonctions non nulles sont $R_{7,2}$,$R_{8,2}$ et $R_{9,2}$ pour le patch 1 face 4 et $R'_{1,2}$,$R'_{4,2}$ et $R'_{7,2}$ pour le patch 2 face 1.
Le calcul pour la matrice $\underline{\underline{P_k}}$ est similaire à ce calcul mais comme tout est calculé à partir d'un seul patch, le problème ne se pose pas.

Le calcul est pris en charge par la fonction PF__Phi_adj__ppp__v4 (cf Annexe E.4 pour le script). Une disjonction de cas est faite, donnant deux types de matrice selon que la face est une interface ou qu'elle est située sur la frontière.

`PF_Phi_adj_ppp_v4`	
`Inputs`	
`mp_sp`	`sp_multipatch`
	Constructeur multipatches pour les espaces de fonctions.
`mp_msh`	`msh_multipatch`
	Constructeur multipatches pour les maillages.
`boundaries`	$1 \times N_{boundaries}$ `struct`

	struct contenant les faces des patches qui sont sur la frontière : patches, faces.
`interface`	$1 \times N_{interface}$ `struct` struct contenant les interfaces : patch1, face1, patch2, face2, ornt. ornt indique si les directions paramétriques sont concourantes ou non
`OMEGA`	$1 \times n_{angle}$ `cell` Dans chaque cell : 1×4 `array de double` l'ensemble des angles utilisés au format $[\mu, \eta, \xi, \omega]$
`bnds_m`	$n_{patch} \times 4$ `cell` Dans chaque cell : `struct` Contient les maillages ("pre_computed") pour les 4 faces de chaque patch.
`bnds_s`	$n_{patch} \times 4$ `cell` Dans chaque cell : `struct` Contient les espaces de fonctions ("pre_computed") pour les 4 faces de chaque patch.
`bc`	`char` 'reflective' ou 'void' ; pour l'instant il n'y a pas de condition aux limites mixtes.
`Outputs`	
`Phi_bc`	$n_{patch} \times n_{angle} \times 4$ `cell` Dans chaque cell : $ndof \times ndof$ `array de double` L'ensemble des matrices $\underline{\underline{M'}}$ lorsque la face est sur la frontière. Une matrice par face par patch et par angle. La cellule est vide si la face correspondante n'est pas sur la frontière
`Phi_in`	$n_{patch} \times n_{angle} \times 4$ `cell` Dans chaque cell : $ndof \times ndof$ `array de double` L'ensemble des matrices $\underline{\underline{M'}}$ lorsque la face est une interface. Une matrice par face par patch et par angle. La cellule est vide si la face correspondante n'est pas une interface
`bnd_angle_sortant`	$n_{patch} \times n_{angle} \times 4$ `cell` Dans chaque cell : `entier ou vide`

	La cellule contient l'angle incident correspondant lorsque la face du patch présente une condition de réflexion pour l'angle. La cellule est vide si la face n'est pas sur la frontière
`in_voisin`	$n_{patch} \times n_{angle} \times 4$ `cell` Dans chaque cell : 1×3 `array ou vide` La cellule contient l'information sur le voisin : le patch, la face, l'orientation. (plus pratique à utiliser les informations organisées dans `interface`). La cellule est vide si la face n'est pas une interface

Les matrices Phi_in et Phi_bc sont les matrices $\underline{\underline{M'}}$ définies plus tôt. Pour une interface, il est vérifié que les neutrons perdus par un patch sur une face via le terme $\underline{\underline{P_k}}\mathbf{\Phi}_k^g$ sont bien récupérés par le patch voisin via le vecteur $\mathbf{\Phi_{adj}}$. Il n'y a donc pas de perte de neutrons au passage d'une interface.

3.3.3 Résolution de l'équation

Dans cette sous-section, nous allons résoudre l'équation (3.39) pour un problème à fission. Il s'agit d'une équation à valeur propre. Sa résolution consiste à trouver la valeur de K_{eff} et les vecteurs $\mathbf{\Phi}_k^g$ vérifiant l'équation. La méthode d'itération de puissance est utilisée pour calculer K_{eff}. Ceci constitue une première boucle d'itération, dite externe. Étant donné que le terme de diffusion couple les équations et fait intervenir l'inconnu $\mathbf{\Phi}_k^g$, nous avons besoin d'une deuxième itération pour les découpler. C'est la méthode de source itérée, et ceci constitue une seconde boucle dite interne.
Cette sous-section présente la méthode d'itération de puissance en premier, suivi de la méthode de source itérée et enfin, nous récapitulerons l'ensemble dans un schéma de calcul.

La méthode d'itération de puissance est classiquement utilisée pour résoudre un problème à fission. Elle est présentée dans [2, 5, 10]. L'objectif de cette méthode est de calculer K_{eff}.
En revenant à la BTE initiale (3.7), on peut réécrire l'équation avec des opérateurs linéaires :

$$\mathcal{L}\Phi(\mathbf{r}, E, \mathbf{\Omega}) = \mathcal{S}\Phi(\mathbf{r}, E, \mathbf{\Omega}) + \frac{1}{K_{eff}}\mathcal{F}\Phi(\mathbf{r}, E) \tag{3.53}$$

Hérédité :
Supposons que $\Phi^{(n)}$ et $K_{eff}^{(n)}$ sont obtenus à l'itération n de la boucle externe. La source de fission $\frac{1}{K_{eff}^{(n)}}\mathcal{F}\Phi^{(n)}$ est utilisée pour calculer $\Phi^{(n+1)}$ avec la boucle interne (cf paragraphe suivant sur la méthode source itérée). Nous obtenons alors :

$$\mathcal{L}\Phi^{(n+1)}(\mathbf{r},E,\mathbf{\Omega}) = \mathcal{S}\Phi^{(n+1)}(\mathbf{r},E,\mathbf{\Omega}) + \frac{1}{K_{eff}^{(n)}}\mathcal{F}\Phi^{(n)}(\mathbf{r},E) \tag{3.54}$$

L'intégration sur l'espace des phases de l'équation (3.54) et l'équation (3.53) pour $\Phi = \Phi^{(n+1)}$ donne :

$$\int d\mathbf{r}dEd\mathbf{\Omega}\ (\mathcal{L}-\mathcal{S})\Phi^{(n+1)}(\mathbf{r},E,\mathbf{\Omega}) = \int d\mathbf{r}dEd\mathbf{\Omega}\ \frac{1}{K_{eff}^{(n)}}\mathcal{F}\Phi^{(n)}(\mathbf{r},E) \tag{3.55}$$

$$\int d\mathbf{r}dEd\mathbf{\Omega}\ (\mathcal{L}-\mathcal{S})\Phi^{(n+1)}(\mathbf{r},E,\mathbf{\Omega}) = \int d\mathbf{r}dEd\mathbf{\Omega}\ \frac{1}{K_{eff}^{(n+1)}}\mathcal{F}\Phi^{(n+1)}(\mathbf{r},E) \tag{3.56}$$

L'égalité suivante s'en déduit :

$$K_{eff}^{(n+1)} = \frac{1}{K_{eff}^{(n)}}\frac{\int d\mathbf{r}dEd\mathbf{\Omega}\ \mathcal{F}\Phi^{(n+1)}(\mathbf{r},E)}{\int d\mathbf{r}dEd\mathbf{\Omega}\ \mathcal{F}\Phi^{(n)}(\mathbf{r},E)} \tag{3.57}$$

Ainsi la valeur de K_{eff} est mise à jour.
Critère de convergence :

$$\frac{|K_{eff}^{(n+1)} - K_{eff}^{(n)}|}{K_{eff}^{(n)}} \leq \epsilon_K \tag{3.58}$$

ϵ_K est souvent fixé à 10^{-5}.
Initialisation :
$K_{eff}^{(0)}$ prend généralement la valeur initiale de 1. Quant à $\Phi^{(0)}(\mathbf{r},E)$, un flux uniforme égal à 1 est choisi.

En terme de code,

$$\int d\mathbf{r}dEd\mathbf{\Omega}\ \mathcal{F}\Phi(\mathbf{r},E) = 4\pi \sum_{patch}\sum_{g'}\chi^{g'}\sum_{g''}\nu\Sigma_f^{g''}\int_V d\mathbf{r}\ \Phi^{g''}(\mathbf{r}) \tag{3.59}$$

Le calcul de $\int_V d\mathbf{r}\ \Phi^{g''}(\mathbf{r})$ se fait par un produit de vecteur ligne avec le vecteur colonne $\mathbf{\Phi}^{g''}$. Le vecteur ligne s'obtient avec la fonction MATLAB PF_vec.

PF_vec	
Inputs	
mp_sp	sp_multipatch Constructeur multipatches pour les espaces de fonctions.
mp_msh	msh_multipatch Constructeur multipatches pour les maillages.
MIX	$1 \times N_{MIX}$ array de struct Chaque struct représente un matériau et contient les champs NAME, TOTAL, NUSIGF, CHI, SCAT0 et SCAT1.
region_mix	$1 \times N_{patch}$ array d'entier L'entier m à la position i indique que le patch i utilise le m-ième matériau contenu dans MIX
Output	
vec	$n_{patch} \times 1$ cell Dans chaque cell : $1 \times ndof$ array de double vec$\{i_patch\}\mathbf{\Phi}^{\mathbf{g}''} = \int_{V_i} d\mathbf{r}\ \Phi^{g''}(\mathbf{r})$

La méthode de source itérée constitue la boucle interne. Supposons que nous sommes à l'itération (n) de la boucle externe, nous allons calculer $\Phi^{(n+1)}$ avec la méthode de source itérée [2].

Hérédité :
Supposons que nous sommes à l'itération (j) de la boucle interne, les $\mathbf{\Phi}_{\mathbf{k}}^{\mathbf{g},(\mathbf{j})}$ sont connus (pour ne pas alourdir les notations avec les doubles indices, l'indice (n) sera omis). Le but est de calculer $\mathbf{\Phi}_{\mathbf{k}}^{\mathbf{g},(\mathbf{j+1})}$ pour tous les patches, $\forall\ k, g$.

La résolution de l'équation (3.39) se fait par groupe d'énergie, dans l'ordre croissant de g (donc énergie décroissante). Les membres de gauche et la source de fission (calculée à partir de $\Phi^{(n)}$) sont déjà connus, il reste à préciser le terme de diffusion.

Le terme de diffusion peut être divisé en 3 parties, en fonction du groupe d'énergie des neutrons entrés en collision :

$$\mathbf{S_{up}} = \sum_{g'=g+1}^{G} \sum_{k'=1}^{M} \frac{1}{4\pi} \left(\Sigma_{s,0}^{g \leftarrow g'} + 3\Sigma_{s,1}^{g \leftarrow g'} \mu_k \right) \omega_{k'} \underline{\underline{M}} \mathbf{\Phi}_{k'}^{g',(n)} \tag{3.60}$$

C'est le terme dit de "up-scattering", il correspond à la prodcuction de neutrons diffusés d'énergie supérieure à celle des neutrons entrés en collision. En pratique, les sections efficaces

correspondantes sont nulles ou bien très petites. C'est pourquoi ce terme est calculé en remplaçant les $\mathbf{\Phi}_{k'}^{g'}$ par $\mathbf{\Phi}_{k'}^{g',(n)}$.

$$\mathbf{S_{down}} = \sum_{g'=1}^{g-1} \sum_{k'=1}^{M} \frac{1}{4\pi} \left(\Sigma_{s,0}^{g \leftarrow g'} + 3\Sigma_{s,1}^{g \leftarrow g'} \mu_k \right) \omega_{k'} \underline{\underline{M}} \mathbf{\Phi}_{k'}^{g',(j+1)} \tag{3.61}$$

C'est le terme dit de "down-scattering", il correspond à la production de neutrons diffusés d'énergie inférieure à celle des neutrons entrés en collision. Comme la résolution de l'équation se fait dans l'ordre croissant de g, les $\mathbf{\Phi}_{k'}^{g'(j+1)}$ ont déjà été calculés.

$$\mathbf{S_{self}} = \sum_{k'=1}^{M} \frac{1}{4\pi} \left(\Sigma_{s,0}^{g \leftarrow g} + 3\Sigma_{s,1}^{g \leftarrow g} \mu_k \right) \omega_{k'} \underline{\underline{M}} \mathbf{\Phi}_{k'}^{g,(j)} \tag{3.62}$$

C'est le terme dit de "self-scattering", il correspond à la production des neutrons du groupe g dans lui même. Il est calculé à partir des flux de l'itération (j).

Ainsi le terme de diffusion est bien défini, il est possible de calculer $\mathbf{\Phi}_k^{g,(j+1)}$. Pour un groupe g, un angle $\mathbf{\Omega}_k$ et un patch :

$$\mathbf{\Phi}_k^{g,(j+1)} = \underline{\underline{mat}} \setminus (\mathbf{RHS}); \tag{3.63}$$

Où on aurait posé :

$$\underline{\underline{mat}} = -\mu_k \underline{\underline{S_x}} - \eta_k \underline{\underline{S_y}} + \Sigma^g \underline{\underline{M}} + \underline{\underline{P_k}} \tag{3.64}$$

et

$$\mathbf{RHS} = \mathbf{S_{up}} + \mathbf{S_{self}} + \mathbf{S_{down}} + \frac{\chi^g}{4\pi K_{eff}} \sum_{g'=1}^{G} \nu\Sigma_f^{g'} \underline{\underline{M}} \mathbf{\Phi}^{\mathbf{g'},(\mathbf{n})} - \mathbf{\Phi}_{adj}^{(j+1)} \tag{3.65}$$

Il suffit alors de répéter les calculs pour l'ensemble des directions et pour tous les patches. Ensuite nous testons la convergence des flux angulaires $\mathbf{\Phi}_k^g$. Une fois les flux convergés, on a obtenu les flux $\mathbf{\Phi}_k^{g,(n+1)}$ et nous passons au groupe d'énergie $g+1$.

Critère de convergence :
La valeur seuil ϵ_Φ usuellement utilisée est 10^{-5}

$$\sum_{patch} \frac{\| \mathbf{\Phi}^{g,(j+1)} - \mathbf{\Phi}^{g,(j)} \|_2}{\| \mathbf{\Phi}^{g,(j)} \|_2} \leq \epsilon_\Phi \tag{3.66}$$

Où $\| \, . \, \|_2$ est la norme 2 canonique des vecteurs.
Initialisation :

$$\mathbf{\Phi}^{g,(j=0)} = \mathbf{\Phi}^{g,(n)} \tag{3.67}$$

Remarque :
L'ordre de balayage de l'espace des phases est important ici ! Comme l'équation a un terme de upwinding, l'ordre de balayage des patches dépend donc de la direction $\mathbf{\Omega_k}$. Donc il faudrait réaliser les calculs dans cet ordre : pour un groupe d'énergie donné, pour un angle donné et pour un patch donné...
L'ordonnance des patches est un problème en soi, elle est d'autant plus compliquée que les patches NURBS peuvent avoir une interface à la fois sortante et entrante. Une des méthodes consiste à prendre l'intégrale $\int_{dV} d\mathbf{r}\ \mathbf{\Omega} \cdot \mathbf{n}$ et d'en déduire si le voisin devrait plutôt recevoir de l'information ou bien en fournir. [6, 13]. Pour la réalisation de ce mémoire, les directions sont divisées en 4 quadrants et chaque quadrant a une ordonnance de patches.

Les fonctions MATLAB suivantes (cf Annexe E.5) calculent les termes de diffusion :

`PF_ComputeSCAT_up`	
`Inputs`	
`mp_sp`	`sp_multipatch` Constructeur multipatches pour les espaces de fonctions.
`MIX`	$1 \times N_{MIX}$ `array de struct` Chaque struct représente un matériau et contient les champs NAME, TOTAL, NUSIGF, CHI, SCAT0 et SCAT1.
`region_mix`	$1 \times N_{patch}$ `array d'entier` L'entier m à la position i indique que le patch i utilise le m-ième matériau contenu dans MIX
`OMEGA`	$1 \times n_{angle}$ `cell` Dans chaque cell : 1×4 `array de double` l'ensemble des angles utilisés au format $[\mu, \eta, \xi, \omega]$
`ANISO`	1 `ou` 2 Flag. La valeur 1 correspond à une diffusion isotrope et 2 à une diffusion anisotrope d'ordre 1.
`M`	$n_{patch} \times 1$ `cell` Dans chaque cell : $ndof \times ndof$ `array de double` L'ensemble des matrices $\underline{\underline{M}}$ pour chaque patch
`Phi`	$G \times n_{patch} \times n_{angle} cell$ Dans chaque cell : $ndof \times 1$ `array de double` Les flux angulaires $\Phi_k^{g,(n)}$
`Output`	

SCAT	$n_{groupe} \times n_{patch} \times n_{angle}$ cell Dans chaque cell : $ndof \times 1$ array de double Sources de diffusion "up-scattering"

Les fonctions PF_ComputeSCAT_self et PF_ComputeSCAT_down ne sont différentes que par les flux angulaires à donner en argument et les termes à calculer. Elles ne vont pas être développées. En revanche la source de fission est calculée par (cf Annexe E.6) :

PF_ComputeFIS	
Inputs	
mp_sp	sp_multipatch Constructeur multipatches pour les espaces de fonctions.
MIX	$1 \times N_{MIX}$ array de struct Chaque struct représente un matériau et contient les champs NAME, TOTAL, NUSIGF, CHI, SCAT0 et SCAT1.
region_mix	$1 \times N_{patch}$ array d'entier L'entier m à la position i indique que le patch i utilise le m-ième matériau contenu dans MIX
Keff	double Le facteur de multiplication effectif $K_{eff}^{(n)}$
M	$n_{patch} \times 1$ cell Dans chaque cell : $ndof \times ndof$ array de double L'ensemble des matrices $\underline{\underline{M}}$ pour chaque patch
PHI	$G \times n_{patch} cell$ Dans chaque cell : $ndof \times 1$ array de double Les flux intégrés $\Phi^{g,(n)}$
Output	
FIS	$n_{groupe} \times n_{patch}$ cell Dans chaque cell : $ndof \times 1$ array de double Sources de fission

Schéma de calcul Sur le diagramme de la figure 3.22, la boucle "for" pour l'énergie est représentée par un test "$g = G$?". Bien que le terme de diffusion "up-scattering" soit inclus dans la boucle de source itérée, il est de fait calculé à partir du flux de la boucle externe. La raison est que l'algorithme réalise les calculs dans l'ordre décroissant d'énergie. Si nous calculons le flux pour un groupe d'énergie g ; les flux angulaires d'énergie plus faible (donc

les groupes $g' > g$) ne sont pas encore calculés par l'itération interne. Un exemple de script est donné dans l'Annexe E.7.

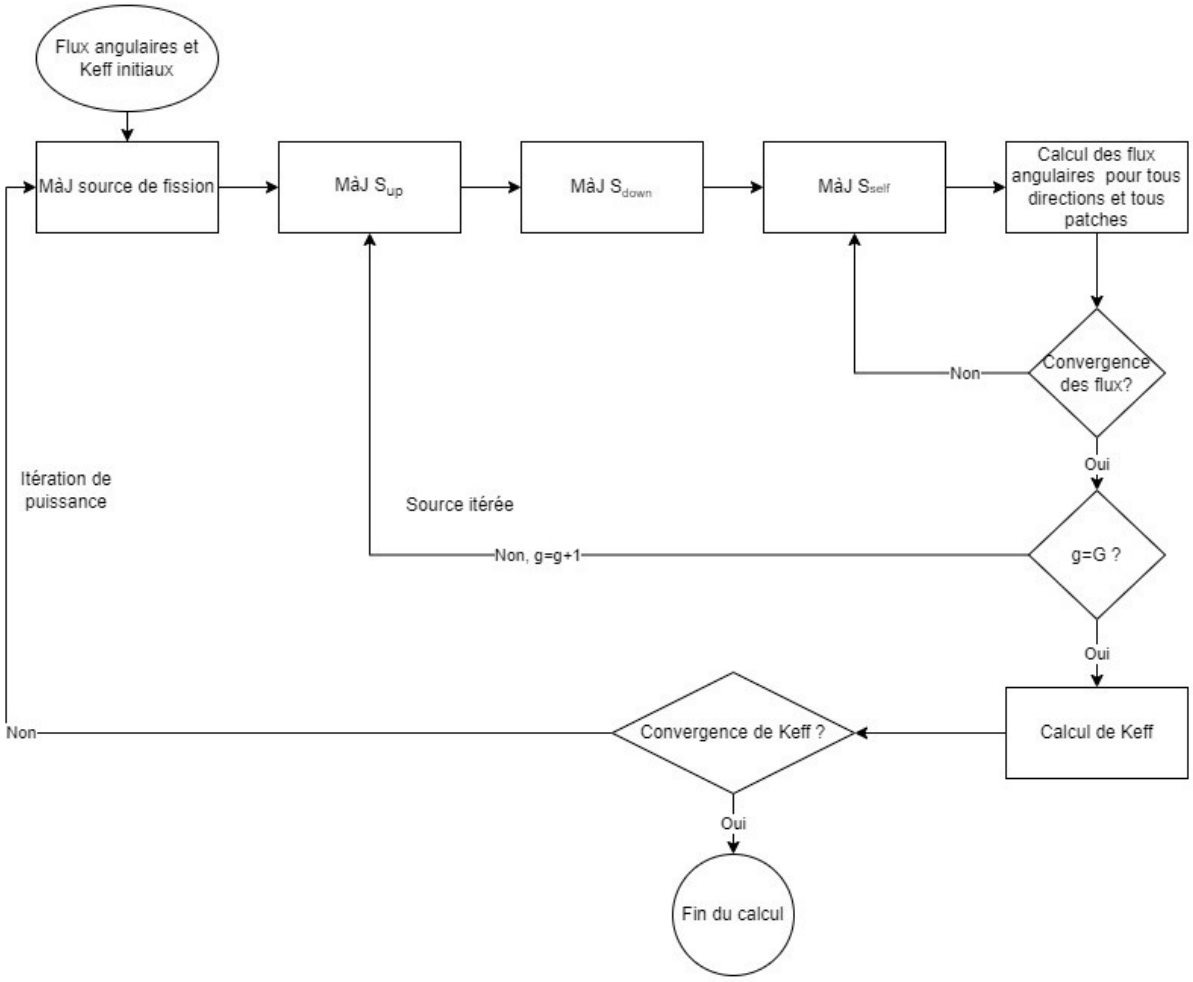


Figure 3.22 Schéma de calcul résumant la résolution de l'équation

CHAPITRE 4 RÉSULTATS

Ce chapitre présente les résultats obtenus avec l'algorithme développé.

Dans un premier temps, le bon assemblage des matrices est vérifié. Un patch du Pincell est choisi pour illustrer les calculs. Les matrices sont calculées à partir de formules théoriques, les intégrales sont prises en charge par la fonctionnalité de calcul symbolique de MATLAB. Les matrices obtenues sont confrontées à celles calculées par l'algorithme, qui utilise les fonctions de GeoPDE.

Ensuite, nous vérifions le bon fonctionnement de l'algorithme avec un problème à source externe. C'est la méthode de solution manufacturée, utilisée dans [6,13]. Ce problème demeure simple, avec un seul groupe d'énergie et sans anisotropie. Le but de ce problème est de mettre la discrétisation spatiale sous inspection. Une étude de convergence est menée à cette occasion.

Puis l'anisotropie est introduite avec un problème à fission monogroupe sur une géométrie cartésienne. Le choix de la géométrie permet d'éliminer l'influence de la discrétisation et de vérifier le bon fonctionnement de l'algorithme pour traiter un problème à diffusion anisotrope.

Enfin, un problème à fission multigroupe anisotrope sur la géométrie Pincell est résolu.

La motivation derrière cette gradation progressive est de vouloir repérer, étape après étape, les éventuelles erreurs dans la conception de l'algorithme. Il s'avère qu'un problème demeure non résolu au moment de la rédaction. L'étude de ce problème sera documentée avec détails dans la dernière partie de ce chapitre, dans l'espoir de guider quelconque lecteur voulant reproduire cette méthode de résolution.

4.1 Vérification de l'assemblage des matrices

L'exemple utilisé est un patch du Pincell uniformisé (fig 4.1)

Tableau 4.1 L'ensemble des paramètres du patch

Paramtres	
Vecteurs nodaux U, V	$\{0,0,0,1,1,1\}$ et $\{0,0,0,1,1,1\}$
Poids	$\omega_4 = 0.8535533$ $\omega_7 = \frac{1}{\sqrt{2}}$, les autres sont égaux à 1
Points de contrôle	$\begin{pmatrix}0.4725\\0.7896\end{pmatrix}, \begin{pmatrix}0.6310\\0.7896\end{pmatrix}, \begin{pmatrix}0.7896\\0.7896\end{pmatrix}, \begin{pmatrix}0.4104\\0.8517\end{pmatrix}, \begin{pmatrix}0.6310\\0.9582\end{pmatrix}, \begin{pmatrix}0.8517\\0.8517\end{pmatrix}, \begin{pmatrix}0.3482\\0.9139\end{pmatrix}, \begin{pmatrix}0.6310\\1.1967\end{pmatrix}, \begin{pmatrix}0.9139\\0.9139\end{pmatrix}$

On calcule les fonctions B-splines (cf équation 3.1) pour chaque direction paramétrique, et les $R_{i,p}$ sont calculés à partir de la formule 3.4 (cf Annexe A.2). Les résultats sont regroupés dans le tableau 4.2 :

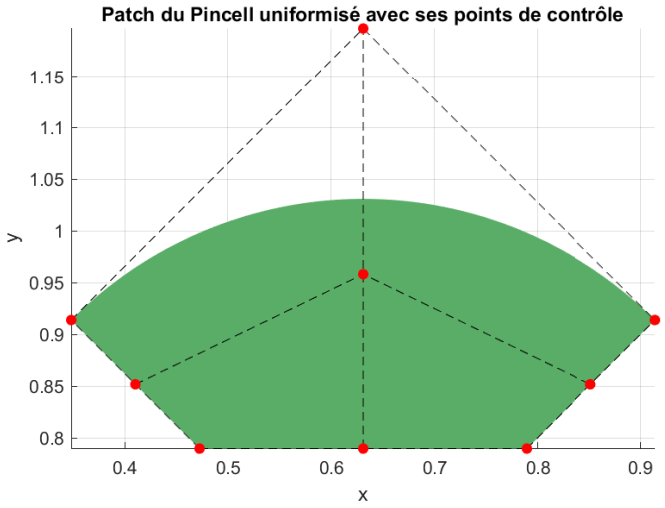


Figure 4.1 Patch utilisé pour la vérification des assemblages

Tableau 4.2 Les fonctions NURBS 2D du patch de la figure 4.1

Nom	Expression
$R_{0,2}$	$(1-u)^2(1-v)^2/S$
$R_{1,2}$	$2u(1-u)(1-v)^2/S$
$R_{2,2}$	$u^2(1-v)^2/S$
$R_{3,2}$	$(1-u)^2 2v(1-v)/S$
$R_{4,2}$	$2\omega_4 u(1-u)2v(1-v)/S$
$R_{5,2}$	$u^2 2v(1-v)/S$
$R_{6,2}$	$(1-u)^2v^2/S$
$R_{7,2}$	$2\omega_7 u(1-u)v^2/S$
$R_{8,2}$	u^2v^2/S

Où $S = \sum_{i=0}^{8} R_{i,2} = 1 - 2u(1-u)\left[2v(1-v)(1-\omega_4) + v^2(1-\omega_7)\right]$

Comme il n'y pas eu d'insertion de noeud, les fonctions sont définies sur le seul élément [0,1]. Les calculs suivant sont effectués sur MATLAB avec les calculs symboliques.

4.1.1 Vérification du package GeoPDE

Les fonctions NURBS sont évaluées aux 9 points de quadrature et comparées aux valeurs données par GeoPDE.

Tableau 4.3 Les points de quadrature du patch de la figure 4.1

q_u	0.1127	0.5000	0.8873	0.1127	0.5000	0.8873	0.1127	0.5000	0.8873
q_v	0.1127	0.1127	0.1127	0.5000	0.5000	0.5000	0.8873	0.8873	0.8873

Les résultats de comparaison sont regroupés dans le tableau 4.4, la position (i,j) correspond à la fonction $R_{j,2}$ au point de quadrature i :

Tableau 4.4 Différences en valeur absolue des fonctions NURBS évaluées aux points de quadrature

1.1102e-16	2.7756e-17	1.7347e-18	2.7756e-17	0	4.3368e-19	1.7347e-18	4.3368e-19	0
0	0	0	0	0	0	8.6736e-19	0	8.6736e-19
1.7347e-18	0	1.1102e-16	4.3368e-19	0	0	2.7105e-20	4.3368e-19	0
2.7756e-17	6.9389e-18	8.6736e-19	5.5511e-17	1.3878e-17	1.7347e-18	2.7756e-17	0	8.6736e-19
1.3878e-17	2.7756e-17	1.3878e-17	2.7756e-17	2.7756e-17	2.7756e-17	1.3878e-17	1.3878e-17	1.3878e-17
4.3368e-19	0	0	8.6736e-19	0	0	4.3368e-19	6.9389e-18	0
1.7347e-18	4.3368e-19	2.7105e-20	0	6.9389e-18	0	0	1.3878e-17	0
0	0	0	0	1.3878e-17	0	2.7756e-17	0	2.7756e-17
2.7105e-20	0	0	0	0	0	0	1.3878e-17	0

Nous vérifions également le jacobien (3.42), qui nous sert ensuite pour le calcul des gradients de fonctions. La position (j) correspond au point de quadrature (j) :

Tableau 4.5 Différence en valeur absolue du gradient évalué aux points de quadrature

0	1.3878e-17	1.3878e-17	0	2.7756e-17	1.3878e-17	5.5511e-17	0	1.3878e-17

Enfin, l'évaluation des gradients $\frac{\partial R_{j,p}}{\partial x}$ et $\frac{\partial R_{j,p}}{\partial y}$ sont vérifiées (équation 3.44) :

Tableau 4.6 Différences en valeur absolue des $\frac{\partial R_{j,p}}{\partial x}$ évaluées aux points de quadrature

0	4.4409e-16	0	6.6613e-16	0	2.7756e-17	2.7756e-17	0	8.6736e-19
0	0	0	0	0	0	6.9389e-18	0	6.9389e-18
2.2204e-16	8.8818e-16	8.8818e-16	6.9389e-17	3.3307e-16	4.4409e-16	2.6021e-18	1.3878e-17	2.7756e-17
1.1102e-16	4.4409e-16	5.5511e-17	2.2204e-16	2.2204e-16	1.1102e-16	0	5.5511e-17	4.1633e-17
1.1102e-16	0	1.1102e-16	2.2204e-16	0	2.2204e-16	1.1102e-16	0	1.1102e-16
2.7756e-17	2.2204e-16	1.1102e-16	5.5511e-17	2.2204e-16	2.2204e-16	2.7756e-17	1.1102e-16	4.4409e-16
0	5.5511e-17	5.2042e-18	4.4409e-16	2.2204e-16	2.7756e-17	0	2.2204e-16	2.7756e-17
0	0	0	0	0	0	2.2204e-16	0	2.2204e-16
0	0	5.5511e-17	0	0	4.4409e-16	0	0	8.8818e-16

Tableau 4.7 Différences en valeur absolue des $\frac{\partial R_{j,p}}{\partial y}$ évaluées aux points de quadrature

0	3.3307e-16	8.3267e-17	2.6645e-15	0	5.5511e-17	0	8.3267e-17	3.4694e-18
0	0	0	2.2204e-16	4.4409e-16	0	5.5511e-17	0	5.5511e-17
5.5511e-17	2.2204e-16	1.7764e-15	5.5511e-17	4.4409e-16	8.8818e-16	3.4694e-18	5.5511e-17	2.2204e-16
8.8818e-16	2.2204e-16	1.7347e-18	0	0	2.7756e-17	4.4409e-16	0	5.5511e-17
4.4409e-16	8.8818e-16	4.4409e-16	4.1633e-17	2.7756e-17	2.7756e-17	4.4409e-16	4.4409e-16	4.4409e-16
1.7347e-18	5.5511e-17	0	2.7756e-17	2.2204e-16	3.3307e-16	2.7756e-17	0	0
2.2204e-16	8.3267e-17	3.4694e-18	1.7764e-15	0	6.9389e-18	1.7764e-15	8.8818e-16	1.1102e-16
0	5.5511e-17	2.7756e-17	0	4.4409e-16	2.2204e-16	0	4.4409e-16	0
5.2042e-18	5.5511e-17	2.2204e-16	5.5511e-17	4.4409e-16	8.8818e-16	5.5511e-17	0	8.8818e-16

Il en résulte que les grandeurs calculées par GeoPDE et celles calculées à partir des formules sont concordantes à précision de calcul près pour ce patch. Les autres patches du Pincell sont similaires à ce patch étudié. Nous allons procéder à la vérification des matrices pour assurer leur bon assemblage.

4.1.2 Les matrices de volume

Nous allons vérifier les matrices $\underline{\underline{M}}$, $\underline{\underline{S_x}}$, et $\underline{\underline{S_y}}$. Le calcul des intégrales utilisent la fonction vpaintegral de MATLAB.

La matrice $\underline{\underline{M}}$ Le calcul de la matrice $\underline{\underline{M}}$ utilise la deuxième ligne de l'équation 3.40. Dans le tableau 4.8, la valeur en (i,j) correspond à la différence pour l'élément $\underline{\underline{M}}_{i,j}$

Tableau 4.8 Différences en valeur absolue des $\frac{\partial R_{j,p}}{\partial y}$ évaluées aux points de quadrature

2.5601e-07	1.9265e-05	1.1583e-05	7.1853e-07	6.6040e-06	1.0817e-05	5.5930e-06	4.9434e-06	3.5757e-06
1.9265e-05	4.6330e-05	1.9265e-05	7.7370e-06	3.6932e-05	7.7370e-06	6.9910e-06	1.0113e-05	6.9910e-06
1.1583e-05	1.9265e-05	2.5601e-07	1.0817e-05	6.6040e-06	7.1853e-07	3.5757e-06	4.9434e-06	5.5930e-06
7.1853e-07	7.7370e-06	1.0817e-05	2.2372e-05	2.3869e-05	1.4303e-05	1.0401e-05	5.8133e-06	2.3891e-05
6.6040e-06	3.6932e-05	6.6040e-06	2.3869e-05	4.1681e-05	2.3869e-05	7.0172e-06	5.7677e-05	7.0172e-06
1.0817e-05	7.7370e-06	7.1853e-07	1.4303e-05	2.3869e-05	2.2372e-05	2.3891e-05	5.8133e-06	1.0401e-05
5.5930e-06	6.9910e-06	3.5757e-06	1.0401e-05	7.0172e-06	2.3891e-05	6.1743e-05	2.8632e-05	5.5911e-05
4.9434e-06	1.0113e-05	4.9434e-06	5.8133e-06	5.7677e-05	5.8133e-06	2.8632e-05	1.1182e-04	2.8632e-05
3.5757e-06	6.9910e-06	5.5930e-06	2.3891e-05	7.0172e-06	1.0401e-05	5.5911e-05	2.8632e-05	6.1743e-05

La moyenne des différences est de l'ordre de 10^{-5}. Étant donné que les éléments de la matrice $\underline{\underline{M}}$ sont de l'ordre de 10^{-3}, l'écart relatif est autour du pour cent. Cette différence n'est pas négligeable puisque les seuils ϵ_K et ϵ_Φ sont usuellement 10^{-5}. Mais à ce stade, il est difficile d'expliquer la différence, puisque l'assemblage peut être aussi bien en cause que la différence de règle de quadrature utilisé par MATLAB et GeoPDE. Néanmoins, la fonction vpaintegral a été utilisée avec une tolérance relative de 10^{-32}.

Les matrices $\underline{\underline{S_x}}$ et $\underline{\underline{S_y}}$ Les calculs de ces matrices utilisent la deuxième ligne de l'équation 3.41.

Tableau 4.9 Différences en valeur absolue des éléments de la matrice $\underline{\underline{S_x}}$, calculée d'une part par la fonction d'assemblage et d'autre part via un calcul symbolique

7.5353e-05	7.6042e-05	1.2125e-05	9.6457e-05	1.3342e-04	2.0612e-05	9.0587e-05	9.6903e-05	6.9205e-06
1.0112e-04	8.7619e-19	1.0112e-04	1.6942e-04	4.7248e-19	1.6942e-04	1.5560e-04	8.2077e-20	1.5560e-04
1.2125e-05	7.6042e-05	7.5353e-05	2.0612e-05	1.3342e-04	9.6457e-05	6.9205e-06	9.6903e-05	9.0587e-05
3.0531e-05	5.1961e-05	2.0612e-05	1.7552e-04	1.6859e-04	2.7682e-05	3.0550e-04	4.7492e-04	5.4300e-05
5.5538e-05	1.1655e-19	5.5538e-05	2.3195e-04	3.2751e-19	2.3195e-04	6.0120e-04	4.2935e-19	6.0120e-04
2.0612e-05	5.1961e-05	3.0531e-05	2.7682e-05	1.6859e-04	1.7552e-04	5.4300e-05	4.7492e-04	3.0550e-04
2.8275e-06	3.8282e-05	6.9205e-06	3.1212e-05	9.7945e-05	5.4300e-05	1.9814e-05	2.4662e-04	1.3190e-04
1.3947e-05	5.3213e-20	1.3947e-05	5.8008e-05	5.1058e-20	5.8008e-05	1.4778e-04	1.0107e-20	1.4778e-04
6.9205e-06	3.8282e-05	2.8275e-06	5.4300e-05	9.7945e-05	3.1212e-05	1.3190e-04	2.4662e-04	1.9814e-05

Tableau 4.10 Différences en valeur absolue des éléments de la matrice $\underline{\underline{S_y}}$, calculée d'une part par la fonction d'assemblage et d'autre part via un calcul symbolique

2.4307e-05	1.4332e-04	8.6607e-05	1.8010e-05	1.4588e-04	1.1232e-04	4.3705e-05	7.4821e-05	9.7273e-05
1.2416e-04	3.4643e-04	1.2416e-04	1.4866e-04	3.8349e-04	1.4866e-04	9.0858e-05	2.7513e-04	9.0858e-05
8.6607e-05	1.4332e-04	2.4307e-05	1.1232e-04	1.4588e-04	1.8010e-05	9.7273e-05	7.4821e-05	4.3705e-05
5.0387e-06	3.8709e-05	1.3382e-05	5.0812e-05	1.2977e-04	1.3256e-04	2.7246e-04	2.5647e-04	3.6409e-04
1.4051e-05	4.5691e-05	1.4051e-05	7.8705e-05	3.8631e-04	7.8705e-05	2.5227e-04	8.7899e-04	2.5227e-04
1.3382e-05	3.8709e-05	5.0387e-06	1.3256e-04	1.2977e-04	5.0812e-05	3.6409e-04	2.5647e-04	2.7246e-04
1.8299e-05	2.9798e-05	3.0993e-05	9.4334e-05	3.7762e-05	1.2179e-04	2.9975e-04	3.0160e-05	2.7593e-04
3.1645e-05	8.7661e-05	3.1645e-05	7.8764e-05	2.9404e-04	7.8764e-05	1.7411e-04	5.5186e-04	1.7411e-04
3.0993e-05	2.9798e-05	1.8299e-05	1.2179e-04	3.7762e-05	9.4334e-05	2.7593e-04	3.0160e-05	2.9975e-04

Les différences sont de l'ordre de 10^{-4}. La moyenne en valeur absolue de la matrice $\underline{\underline{S_x}}$ est de $7 * 10^{-3}$ et celle de la matrice $\underline{\underline{S_y}}$ est de l'ordre de 10^{-2}. L'écart relatif est toujours de l'ordre du pour cent.

4.1.3 Les matrices de surface

Ce sont les matrices $\underline{\underline{P_k}}$ et $\underline{\underline{M'}}$ (Phi_bc et Phi_in dans l'algorithme). C'est l'assemblage le plus délicat, puisqu'une discontinuité est introduite dans l'intégrale, ce qui pourrait mettre la règle de quadrature en défaut. Seuls les calculs sur $\underline{\underline{P_k}}$ sont réalisés. Les matrices $\underline{\underline{M'}}$ partagent la même caractéristique de discontinuité et utilisent la même méthode d'assemblage que $\underline{\underline{P_k}}$. Toutefois il est possible de vérifier que $\underline{\underline{P_k}}_{i,j}$ est égale à $\underline{\underline{\Phi_{in}}}_{i',j'}$ du voisin, au signe et à la numérotation des fonctions près (car les patches sont uniformisés).

Pour le calcul des intégrales de surface, une formule supplémentaire doit être introduite pour le jacobien sur la surface qui se calcule à partir du jacobien de volume :

$$\| \underline{\underline{J_{surf}}} \| = \| \underline{\underline{J}}\hat{t} \| \tag{4.1}$$

Où $\hat{t}$ est le vecteur tangent à la surface dans l'espace paramétrique.

Tableau 4.11 Différences en valeur absolue du jacobien surfacique évaluées pour les 4 faces (j), aux points de quadrature (i)

0	0	5.5511e-17	1.1102e-16
2.7756e-17	2.7756e-17	0	0
0	0	0	1.1102e-16

Dans le tableau 4.11, la colonne j correspond à la face j et la ligne i correspond au $i$ème point de quadrature.

La matrice $\underline{\underline{P_k}}$ Un seul angle est choisi pour illustrer les calculs : $\begin{pmatrix} -0.8689 \\ -0.3500 \end{pmatrix}$. Les résultats sont regroupé dans le tableau 4.12.

Tableau 4.12 Différences en valeur absolue des éléments de la matrice $\underline{\underline{P_k}}$

0	3.4694e-18	8.6736e-19	8.6736e-18	0	0	2.6021e-18	0	0
3.4694e-18	3.4694e-18	3.4694e-18	0	0	0	0	0	0
8.6736e-19	3.4694e-18	1.0408e-17	0	0	0	0	0	0
8.6736e-18	0	0	1.0408e-17	0	0	5.2042e-18	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
2.6021e-18	0	0	5.2042e-18	0	0	0.0031	0.0017	1.0739e-04
0	0	0	0	0	0	0.0017	2.1478e-04	8.8820e-06
0	0	0	0	0	0	1.0739e-04	8.8820e-06	6.3284e-06

Le tableau contient beaucoup de 0 car beaucoup de fonctions NURBS s'annulent sur les frontières. Pour aider à lire le tableau, il est nécessaire de savoir que la numérotation des fonctions NURBS est croissante de gauche vers la droite et du bas vers le haut du patch. Ainsi le domaine de non nullité de $R_{0,2}$ est en bas à gauche du patch et celui de $R_{8,2}$ est en haut, à droite.

Nous remarquons que les différences sont beaucoup plus importantes pour les éléments $\underline{\underline{P_k}}_{\{i,j\}}$, $(i,j) \in \{7,8,9\}$. Ce sont les fonctions $R_{i,p}$ qui se trouvent sur la face courbe du patch, où le vecteur normal n'est pas constant. Le produit $\mathbf{\Omega} \cdot \mathbf{n}$ s'annule pour $u \approx 0.26600340870$ et il devient ensuite négatif.

Des points de quadrature supplémentaires sont ajoutés et le calcul est reconduit pour l'élément $(7,7)$. Cet élément est choisi, car c'est celui qui présente le plus grand écart de valeur. (les fonctions $R_{6,2}$ sont non nulles dans le coin supérieur gauche du patch, à proximité de u = 0.26). Les résultats sont présentés dans le tableau 4.13.

Il peut être intéressant d'augmenter la précision de la quadrature pour les intégrales de surface. Sans point supplémentaire, l'écart relatif pour $\underline{\underline{P_k}}_{\{7,7\}}$ est d'environ 5%. Toutefois,

Tableau 4.13 Différences en valeur absolue de $\underline{\underline{P_k}}_{\{7,7\}}$ en fonction du nombre de points de quadrature

Nombre de points	3	10	20	30	50
Différence $\underline{\underline{P_k}}_{\{7,7\}}$	0.0031	1.4086e-04	3.8093e-05	4.9642e-06	7.4735e-06

même avec 50 points de quadrature, l'intégrale est bien moins précise que celle effectuée sur les faces non courbes. L'écart avec 50 points de quadrature est plus grand qu'avec 30 points de quadrature. La différence est minimale quelque part entre les deux. Mais cela n'a pas de signification, puisqu'il s'agit de comparaison de deux calculs numériques et non d'un résultat numérique avec une valeur théorique. Aussi, il faut noter que nous avons un seul élément dans le patch, alors que les fonctions NURBS de degré p peuvent être non nulles sur p+1 éléments, et il y a p+1 points de quadrature par élément. C'est pourquoi il est toujours intéressant d'insérer au moins p+1 nœuds dans les vecteurs nodaux pour avoir une meilleure précision.

4.2 Méthode de la solution manufacturée

Cette méthode est utilisée dans [6, 13]. Le problème à résoudre est monogroupe et isotrope. L'objectif de cette section consiste à valider la méthode IGA pour sa discrétisation spatiale.

4.2.1 Présentation du problème

Pour ce faire, un flux angulaire connu est choisi égal à :

$$\Phi_k(x,y) = x^2y^2(1-x^2)(1-y^2)(1+\mu_k^2+\eta_k^2) \tag{4.2}$$

La source correspondante est calculée via l'équation 3.23 avec les simplifications apportées par l'isotropie de diffusion et le caractère monogroupe :

$$Q_k(x,y) = [(2x-4x^3)y^2(1-y^2)\mu_k + (2y-4y^3)x^2(1-x^2)\eta_k]c + (\Sigma - \frac{5}{3}\Sigma_s)x^2y^2(1-x^2)(1-y^2) \tag{4.3}$$

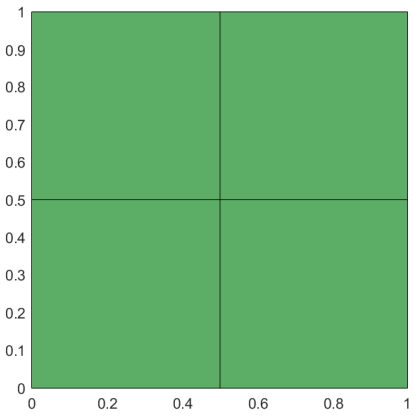
où

$$c = 1+\mu_k^2+\eta_k^2 \quad \Sigma = 1cm^{-1} \quad \Sigma_s = 0.5cm^{-1} \tag{4.4}$$

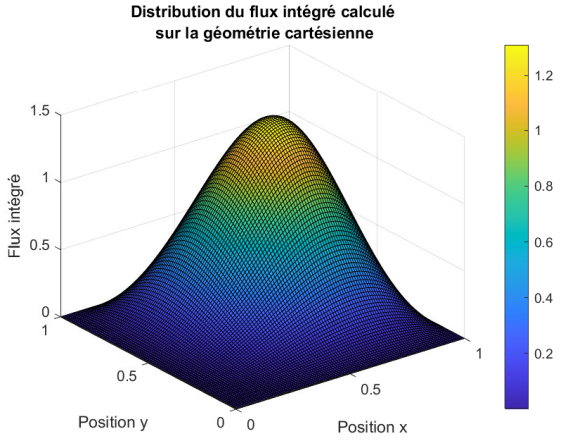
Cette expression de source diffère un peu de celle donnée dans [13] où le terme de diffusion n'est pas en facteur. Et [6] ne fournit pas d'expression explicite pour la source.

4.2.2 Résultats

Nous calculons les flux angulaires numériques en résolvant 3.37 sur deux géométries : l'une est une grille cartésienne composée de 4 carrés (figure 4.2), et l'autre est une géométrie type Pincell (figure 4.3), mais avec plus de cercles concentriques. La condition aux limites imposée est le vide. La norme L_2 est utilisée pour évaluer l'erreur.

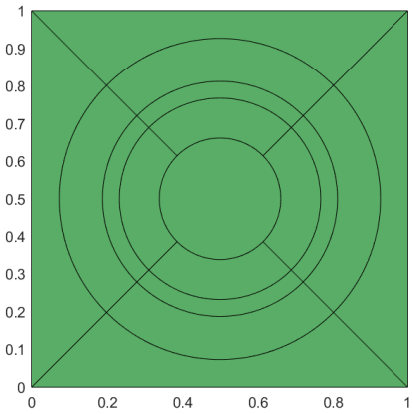


(a) Géométrie utilisée

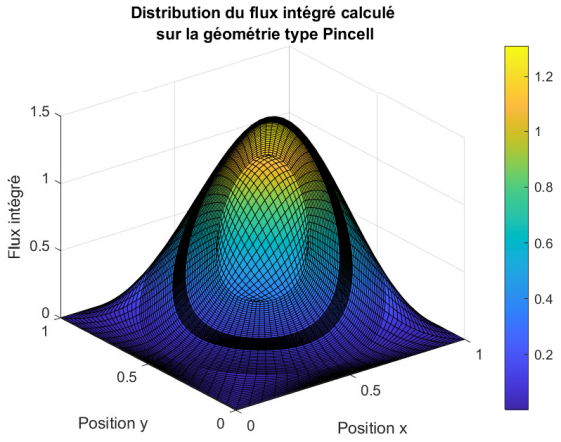


(b) Évaluation de la solution sur le maillage fin (h=0.01)

Figure 4.2 Solution du problème manufacturé, pour la géométrie cartésienne, S4

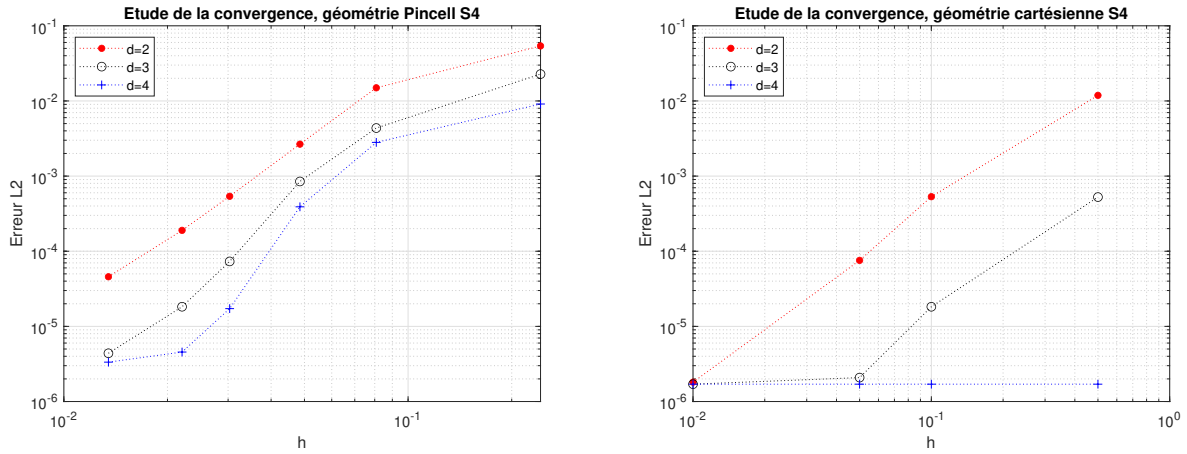


(a) Géométrie utilisée



(b) Évaluation de la solution sur le maillage fin (h=0.006)

Figure 4.3 Solution du problème manufacturé, pour la géométrie type Pincell, S4



(a) Géométrie Pincell

(b) Géométrie cartésienne

Figure 4.4 Graphes de convergence pour la géométrie type Pincell et la géométrie cartésienne

Sur les graphes de convergence (fig.4.4) nous observons qu'à raffinement/degré élevé, les résultats sont moins précis que ceux de [6, 13]. Il semble qu'il existe une erreur dans les calculs qui limite la précision de la solution à 10^{-6}. Il est difficile d'extrapoler les graphes pour la géométrie cartésienne, car ils sont rapidement limités. Selon la littérature [6], un ordre de convergence de p est attendu (où p est le degré des fonctions NURBS). Pour les graphes de la géométrie type Pincell, l'extrapolation est plus facile (en ignorant les points à 10^{-6}). Un ordre de convergence de p+1 est attendu [6], mais ce comportement n'est pas observé de manière convaincante. Cette différence de convergence peut être expliquée. Elle est dûe au fait que la géométrie type Pincell est initialement très divisée, elle contient en effet 17 patches, versus les 4 patches de la géométrie cartésienne. Il faut savoir qu'il existe deux méthodes de raffinements h. Dans le mémoire nous avons présenté une méthode qui consiste à insérer des nœuds pour avoir d'avantage d'éléments dans les patches. Mais le nombre de patch reste inchangé. Avec cette méthode, l'espace est divisé majoritairement en éléments sur lesquels la solution est au moins continue (cas de la géométrie cartésienne). La deuxième consiste alors à diviser un patch en plusieurs patches, ce qui, au sens strict, correspond au raffinement h de la FEM classique. Avec cette méthode, l'espace est divisé majoritairement en patches sur lesquels la solution est discontinue (elle demeure au moins continue à l'intérieur d'un patch). Cette différence de régularité explique le comportement différent sur les deux géométries. L'investigation du potentiel erreur limitant la précision est laissée dans la dernière partie du mémoire.

4.3 Problème à fission monogroupe anisotrope sur une géométrie cartésienne

Un problème à fission monogroupe anisotrope est traité dans cette section.

4.3.1 Présentation du problème

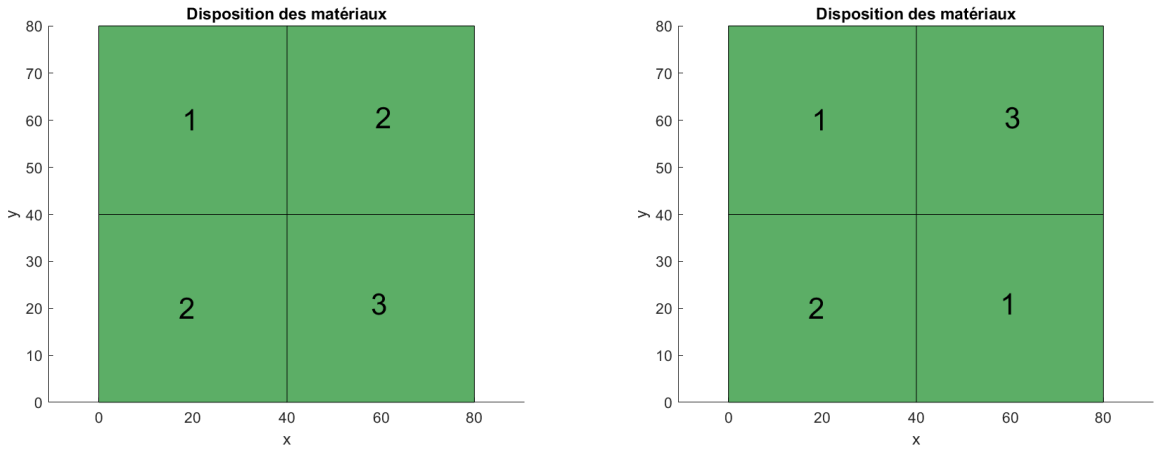
La géométrie est cartésienne, constituée de 4 carrées, comme celle de la figure 4.2a, à qui est appliquée une homothétie. La condition réflective est appliquée aux frontières. Désormais c'est l'équation 3.38 qui gouverne le problème. Les matériaux utilisés sont les suivants :

Tableau 4.14 Les matériaux utilisés dans le problème à fission monogroupe anisotrope

Matériau	Σ	$\nu\Sigma_f$	χ	$\Sigma_{s,0}$	$\Sigma_{s,1}$
1	0.025	0.0155	1.0	0.013	0.0
2	0.025	0.0	0.0	0.024	0.006
3	0.075	0.0	0.0	0.0	0.0

Il s'agit d'un cas test courant de DRAGON pour tester le bon fonctionnement de l'anisotropie. Le matériau 1 est fissile, à diffusion isotrope ; le matériau 2 est à diffusion anisotrope et le matériau 3 est purement absorbant.

Deux dispositions des matériaux ont été étudiées.



(a) Disposition 1

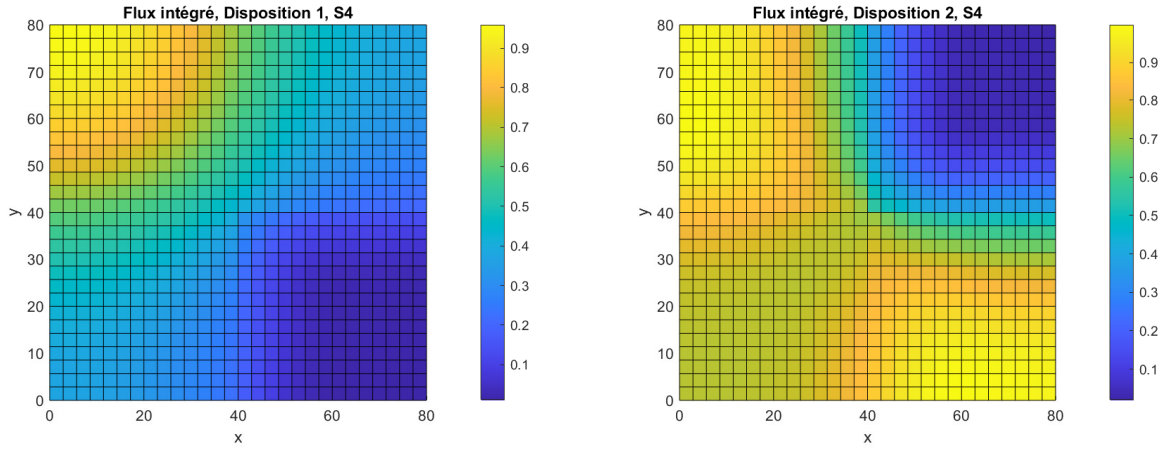
(b) Disposition 2

Figure 4.5 Dispositions des matériaux

4.3.2 Résultats

Les valeurs calculées par DRAGON servent de référence. Le lecteur intéressé peut visiter le site Merlin détenu par l'École Polytechnique de Montréal (accès libre) et se référer à [1].

Comme c'est le module FEM de DRAGON qui est utilisé ici, une attention est apportée à ce que le nombre de régions et le degré des polynômes soient concordants entre les deux méthodes. Les résultats sont regroupés dans les tableaux 4.15 et 4.16



(a) Disposition 1 (b) Disposition 2

Figure 4.6 Flux intégré numérique pour les deux dispositions de matériaux, S4. Le quadrillage est un quadrillage de tracé et ne représente pas les éléments des patches

Tableau 4.15 Comparaison des valeurs de Keff obtenus par l'algorithme avec celles calculées par DRAGON, pour la disposition 1 du problème monogroupe anisotrope

	DRAGON	IGA	écart-relatif (et en pcm)
S_4	0.7520776	0.7520494	0.0037% (2.81pcm)
S_8	0.7637635	0.7637882	0.0032%(2.41pcm)
S_{16}	0.7659803	0.7659652	0.0020% (1.51 pcm)

Les résultats sont satisfaisants, néanmoins cette section ne fait que montrer que l'anisotropie est traitée correctement sur une géométrie cartésienne. Cela valide en partie le calcul des sources de diffusion.

Tableau 4.16 Comparaison des valeurs de Keff obtenus par l'algorithme avec celles calculées par DRAGON, pour la disposition 2 du problème monogroupe anisotrope

	DRAGON	IGA	écart-relatif (et en pcm)
S_4	0.8203996	0.8201319	0.0326% (26.8pcm)
S_8	0.8270494	0.8269634	0.0104%(8.59pcm)
S_{16}	0.8283880	0.8283086	0.0096% (7.94pcm)

4.4 Problème à fission multigroupe anisotrope sur une géométrie Pincell

Le caractère multigroupe et la géométrie non cartésienne sont les nouveautés de cette section.

4.4.1 Présentation du problème

Nous étudions dans cette section un problème à 4 groupes avec une anisotropie d'ordre 1. La géométrie utilisée est similaire à celle de la 3.11, à laquelle est ajoutée une couronne fine autour du disque central. L'étude est menée en comparant les flux intégrés et le facteur de multiplication effectif entre notre algorithme et DRAGON. DRAGON utilise la méthode des caractéristiques pour la résolution de l'équation, donc la discrétisation S_N ne concerne que la méthode IGA. Quant au raffinement de la géométrie, une élévation de 1 est adoptée pour le degré et 4 nœuds sont insérés par patch. La géométrie est constituée de 5 patches pour le disque, 4 patches pour la couronne et 4 patches pour le reste. Il y en a en tout 325 éléments. La condition de réflection est imposée sur les frontières.

Nous allons étudier 2 dispositions. La première disposition ne contient que 2 matériaux. Le disque central et la couronne contiennent du combustible, et de l'eau les entoure. Dans la deuxième disposition, la couronne contient désormais de l'AIC (gaine, très absorbante pour les neutrons lents), le reste étant inchangé.

Les matériaux utilisés sont regroupés dans les tableaux ci-dessous, les données proviennent de du cas test TCM77 de DRAGON, édité par A.Hébert. La convention suivante est adoptée pour les sections efficaces de diffusion : (i, j) correspond à la diffusion de neutrons de groupe j vers le groupe i.

4.4.2 Résultats

Disposition sans gaine

Les flux sont représentés aux figures 4.7 et 4.8. Les flux intégrés(cf tab.4.21) sont normalisés

Tableau 4.17 Le matériau 1 - Eau utilisé dans le problème à fission multigroupe anisotrope

Matériau	1 - Eau			
Σ	5.316E-01	9.352E-01	9.931E-01	1.5931E+00
$\nu\Sigma_f$	0	0	0	0
χ	0	0	0	0
$\Sigma_{s,0}$	4.973E-01	0	0	0
	3.134E-02	6.570E-01	0	0
	2.304E-03	2.528E-01	7.176E-01	1.163E-02
	2.228E-04	2.412E-02	2.707E-01	1.565E+00
$\Sigma_{s,1}$	2.735E-01	0	0	0
	1.410E-02	4.253E-01	0	0
	3.050E-04	1.203E-01	4.286E-01	5.064E-03
	1.245E-05	3.373E-03	6.426E-02	4.384E-01

Tableau 4.18 Le matériau 2 - Combustible utilisé dans le problème à fission multigroupe anisotrope

Matériau	2 - Combustible			
Σ	4.097E-01	6.286E-01	4.594E-01	7.508E-01
$\nu\Sigma_f$	1.632E-02	9.259E-02	1.129E-01	6.508E-01
χ	1	0	0	0
$\Sigma_{s,0}$	3.908E-01	0	0	0
	8.546E-04	4.407E-01	0	0
	0	8.057E-03	3.785E-01	5.318E-03
	0	0	1.203E-02	3.983E-01
$\Sigma_{s,1}$	4.793E-02	0	0	0
	-2.640E-04	5.419E-03	0	0
	0	-2.453E-03	9.014E-03	-7.338E-04
	0	0	-2.572E-03	8.769E-03

Tableau 4.19 Le matériau 3 - AIC utilisé dans le problème à fission multigroupe anisotrope

Matériau	3 - AIC			
Σ	4.240E-01	1.0476E+00	2.7503E+00	1.1194E+01
$\nu\Sigma_f$	0	0	0	0
χ	0	0	0	0
$\Sigma_{s,0}$	3.738E-01	0	0	0
	2.439E-04	4.102E-01	0	0
	0	7.370E-04	4.956E-01	4.985E-03
	0	0	3.248E-03	2.996E-01
$\Sigma_{s,1}$	0	0	0	0
	0	-6.161E-02	0	0
	0	0	-2.326E-01	0
	0	0	0	0

Tableau 4.20 Comparaison de K_{eff} entre DRAGON et IGA, Pincell sans gaine

DRAGON	S_N	IGA	ϵ_k
1.301027	S_4	1.301500	0.036%(47pcm)
	S_8	1.301669	0.049%(64pcm)
	S_{16}	1.300942	0.006%(8.5pcm)

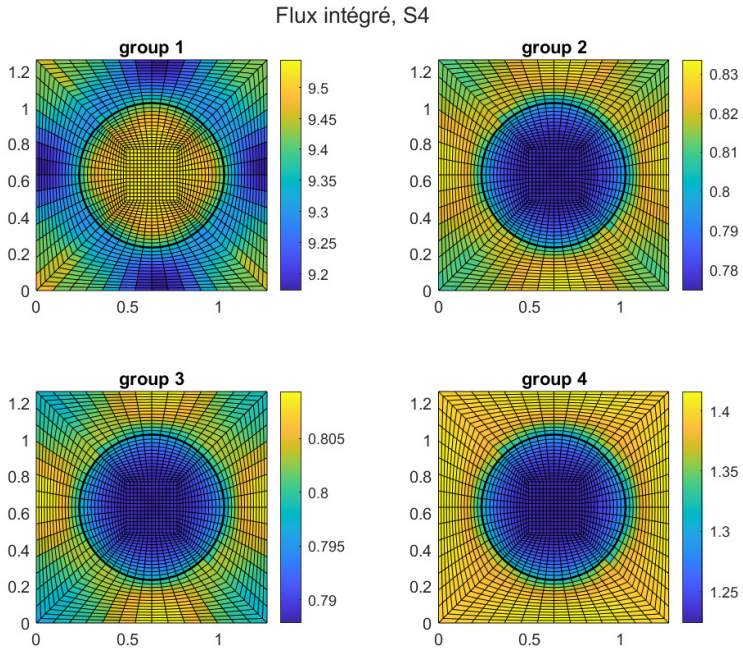


Figure 4.7 Flux intégré représenté sur un quadrillage de tracé, disposition sans gaine, S4

de sorte que $\int_V \nu\Sigma_f\Phi = 1$.

En comparant les résultats entre S_4, S_8 et S_{16}, nous pouvons constater que globalement la précision du calcul s'améliore avec l'ordre S_N. Étant donné que DRAGON utilise la méthode MOC, il est naturel que les solutions soient d'autant plus proches qu'il y a d'angles. Cependant la valeur de K_{eff} pour S_8 est étrange, l'écart de K_{eff} croît alors que l'écart des flux normalisés diminue ! Pour S_{16}, bien que les précisions soient satisfaisantes, l'observation du flux intégré avec une vue de profil (figure 4.9) suggère qu'il y a une imperfection dans l'algorithme. En effet, malgré le faible écart pour le flux à l'intérieur du disque, la solution n'est pas très lisse au niveau des interfaces.

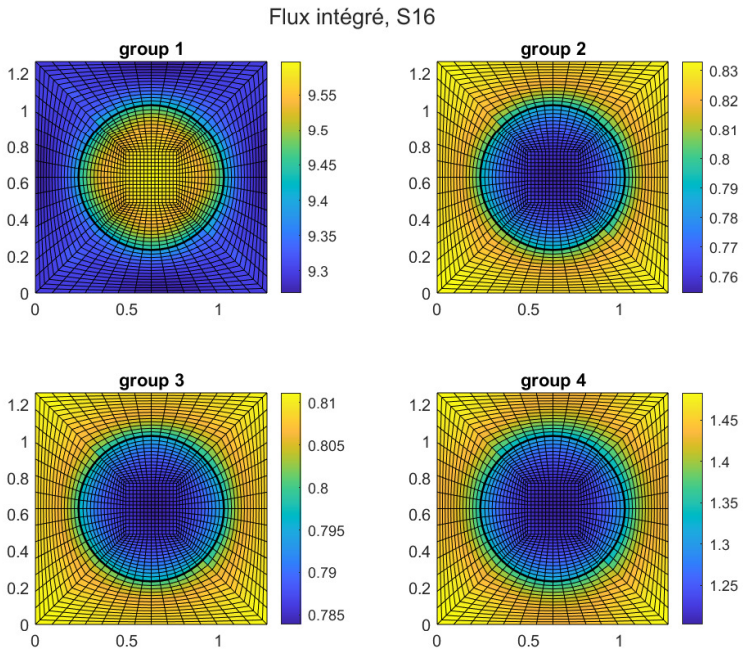


Figure 4.8 Flux intégré représenté sur un quadrillage de tracé, disposition sans gaine, S16

Tableau 4.21 Comparaison de Φ normalisé entre DRAGON et IGA, Pincell sans gaine

S_N	Groupe	Région	DRAGON	IGA	ϵ_Φ
S_4	1	1	129.5047	130.8776	1.0602%
		2	61.27470	61.27451	0.0003%
	2	1	24.93594	24.41684	2.0817%
		2	10.80030	10.80030	0.00006%
	3	1	19.63385	19.46827	0.8433%
		2	8.857361	8.857396	0.0004%
	4	1	3.774457	3.678325	2.5469%
		2	1.536572	1.536570	0.0001%
S_8	1	1	129.5047	130.5423	0.8012%
		2	61.27470	61.27451	0.0003%
	2	1	24.93594	24.71288	0.8945%
		2	10.80030	10.80030	0.00006%
	3	1	19.63385	19.55573	0.3979%
		2	8.857361	8.857396	0.0004%
	4	1	3.774457	3.758241	0.4296%
		2	1.536572	1.536570	0.0001%
S_{16}	1	1	129.5047	130.0092	0.3895%
		2	61.27470	61.27451	0.0003%
	2	1	24.93594	24.91903	0.0677%
		2	10.80030	10.80030	0.00006%
	3	1	19.63385	19.62344	0.0530%
		2	8.857361	8.857396	0.0004%
	4	1	3.774457	3.788554	0.3735%
		2	1.536572	1.536570	0.0001%

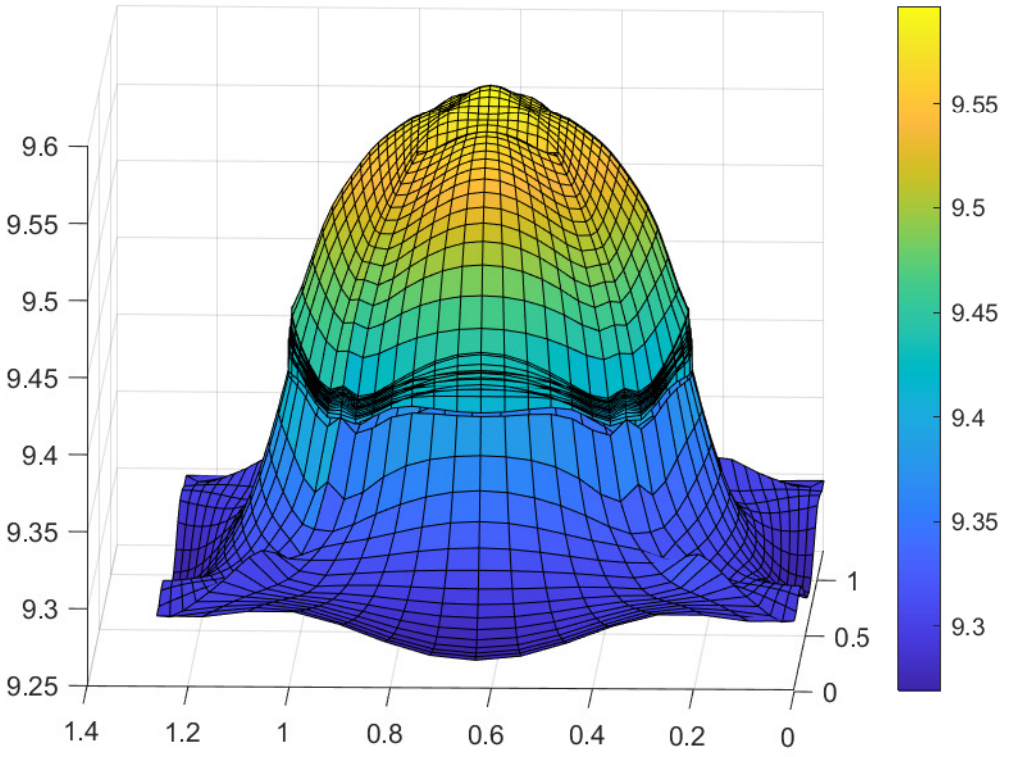


Figure 4.9 Flux intégré groupe 1, S16 vue de profil

Disposition avec gaine

Tableau 4.22 Comparaison de K_{eff} entre DRAGON et IGA, Pincell avec gaine

DRAGON	S_N	IGA	ϵ_k
0.7363632	S_4	0.7520566	2.131%(1500pcm)
	S_8	0.7464898	1.375%(1013pcm)
	S_{16}	0.7502602	1.887%(1390pcm)

Les flux sont représentés sur la figure 4.10 et les flux intégrés normalisés sont regroupés dans le tableau 4.23

Les écarts en K_{eff} sont trop importants pour valider la méthode avec cette disposition bien que les flux normalisés soient toujours calculés avec une bonne précision (sauf dans la gaine). L'ajout de la gaine accentue davantage l'importance de la transmission de l'information entre patches, et semble donc indiquer que c'est cette partie de l'algorithme qui est en défaut.

Nous pouvons également remarquer qu'il y a une asymétrie selon la diagonale d'équation $y = -x$ dans la distribution du flux intégré. Cette asymétrie est bien visible pour le groupe

d'énergie 3 sur la figure 4.10. L'origine de cette asymétrie n'est pas claire. Le problème est symétrique par rapport à cette diagonale ; les influences de l'ordre de balayage des patches pour différents angles se compensent mutuellement. Puis cette asymétrie ne se retrouve pas dans le cas du Pincell sans gaine. Par ailleurs, la distribution du flux est symétrique selon l'autre diagonale $y = x$. La différence entre ces deux diagonales réside dans le fait que les interfaces selon ces diagonales correspondent à des patches dont les directions des paramètres sont concourantes ou non. Une interface sur la diagonale $y = -x$ sépare deux patches avec des directions de paramètres opposées, tandis que les directions de paramètres sont concourantes pour l'autre diagonale $y = x$. Il s'agit encore, probablement, d'un problème de bonne transmission d'information.

Tableau 4.23 Comparaison de Φ normalisé entre DRAGON et IGA, Pincell avec gaine

S_N	Groupe	Région	DRAGON	IGA	ϵ_Φ
S_{16}	1	1	134.4772	134.0111	0.3478%
		2	61.27450	61.27469	0.0003%
		3	2.165523	2.154338	0.5165%
	2	1	25.95032	25.95032	0.1356%
		2	10.80031	10.80030	0.00006%
		3	0.391892	0.393753	0.4747%
	3	1	20.76884	20.65737	0.5367%
		2	8.857395	8.857396	0.00001%
		3	0.316321	0.310287	1.9077%
	4	1	4.372095	4.313488	1.3404%
		2	1.536572	1.536570	0.0001%
		3	0.056327	0.052930	6.0305%

Il faut aussi noter que le temps de calcul de l'algorithme est très long par rapport à DRAGON. Pour S_{16}, le calcul a pris approximativement 1 heure tandis que le calcul avec la méthode MOC dans DRAGON est quasi instantané. Le seuil d'erreur est le même pour les deux méthodes : $\epsilon = 10^{-5}$. Il existe plusieurs méthodes d'accélération dans la littérature. La méthode de diffusion synthétique est couramment utilisée pour accélérer la méthode de source itérée [2, 5], et elle a déjà été déployée dans le cadre de la méthode IGA [13]. L'optimisation n'a pas été le sujet de ce mémoire, car l'algorithme n'est pas suffisamment précis.

Remarque :

Les calculs ont été refaits avec $\epsilon = 10^{-10}$ pour le cas du Pincell avec gaine, avec la discrétisation S_4. Les résultats sont similaires à ceux déjà obtenus, avec $K_{eff} = 0.7520645$ soit $\epsilon_k = 2.132\%$ $(1570pcm)$

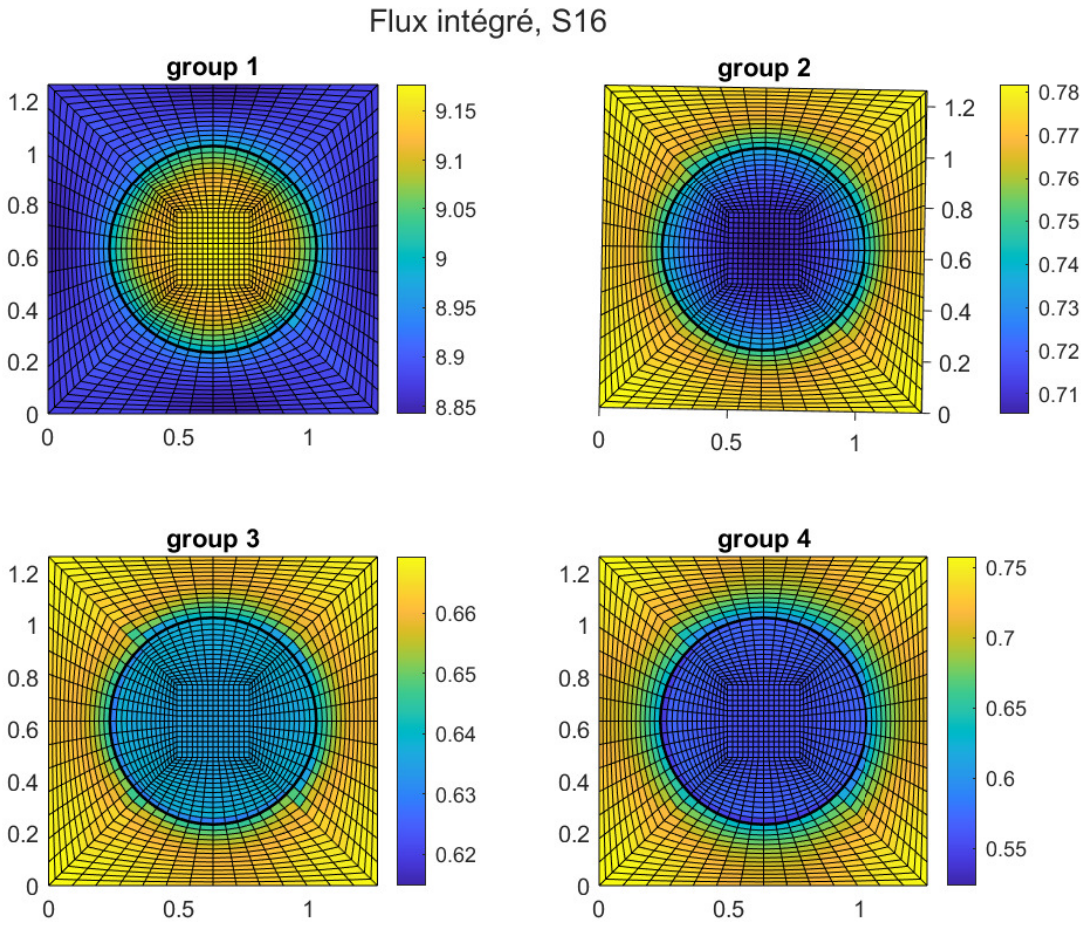


Figure 4.10 Flux intégré représenté sur un quadrillage de tracé, disposition avec gaine, S16

4.5 Problème à fission multigroupe anisotrope sur un Colorset

Un Colorset est un assemblage de 4 Pincells. Cette section teste différentes compositions du Colorset. Les matériaux de la section précédente (sec 4.4) sont utilisés pour constituer les Pincells. L'étude est réalisée avec une discrétisation S_{16} ; une condition aux limites réflective est appliquée, et 3 noeuds sont insérés pour des fonctions NURBS de degré 2. La méthode MOC de DRAGON sert toujours de référence.

4.5.1 Colorset 1

Ce Colorset est composé de deux types de Pincell. Ils sont constitués de la même façon : du combustible entouré par de l'eau (sans gaine). Les deux types de Pincell se différencient par le combustible utilisé. Un nouveau combustible est donc introduit, ses caractéristiques sont regroupées dans le tableau 4.24.

Tableau 4.24 Combustible 2 utilisé dans le problème à fission multigroupe anisotrope

Matériau	Combustible 2			
Σ	4.072E-01	6.223E-01	4.599E-01	7.504E-01
$\nu\Sigma_f$	1.652E-02	9.136E-02	1.143E-01	6.500E-01
χ	1	0	0	0
$\Sigma_{s,0}$	3.885E-01	0	0	0
	8.562E-04	4.387E-01	0	0
	0	8.591E-03	3.780E-01	5.236E-03
	0	0	1.229E-02	3.984E-01
$\Sigma_{s,1}$	4.981E-02	0	0	0
	-2.624E-04	3.100E-03	0	0
	0	-2.610E-03	1.142E-02	-7.231E-04
	0	0	-2.604E-03	8.764E-03

La disposition est représentée par la figure 4.11. Le dessin n'est pas à échelle. Les comparaisons de K_{eff} et du flux normalisé sont présentées dans les tableaux 4.25 et 4.26. Les graphes des flux intégrés, n'apportant pas d'informations supplémentaires, sont laissés en Annexe (cf Annexe B)

Tableau 4.25 Comparaison de K_{eff} entre DRAGON et IGA, Colorset 1

DRAGON	S_N	IGA	ϵ_k
1.305878	S_{16}	1.305922	0.003%(4.4 pcm)

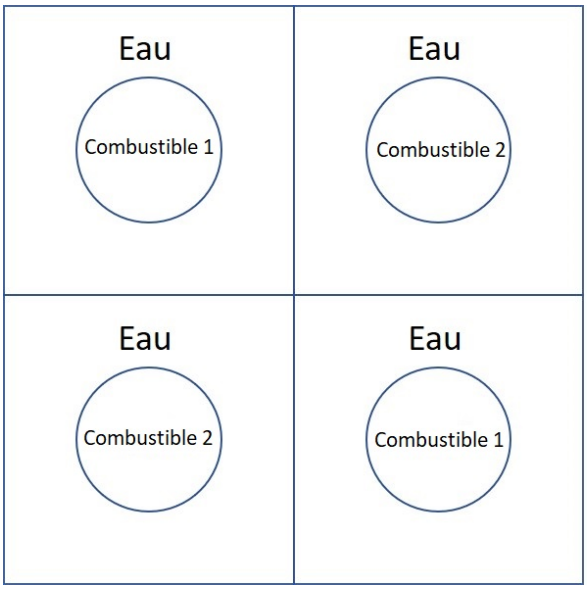


Figure 4.11 Disposition du Colorset 1

Tableau 4.26 Comparaison de Φ normalisé entre DRAGON et IGA, Colorset 1 (Régions 1 :Eau, 2 :Combustible 1, 3 :Combustible 2)

S_N	Groupe	Région	DRAGON	IGA	ϵ_Φ
S_{16}	1	1	128.6311	129.2221	0.4595%
		2	30.44827	30.44861	0.0011%
		3	30.45305	30.45270	0.0011%
	2	1	25.12078	25.06927	0.2050%
		2	5.430542	5.430981	0.0081%
		3	5.442054	5.441609	0.0082%
	3	1	19.52758	19.50486	0.1164%
		2	4.401984	4.402018	0.0007%
		3	4.400839	4.400806	0.0007%
	4	1	3.786689	3.790845	0.1098%
		2	0.768616	0.768624	0.0010%
		3	0.768899	0.768892	0.0010%

La précision du résultat est satisfaisant, cependant le temps de calcul est excessivement long (3 à 4 heures de calcul). Le flux converge très lentement vers la fin des itérations (lorsque l'erreur avoisine 10^{-5}).

4.5.2 Colorset 2

Le Colorset 2 remplace un des Pincells du Colorset 1 par un Pincell rempli d'eau.

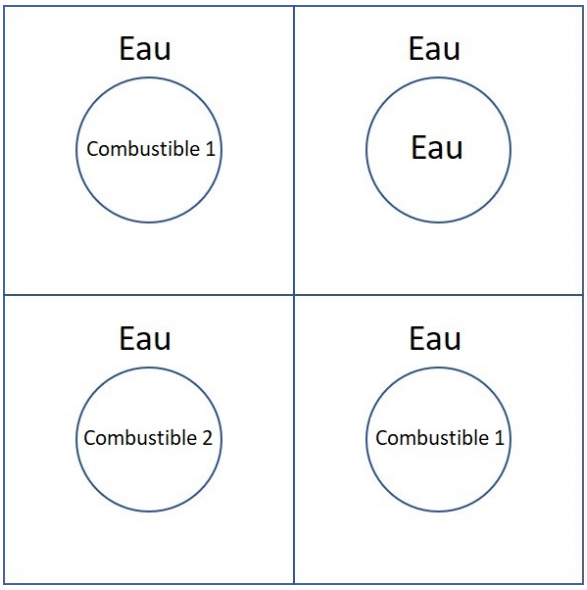


Figure 4.12 Disposition du Colorset 2

La comparaison de K_{eff} et du flux normalisé est présentée dans les tableaux 4.27 et 4.28. Les graphes sont en Annexe C.

Tableau 4.27 Comparaison de K_{eff} entre DRAGON et IGA, Colorset 2

DRAGON	S_N	IGA	ϵ_k
1.317965	S_{16}	1.318779	0.06%(81 pcm)

Tableau 4.28 Comparaison de Φ normalisé entre DRAGON et IGA, Colorset 2 (Régions 1 :Eau, 2 :Combustible 1, 3 :Combustible 2)

S_N	Groupe	Région	DRAGON	IGA	ϵ_Φ
S_{16}	1	1	186.9627	188.2415	0.6840%
		2	40.60075	40.61057	0.0242%
		3	20.42345	20.41377	0.0474%
	2	1	37.85193	37.74667	0.2781%
		2	7.251344	7.251953	0.0084%
		3	3.596735	3.596122	0.0170%
	3	1	29.38852	29.31391	0.2539%
		2	5.901042	5.900905	0.0023%
		3	2.920148	2.920278	0.0045%
	4	1	6.028654	6.006860	0.3615%
		2	1.044478	1.044870	0.0375%
		3	0.492695	0.492305	0.0790%

Les résultats de ce Colorset est moins précis que ceux du Colorset 1. Encore une fois, ceci s'explique par le fait que la transmission d'information joue un rôle plus important puisqu'il y a moins de production de neutrons par fission. Ce dernier point a déjà été mis en évidence par l'étude des Pincells et il se reproduit ici.

4.5.3 Colorset 3

Une gaine est désormais ajoutée aux Pincells, et nous reprenons la disposition du Colorset 1.

Tableau 4.29 Comparaison de K_{eff} entre DRAGON et IGA, Colorset 3

DRAGON	S_N	IGA	ϵ_k
0.7399706	S_{16}	0.752417	1.682%(1244 pcm)

Tableau 4.30 Comparaison de Φ normalisé entre DRAGON et IGA, Colorset 3 (Régions 1 :Eau, 2 :Combustible 1, 3 :Combustible 2 4 :AIC)

S_N	Groupe	Région	DRAGON	IGA	ϵ_Φ
S_{16}	1	1	133.0882	133.6589	0.4288%
		2	30.44735	30.44777	0.0014%
		3	30.45393	30.45353	0.0013%
		4	2.149330	2.139907	0.4384%
	2	1	26.14269	26.07054	0.2760%
		2	5.430672	5.431051	0.0070%
		3	5.441926	5.441538	0.0071%
		4	0.395302	0.396167	0.2189%
	3	1	20.64776	20.53011	0.5698%
		2	4.401796	4.401870	0.0017%
		3	4.401020	4.400952	0.0015%
		4	0.314243	0.308837	1.7204%
	4	1	4.378438	4.315646	1.4341%
		2	0.768594	0.768604	0.0014%
		3	0.768918	0.768911	0.0009%
		4	0.056183	0.053150	5.3990%

Les graphes se trouvent en Annexe D. Les résultats manquent de précision. L'écart de flux est particulièrement important dans la gaine pour les groupes d'énergie faible, pour lesquels la section efficace totale est importante. Les résultats corroborent ceux du Pincell (cf le tableau 4.23). Ce qui n'est pas surprenant, le même phénomène limite la précision.

4.6 Erreur de l'algorithme

Au cours des dernières sections, nous avons pu remarquer quelques anomalies dans le fonctionnement de l'algorithme. Il semble qu'il y ait un problème dans la transmission d'information entre patches lorsqu'ils ne sont pas cartésiens. Ce problème demeure non résolu lors de la rédaction, néanmoins, un cas d'étude suffisamment simple a été trouvé pour essayer de débugger l'algorithme. L'algorithme a été codé à nouveau, en vain. Cette section présente ce cas d'étude avec détails, avec les démarches entreprises et les observations obtenues, dans l'espoir de guider quelconque successeur.

4.6.1 Présentation

La géométrie est un carré, séparé en 2 patches par un segment oblique reliant deux faces opposées. Des conditions de réflexions spéculaires sont appliquées autour du domaine. L'exemple choisi pour illustrer cette section utilise le segment $\left[\begin{pmatrix}0.6\\0\end{pmatrix};\begin{pmatrix}0.4\\1\end{pmatrix}\right]$.

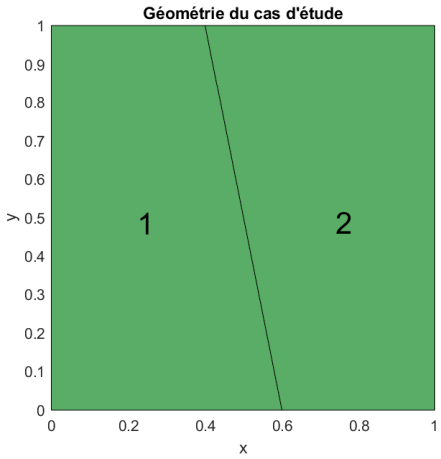


Figure 4.13 Géométrie du cas d'étude

Le code suivant permet de générer la géométrie. La fonction nrb4surf provient de NURBS Toolbox, inclu dans le package GeoPDE.
$z1 = nrb4surf([0\ 0], [0.6\ 0], [0\ 1], [0.4\ 1]);$
$z2 = nrb4surf([0.6\ 0], [1\ 0], [0.4\ 1], [1\ 1]);$
Il est aisé de modifier le segment en changeant les coordonnées des points intervenant dans ces deux lignes de code.

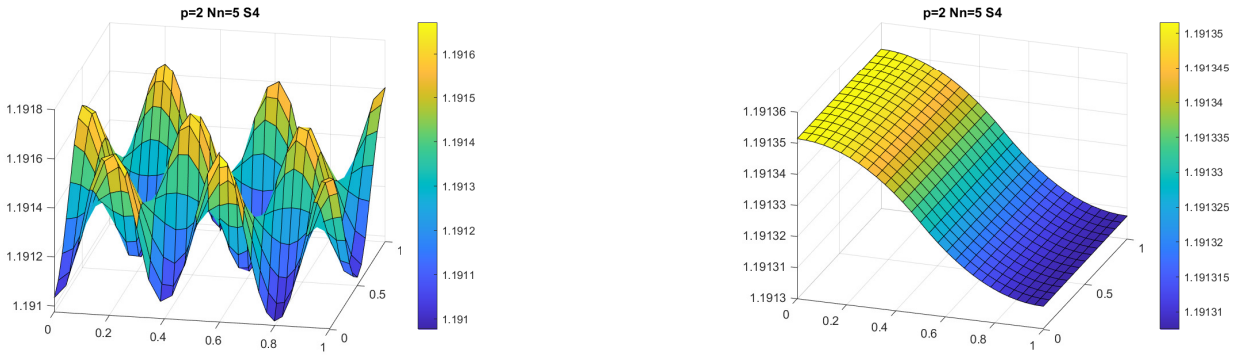
Les matériaux de la section 4.3 sont utilisés, mais le développement de la section efficace est limité à 0. Il s'agit donc d'un problème à fission monogroupe isotrope, avec des conditions

aux limites réflectives.

4.6.2 Anomalie et étude

Anomalie :

Le flux intégré calculé par l'algorithme présente des oscillations artificielles. Ces oscillations sont d'amplitude faible mais n'ont pas d'explication physique à leur existence. C'est probablement un facteur limitant la précision de nos calculs. Et si le segment de séparation devient vertical, les oscillations disparaissent, ce qui correspond à l'hypothèse de mauvaise transmission d'information au travers d'une interface non cartésienne.



(a) Géométrie avec segment oblique

(b) Géométrie avec segment droit

Figure 4.14 Flux intégré pour la géométrie avec un segment de séparation oblique et droit

L'investigation commence avec une étude paramétrique, où les paramètres sont le degré p, le nombre de noeuds insérés N_n et l'ordre S_N. Le paramètre p varie de 1 à 3 ; N_n prend les valeurs 0, 5 et 15. Quant à l'ordre S_N, il peut être 4, 8 ou 16. L'étude sur p et N_n est menée avec S_4. Le tableau 4.31 regroupe les valeurs de K_{eff} sous forme d'écart par rapport à la moyenne. La figure 4.15 représente les flux intégrés.

Tableau 4.31 Écart par rapport à la moyenne $\overline{K_{eff}} = 1.191345292496776$

Nombre de noeuds insérés	0	5	15
p=0	9.9024e-07	1.4233e-07	3.9208e-08
p=1	-3.8930e-07	1.3285e-07	-9.1552e-09
p=2	-8.8918e-07	-5.3961e-09	-1.1598e-08

L'influence de l'ordre S_N est étudiée avec $p = 2$ et $N_n = 5$. Les flux sont représentés à la figure 4.16 :

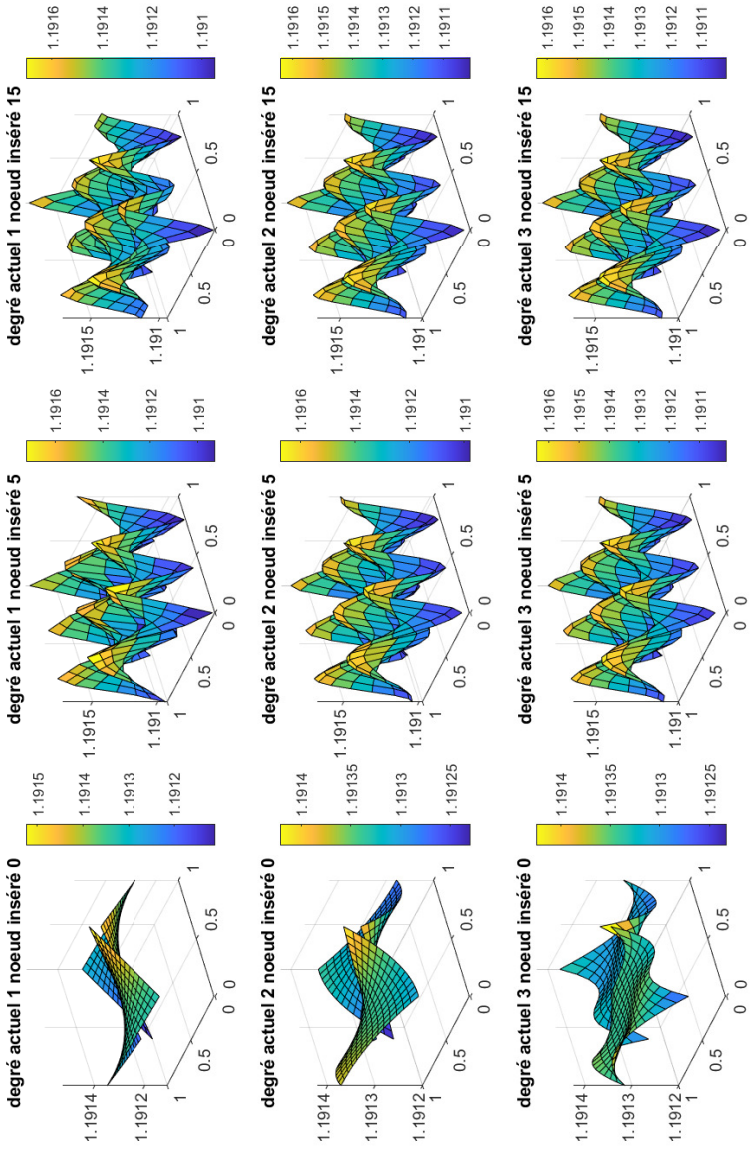
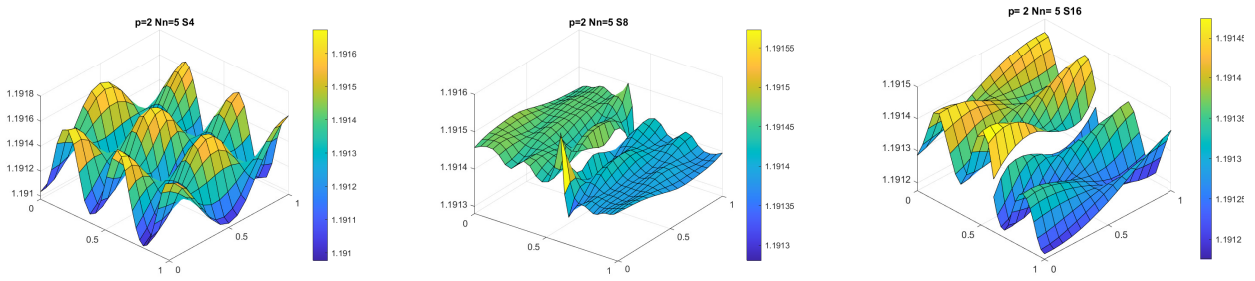


Figure 4.15 Flux intégré, pour $p = 0, 1, 2$ et $N_n = 0, 5, 15$



(a) S4 (b) S8 (c) S16

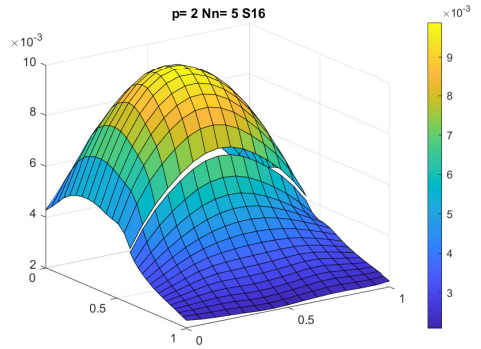
Figure 4.16 Flux intégré pour différents ordres S_N

De cette étude, il semblerait que nous ne pouvons pas déduire d'informations pertinentes à partir de K_{eff}, ses valeurs ne seront plus mentionnées. Néanmoins, au-delà d'une certaine précision, nous remarquons que les oscillations ne changent plus en fonction des paramètres p et N_n. La limite de la précision n'est probablement pas en lien avec ces paramètres. Puis le fait qu'il n'y ait pas d'oscillation pour un segment vertical élimine aussi le potentiel problème de maillage insuffisant.

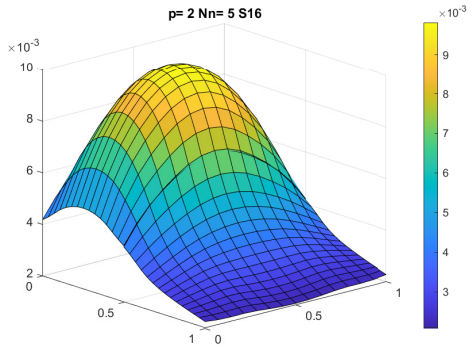
Il est possible de remarquer que les oscillations sont sensibles à l'ordre S_N. Malgré la différence de géométrie, la solution théorique devrait être similaire au flux sans oscillation de la figure 4.14b. L'ordre S_{16} donne un flux qui s'en approche le plus. Il semblerait que l'angle soit le paramètre en question.

L'angle intervient à 2 niveaux dans la BTE : pour les matrice $\underline{\underline{S_x}}$ et $\underline{\underline{S_y}}$ et pour les matrices avec des intégrales de surfaces $\underline{\underline{P_k}}$ et $\underline{\underline{M'}}$. Pour explorer les possibilités, la condition aux limites, jusque là réflective est remplacée par une condition de vide. Les flux sont représentés à la figure 4.17. Les oscillations disparaissent, cependant la discontinuité reste large pour le cas de segment oblique comparé au segment droit.

Par pure curiosité des conditions aux limites mixtes ont été imposées. Il est observé que les oscillations apparaissent seulement si les deux faces 3 et 4 ont simultanément une condition réflective (figures 4.19 et 4.18). Ce sont les faces du rectangle qui sont coupées par le segment.

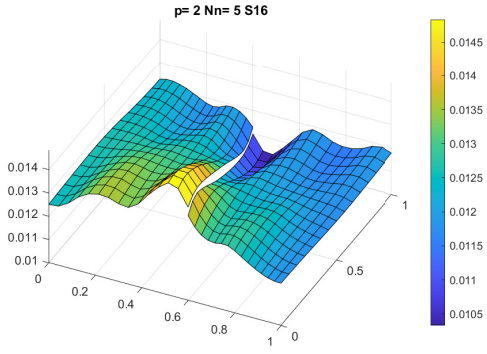


(a) Géométrie avec segment oblique

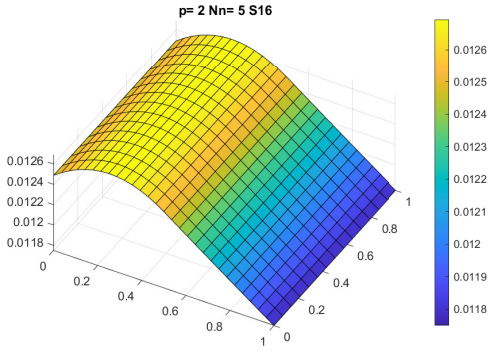


(b) Géométrie avec segment droit

Figure 4.17 Flux intégré pour la géométrie avec un segment de séparation oblique et droit, condition aux limites vide



(a) Segment oblique - Faces 3,4 réflectives



(b) Segment droit - Faces 3,4 réflectives

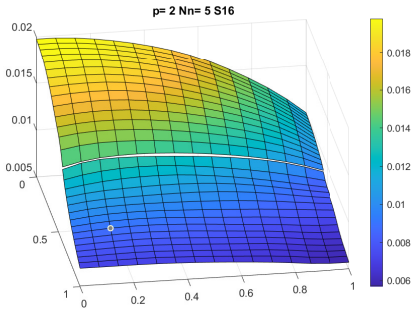
Figure 4.18 Flux intégré, géométrie avec un segment de séparation oblique ou droit, condition aux limites réflective sur les faces 3 et 4

Malheureusement, l'origine de la défaillance a été repérée, mais le problème demeure non résolu. Les conditions nécessaires pour l'apparition des oscillations sont résumées ci-dessous :

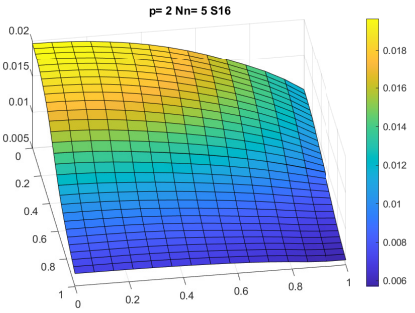
— Existence d'une interface non parallèle aux axes cartésiennes.

— Existence de condition réflective sur les faces perpendiculaires (dans l'espace paramétrique) à l'interface problématique.

La résolution de ce problème conduirait à revoir l'ensemble des fonctions MATLAB générant les matrices à intégrales de surface parce qu'elles partagent la même logique de construction.



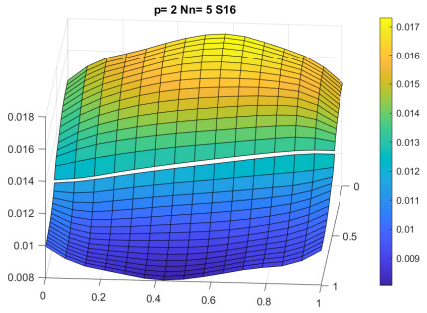
(a) Segment oblique - Faces 1,3 réflectives



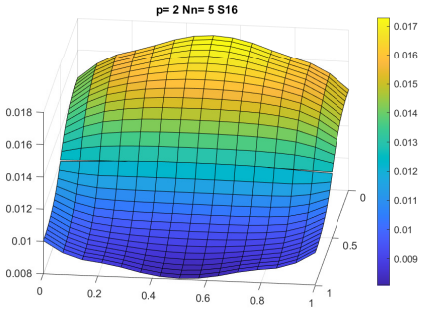
(b) Segment droit - Faces 1,3 réflectives

!htb

!htB



(c) Segment oblique - Faces 1,2 réflectives



(d) Segment droit - Faces 1,2 réflectives

Figure 4.19 Les conditions aux limites mixtes n'introduisant pas d'oscillations

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

Dans ce mémoire un prototype MATLAB est développé pour appliquer la méthode IGA à la résolution de la BTE. Ses résultats ont été comparés à ceux de DRAGON pour divers problèmes et géométrie.
La documentation sur la méthode est fournie de manière exhaustive, allant de la nature des NURBS aux équations théoriques et jusqu'à l'implémentation algorithmique.
Malheureusement, l'objectif de veille technologique n'est que partiellement accomplie. La précision de nos résultats n'est pas à la hauteur de celle présentée dans la littérature. Toutefois, la capacité de la méthode IGA à résoudre la BTE sur une géométrie non cartésienne est vérifiée.
Bien que non résolue, la faille dans la conception du prototype semble avoir été identifiée et mise à disposition pour étude future.

5.2 Limitations et améliorations futures

Pour le prototype réalisé, il y a 3 aspects limitants et donc il fournit 3 pistes d'améliorations possible :

1. Outre la précision, le prototype développé est encore limité par sa vitesse de calcul. L'optimisation du schéma de calcul est bien documentée dans la littérature, et un cas d'application réussie est observé. Elle n'est pas étudiée dans le mémoire, car la priorité est accordée à la précision. Mais l'optimisation est la plus importante parmi le reste. Cela rendrait la méthode IGA plus compétitive vis-à-vis des autres méthodes de résolution.
2. Le prototype ne fait pas encore le lien avec la CAD. En effet, les géométries étudiées sont conçues en respectant les règles imposées. Mais il en est pas de même si la méthode doit être utilisée à l'échelle industrielle. Les règles sont trop restrictives et limitent la liberté de la conception. Il faudrait ajouter une fonctionnalité au prototype, pour qu'il puisse automatiquement diviser, via l'insertion de points de contrôle et de poids, une géométrie NURBS brute et intégrale, en plusieurs patches respectant les règles d'analyse.
3. La capacité du prototype est restreinte. Ce dernier point mérite plus de développement.

Pour le prototype actuel, et de même dans la littérature consultée, l'algorithme travaille avec une géométrie fixe. Alors que les géométries NURBS sont très flexibles et maniables. Il a été question d'étudier le vecteur nodal et le degré des NURBS dans le mémoire, faut il encore porter l'attention sur les deux autres paramètres définissant les NURBS : les poids et les points de contrôle. En CAD, le design d'une courbure NURBS est réalisé en fixant le nombre de points et en les déplaçant.(donc changer les coordonnées et le poids correspondant). Or comme il a été vu, une modification locale des paramètres conduit à des effets locaux. La méthode IGA permettrait de travailler facilement avec une géométrie dynamique. En effet, si une équation temporelle aurait été étudiée et que la géométrie devrait répondre aux équations, les petites modifications de géométries conduiraient seulement à des modifications locales dans les matrices ! Il suffirait de recalculer les éléments touchés par la modification de géométrie et ceci représente un gain de temps de calcul non négligeable. Cette piste est moins intéressante en génie nucléaire, mais pourrait être plus utile dans d'autre domaine, comme la mécanique des fluides.

RÉFÉRENCES

[1] G. Marleau, A. Hebert et R. Roy, "A USER GUIDE FOR DRAGON VERSION5," Institut de génie nucléaire, Ecole Polytechnique de Montréal, Rapport technique IGE-335, 2022.

[2] A. A. Calloo, "Parallel Discontinuous Finite Element SN Solver in Cartesian and Hexagonal Geometries for the Boltzmann Transport Equation in DRAGON5," Thèse de doctorat, Ecole Polytechnique de Montréal, 2023.

[3] L. Piegl et W. Tiller, *The NURBS Book*, 2e éd. Springer, 1997.

[4] T. Hughes, J. Cottrell et Y. Bazilevs, "Isogeometric analysis : CAD, finite elements, NURBS, exact geometry and mesh refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 39-41, p. 4135–4195, oct. 2005. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0045782504005171

[5] J.-Y. Moller, "Eléments finis courbes et accélération pour le transport de neutrons," Thèse de doctorat, Université Henri Poincaré, Nancy, 2012.

[6] A. Owens, "Discontinuous Isogeometric Analysis Methods for the First Order Form of the Neutron Transport Equation with Discrete Ordinate Angular Discretisation," Thèse de doctorat, Imperial College London, 2017.

[7] R. Horita, W. F. G. Van Rooijen et D. Lathouwers, "Study of neutron transport calculation using isogeometric analysis method," dans *PHYTRA4 - The Fourth International Conference on Physics and Technology of Reactors and Applications*, Marrakech, Morocco, sept. 2018.

[8] C. De Boor, "On Calculating with B-Splines," *Journal of approximation theory*, p. 50–62, 1970.

[9] A.-V. Vuong, C. Heinrich et B. Simeon, "ISOGAT : A 2D tutorial MATLAB code for Isogeometric Analysis," *Computer Aided Geometric Design*, vol. 27, no. 8, p. 644–655, nov. 2010. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0167839610000713

[10] A. Hébert, *Applied reactor physics*, 3e éd. Montreal : Presses Internationales Polytechnique, 2020, CLC : 1036192351.

[11] E. W. Larsen, "International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (MC 2011)," *Transport Theory and Statistical Physics*, vol. 40, no. 4, p. 225–242, oct. 2011. [En ligne]. Disponible : http://www.tandfonline.com/doi/abs/10.1080/00411450.2011.610205

[12] R. Vázquez, “A new design for the implementation of isogeometric analysis in Octave and Matlab : GeoPDEs 3.0,” *Computers & Mathematics with Applications*, vol. 72, n°. 3, p. 523–554, août 2016. [En ligne]. Disponible : https://linkinghub.elsevier.com/retrieve/pii/S0898122116302681

[13] R. Horita, “IGA 法を用いた 中性子輸送計算手法の研究,” Thèse de doctorat, Université Fukui, Fukui, 2019.

ANNEXE A CALCUL DES FONCTIONS B-SPLINES

Les calculs des fonctions B-splines.

A.1 Les B-splines pour l'arc de cercle

Rappelons les paramètres de la courbe NURBS :

Degré	p = 2
Vecteur nodal	$U = \{0, 0, 0, 0.5, 0.5, 1, 1, 1\}$
Points de contrôle	$\{\mathbf{P}_i\} = \left\{ \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\}$
Poids	$\{\omega_i\} = \left\{1, \frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}, 1\right\}$

En utilisant l'équation 3.1 :

$$N_{2,0}(u) = 1 \text{ si } u \in [0, 0.5[\tag{A.1a}$$
$$N_{4,0}(u) = 1 \text{ si } u \in [0.5, 1[\tag{A.1b}$$
$$N_{i,0}(u) = 0 \text{ si } i \neq 2 \text{ et } i \neq 4 \tag{A.1c}$$

$$N_{1,1}(u) = \frac{u - u_1}{u_2 - u_1} N_{1,0}(u) + \frac{u_3 - u}{u_3 - u_2} N_{2,0}(u) = 0 + \frac{0.5 - u}{0.5} \quad \text{si } u \in [0, 0.5[\tag{A.2a}$$
$$N_{2,1}(u) = \frac{u - u_2}{u_3 - u_2} N_{2,0}(u) + \frac{u_4 - u}{u_4 - u_3} N_{3,0}(u) = \frac{u - 0}{0.5 - 0} + 0 \quad \text{si } u \in [0, 0.5[\tag{A.2b}$$
$$N_{3,1}(u) = \frac{u - u_3}{u_4 - u_3} N_{3,0}(u) + \frac{u_5 - u}{u_5 - u_4} N_{4,0}(u) = 0 + \frac{1 - u}{0.5} \quad \text{si } u \in [0.5, 1[\tag{A.2c}$$
$$N_{4,1}(u) = \frac{u - u_4}{u_5 - u_4} N_{4,0}(u) + \frac{u_6 - u}{u_6 - u_5} N_{5,0}(u) = \frac{u - 0.5}{0.5} + 0 \quad \text{si } u \in [0.5, 1[\tag{A.2d}$$
$$N_{i,1}(u) = 0 \text{ pour les autres i} \tag{A.2e}$$

$$N_{0,2}(u) = \frac{u-u_0}{u_2-u_0}N_{0,1}(u) + \frac{u_3-u}{u_3-u_1}N_{1,1}(u) = 0 + \left(\frac{0.5-u}{0.5}\right)^2 \quad \text{si } u \in [0, 0.5[\tag{A.3a}$$

$$N_{1,2}(u) = \frac{u-u_1}{u_3-u_1}N_{1,1}(u) + \frac{u_4-u}{u_4-u_2}N_{2,1}(u) = 2\frac{u(0.5-u)}{0.5^2} \quad \text{si } u \in [0, 0.5[\tag{A.3b}$$

$$N_{2,2}(u) = \frac{u-u_2}{u_4-u_2}N_{2,1}(u) + \frac{u_5-u}{u_5-u_3}N_{3,1}(u) = \left(\frac{u}{0.5}\right)^2 \quad \text{si } u \in [0, 0.5[\tag{A.3c}$$

$$N_{2,2}(u) = \left(\frac{1-u}{0.5}\right)^2 \quad \text{si } u \in [0.5, 1[\tag{A.3d}$$

$$N_{3,2}(u) = \frac{u-u_3}{u_5-u_3}N_{3,1}(u) + \frac{u_6-u}{u_6-u_4}N_{4,1}(u) = 2\frac{(u-0.5)(1-u)}{0.5^2} \quad \text{si } u \in [0.5, 1[\tag{A.3e}$$

$$N_{4,2}(u) = \frac{u-u_4}{u_6-u_4}N_{4,1}(u) + \frac{u_7-u}{u_7-u_5}N_{5,1}(u) = \left(\frac{u-0.5}{0.5}\right)^2 \quad \text{si } u \in [0.5, 1[\tag{A.3f}$$

$$N_{i,2}(u) = 0 \text{ pour les autres i} \tag{A.3g}$$

Puis on calcule la somme :

$$\sum_{j=0}^{4} N_{j,2}(u)\omega_j = \begin{cases} \left(\frac{0.5-u}{0.5}\right)^2 + \frac{2u(0.5-u)}{0.5^2*\sqrt{2}} + \left(\frac{u}{0.5}\right)^2 & \text{si } u \in [0, 0.5[\\ \left(\frac{1-u}{0.5}\right)^2 + \frac{2(u-0.5)(1-u)}{0.5^2*\sqrt{2}} + \left(\frac{u-0.5}{0.5}\right)^2 & \text{si } u \in [0.5, 1[\end{cases} \tag{A.4}$$

Et on obtient les fonctions NURBS avec l'équation 3.2.

A.2 Les B-splines pour le patch du Pincell

Les paramètres du patch :

Tableau A.1 L'ensemble des paramètres du patch

Paramtres	
Vecteurs nodaux U, V	$\{0,0,0,1,1,1\}$ et $\{0,0,0,1,1,1\}$
Poids	$\omega_4 = 0.8535533$ $\omega_7 = \frac{1}{\sqrt{2}}$, les autres sont égaux à 1
Points de contrôle	$\begin{pmatrix}0.4725\\0.7896\end{pmatrix}, \begin{pmatrix}0.6310\\0.7896\end{pmatrix}, \begin{pmatrix}0.7896\\0.7896\end{pmatrix}, \begin{pmatrix}0.4104\\0.8517\end{pmatrix}, \begin{pmatrix}0.6310\\0.9582\end{pmatrix}, \begin{pmatrix}0.8517\\0.8517\end{pmatrix}, \begin{pmatrix}0.3482\\0.9139\end{pmatrix}, \begin{pmatrix}0.6310\\1.1967\end{pmatrix}, \begin{pmatrix}0.9139\\0.9139\end{pmatrix}$

De la même façon que la section précédente, on calcule successivement les $N_{i,p}$, il n'est pas nécessaire de préciser le domaine de définition, car il n'y pas de noeud autre 0 et 1. On calcule

les fonctions B-spline pour une direction, celles de l'autre direction sont identiques.

$$N_{2,0}(u) = 1 \tag{A.5a}$$

$$N_{1,1}(u) = 1 - u \tag{A.6a}$$
$$N_{2,1}(u) = u \tag{A.6b}$$

$$N_{0,2}(u) = (1-u)^2 \tag{A.7a}$$
$$N_{1,2}(u) = 2u(1-u) \tag{A.7b}$$
$$N_{2,2}(u) = u^2 \tag{A.7c}$$

Puis on calcule les produits tensoriels des fonctions B-splines :

$$\tilde{N}_{0,2}(u,v) = (1-u)^2(1-v)^2 \tag{A.8a}$$
$$\tilde{N}_{1,2}(u,v) = 2u(1-u)(1-v)^2 \tag{A.8b}$$
$$\tilde{N}_{2,2}(u,v) = u^2(1-v)^2 \tag{A.8c}$$
$$\tilde{N}_{3,2}(u,v) = (1-u)^2 2v(1-v) \tag{A.8d}$$
$$\tilde{N}_{4,2}(u,v) = 2u(1-u)2v(1-v) \tag{A.8e}$$
$$\tilde{N}_{5,2}(u,v) = u^2 2v(1-v) \tag{A.8f}$$
$$\tilde{N}_{6,2}(u,v) = (1-u)^2 v2 \tag{A.8g}$$
$$\tilde{N}_{7,2}(u,v) = 2u(1-u)v^2 \tag{A.8h}$$
$$\tilde{N}_{8,2}(u,v) = u^2 v^2 \tag{A.8i}$$

Puis on calcule la somme :

$$S = \sum_{j=0}^{8} \tilde{N}_{j,2}\omega_j = 1 - 2u(1-u)\left[2v(1-v)(1-\omega_5) + v^2(1-\omega_8)\right] \tag{A.9}$$

Et on obtient les fonctions NURBS :

$$R_{i,2}(u,v) = \frac{\tilde{N}_{i,2}(u,v)\omega_i}{S} \tag{A.10}$$

ANNEXE B GRAPHES DU COLORSET 1

Les graphes du Colorset 1.

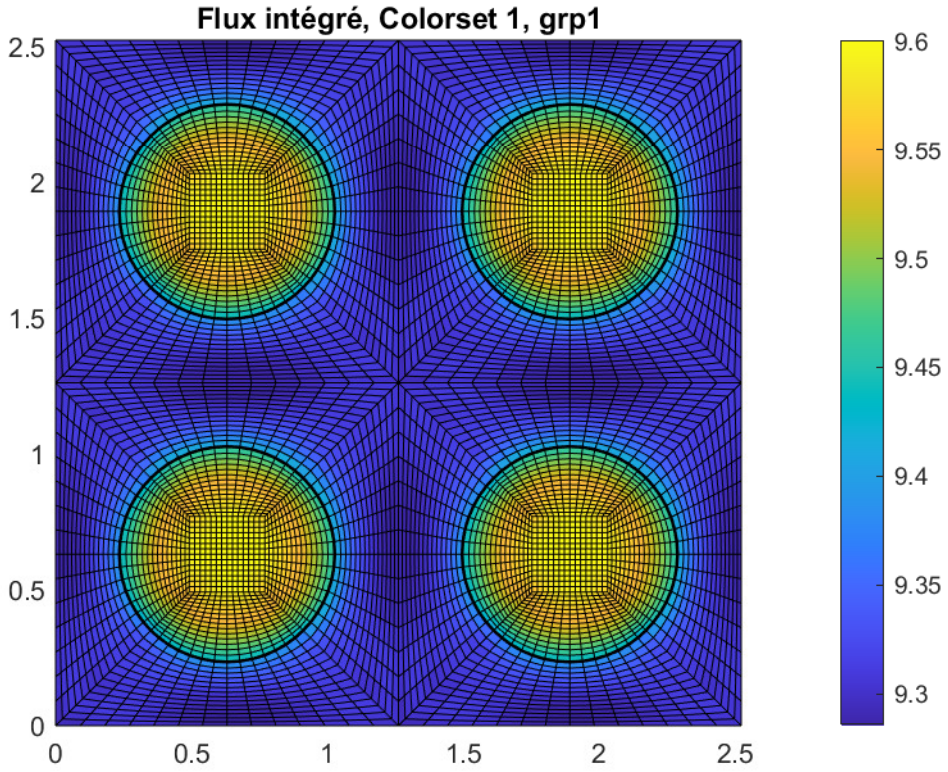


Figure B.1 Flux intégré, Colorset 1 grp 1

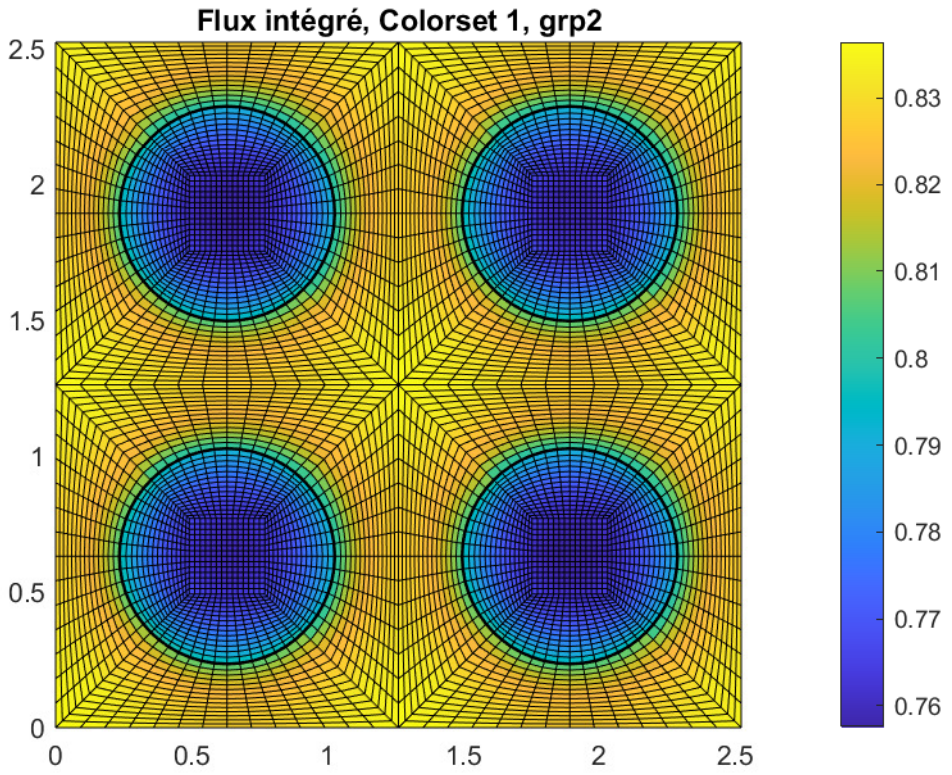


Figure B.2 Flux intégré, Colorset 1 grp 2

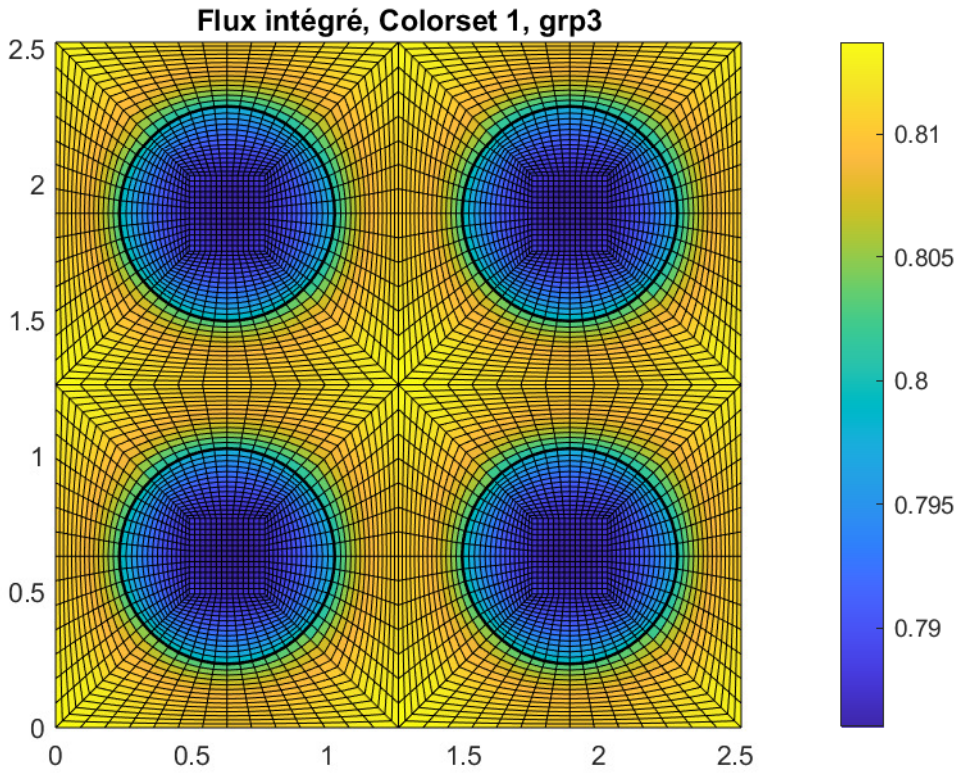


Figure B.3 Flux intégré, Colorset 1 grp 3

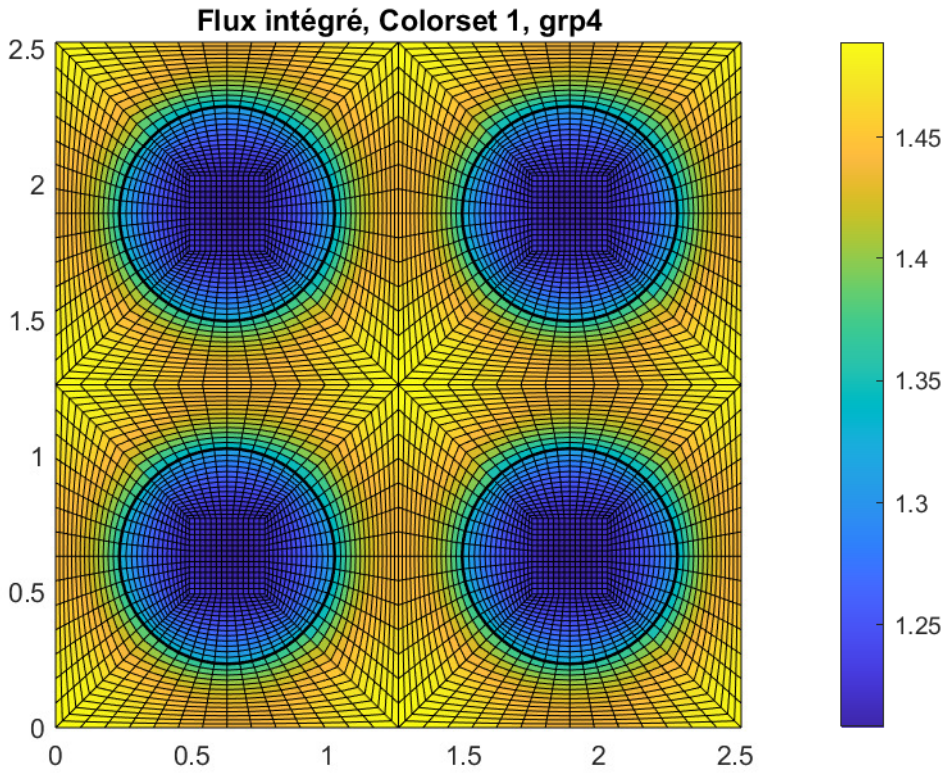


Figure B.4 Flux intégré, Colorset 1 grp 4

ANNEXE C GRAPHES DU COLORSET 2

Les graphes du Colorset 2.

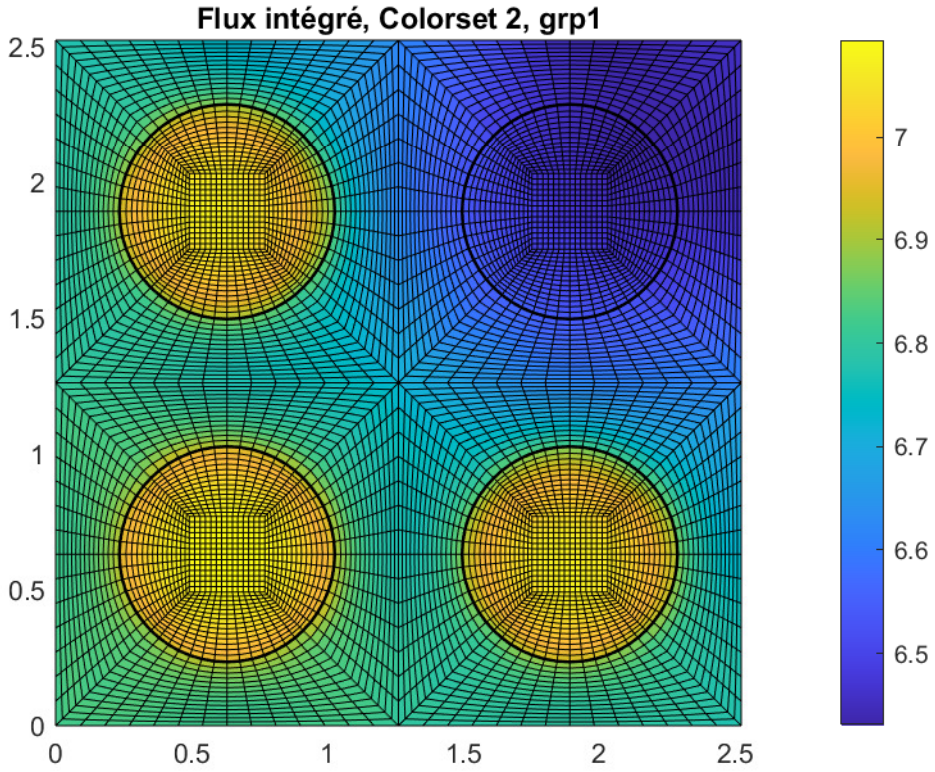


Figure C.1 Flux intégré, Colorset 2 grp 1

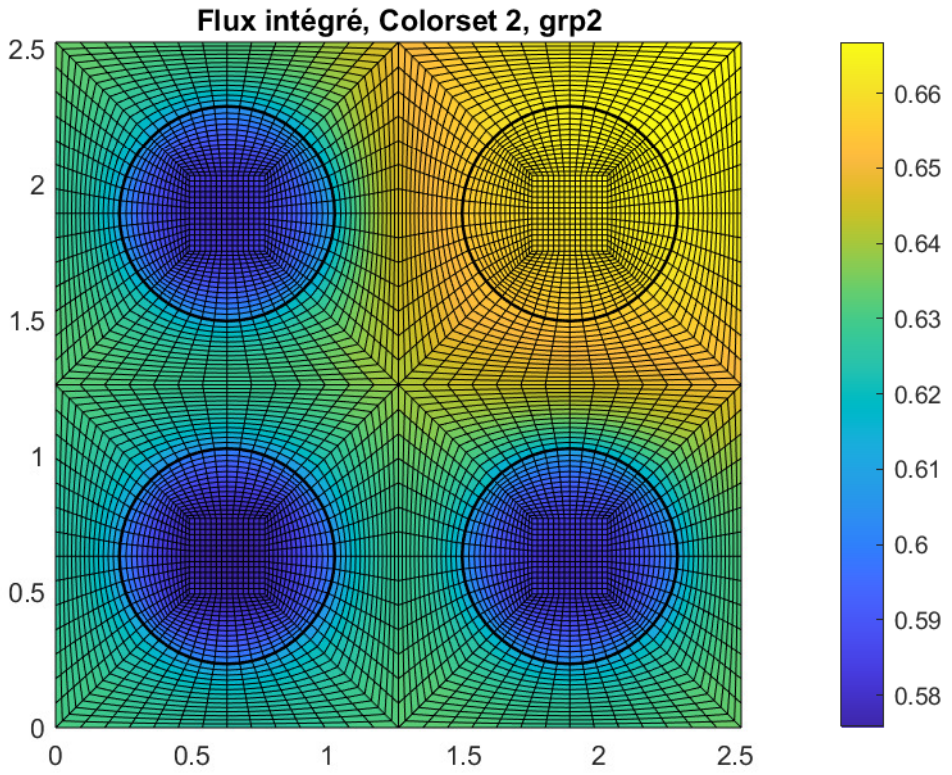


Figure C.2 Flux intégré, Colorset 2 grp 2

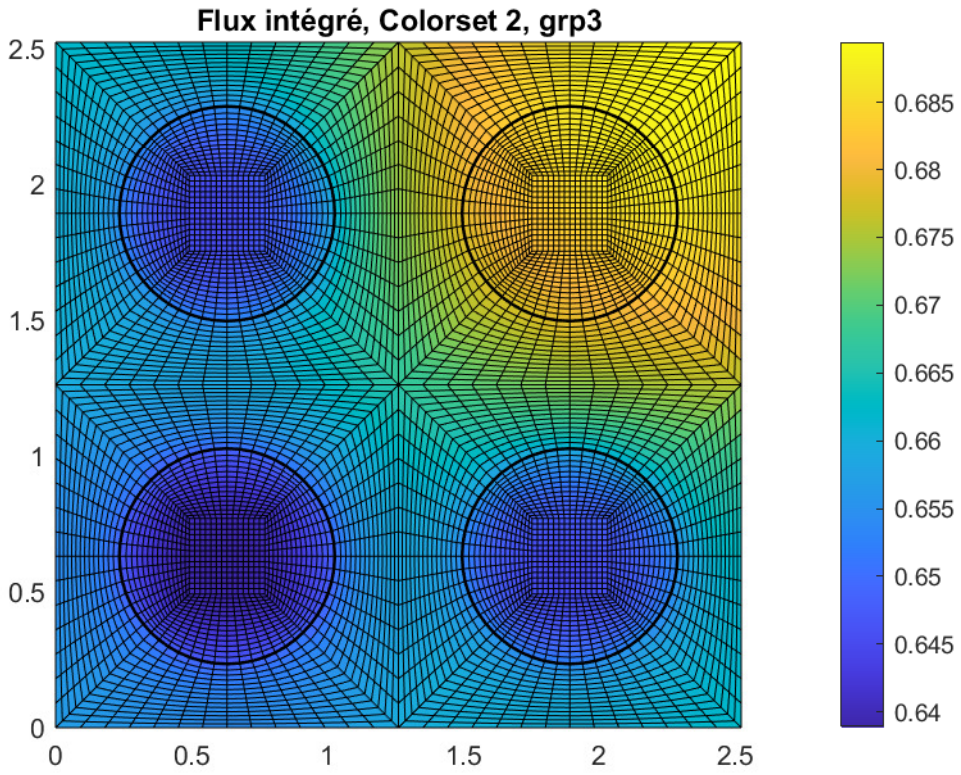


Figure C.3 Flux intégré, Colorset 2 grp 3

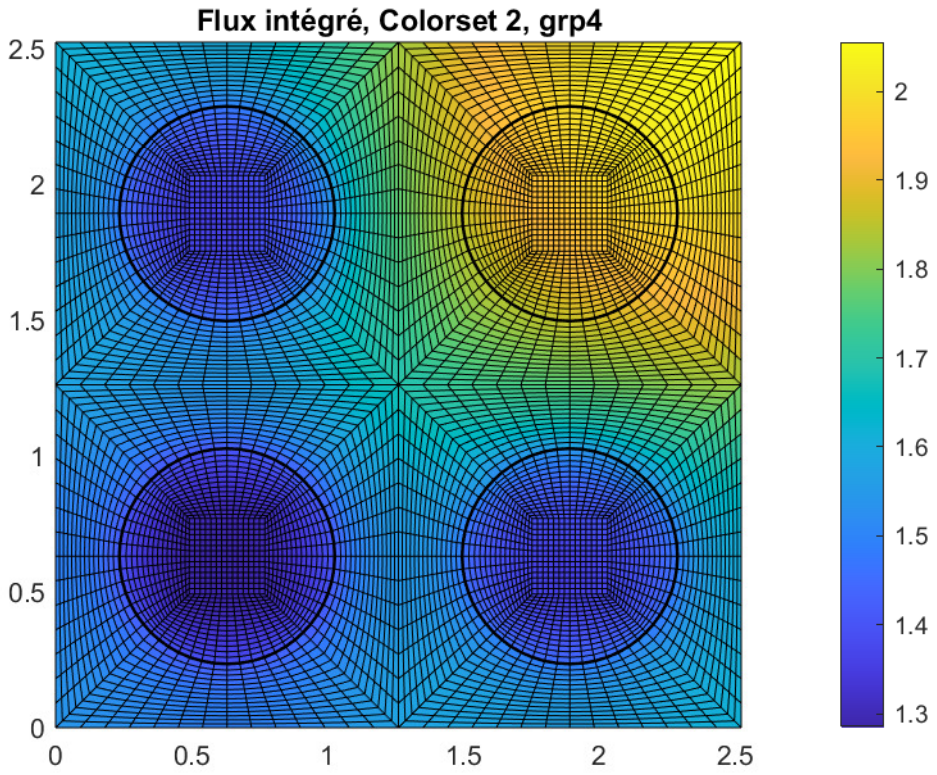


Figure C.4 Flux intégré, Colorset 2 grp 4

ANNEXE D GRAPHES DU COLORSET 3

Les graphes du Colorset 3.

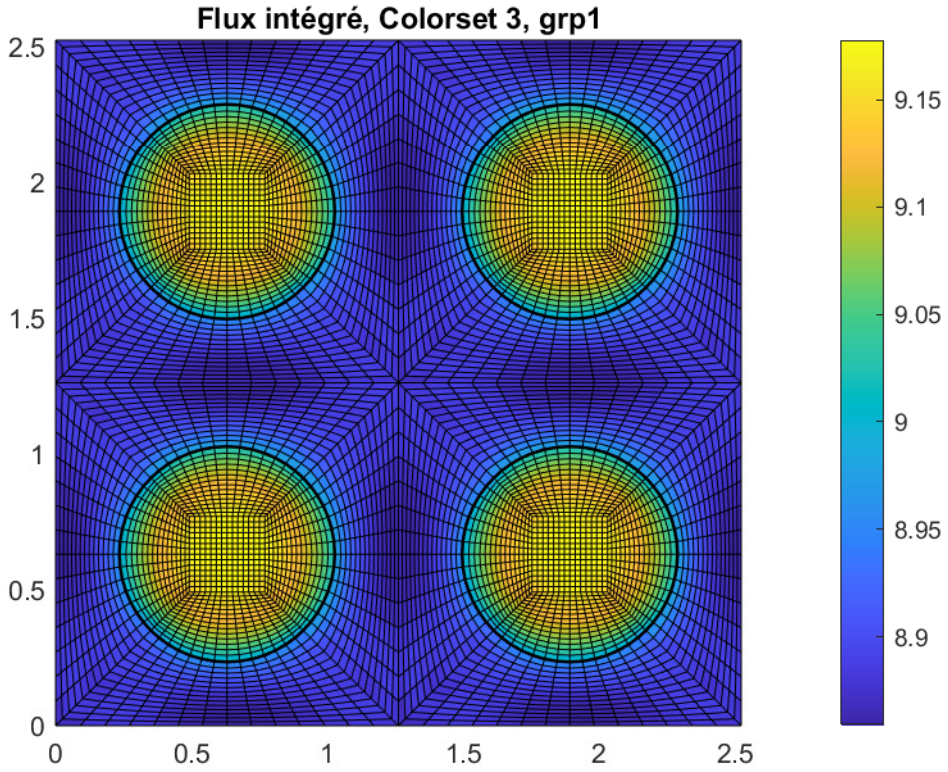


Figure D.1 Flux intégré, Colorset 3 grp 1

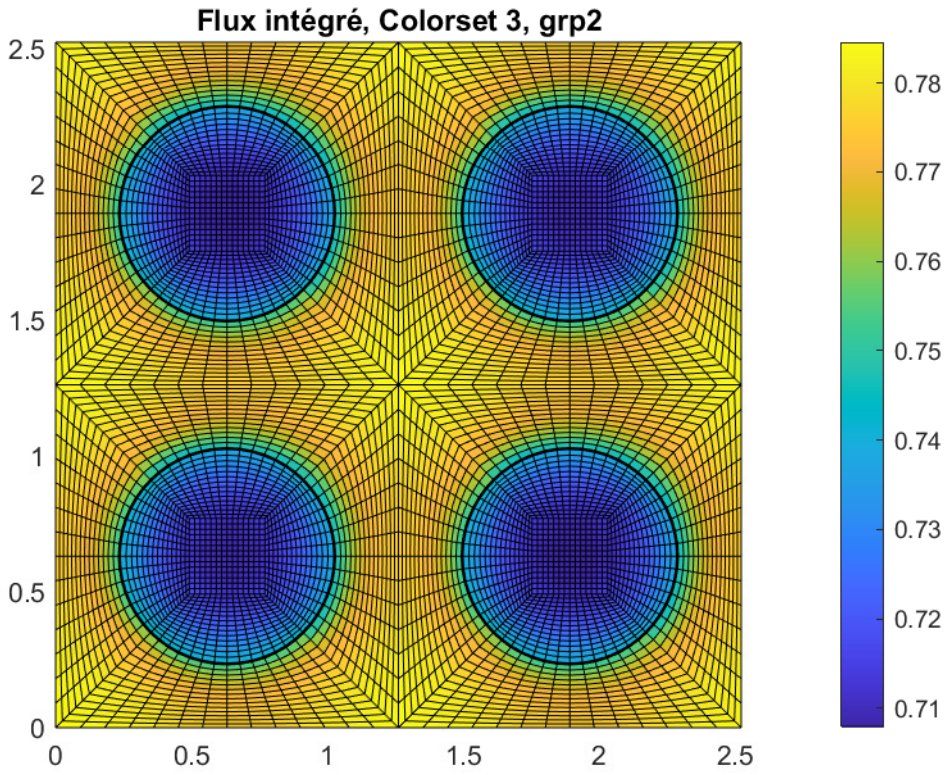


Figure D.2 Flux intégré, Colorset 3 grp 2

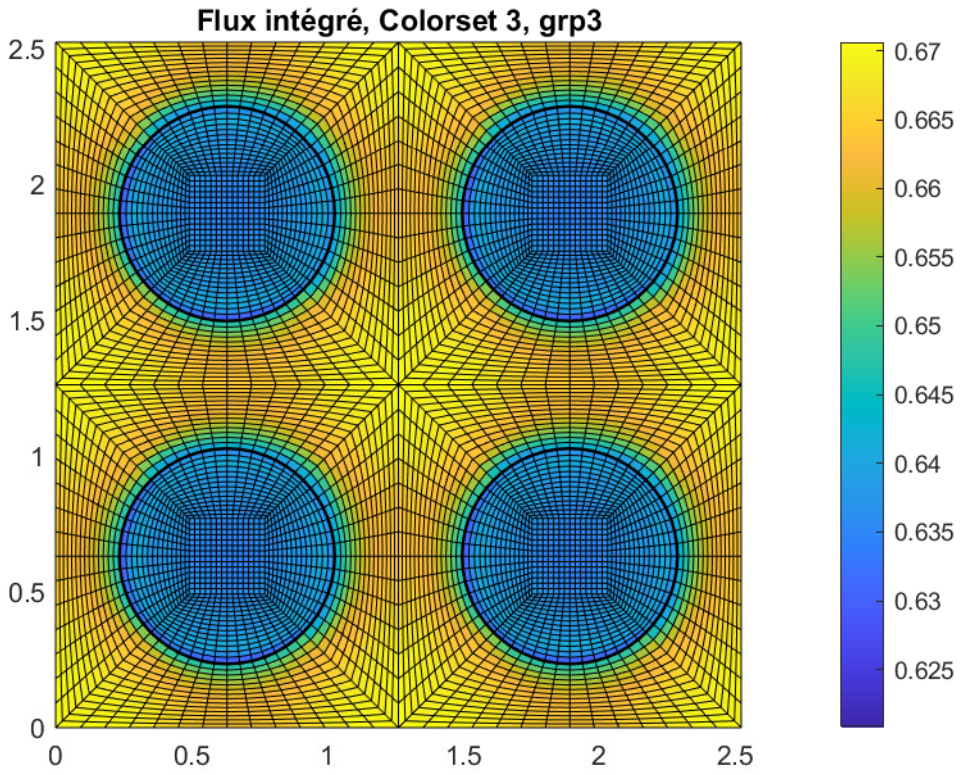


Figure D.3 Flux intégré, Colorset 3 grp 3

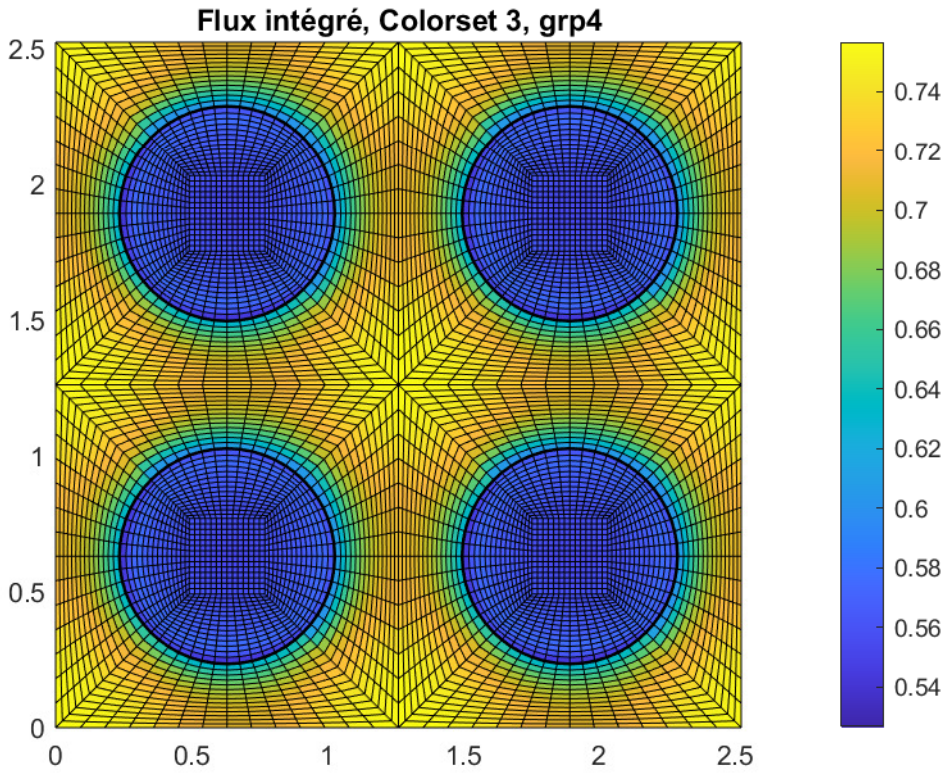


Figure D.4 Flux intégré, Colorset 3 grp 4

ANNEXE E SCRIPTS MATLAB

Les scripts MATLAB utilisés pour construire le prototype sont fournis dans cette annexe.

E.1 PF_knotins_degelev

```
function nrb_m = PF_knotins_degelev (nrb,knot_ins,deg_elev)
    nrb_m = nrb;
    n_knt = linspace (0,1,2+knot_ins);
    n_knt = n_knt(2:end-1);

    max_order = 0;
    for i = 1:numel(nrb_m)
        max_order = max(max_order,max(nrb_m(i).order));
    end

    max_order = max_order + deg_elev;

    for i=1:numel(nrb_m)
        if min(nrb_m(i).order)<max_order
            nrb_m(i) = nrbdegelev(nrb_m(i),[max_order-nrb_m(i).order(1) max_order-nrb_m(
               i).order(2)]);
        end
        nrb_m(i) = nrbkntins(nrb_m(i),{n_knt,n_knt});
    end
```

```
end
```

E.2 PF_Matrix_v2

```
function [M,S_1,S_2]=PF_Matrix_v2(mp_sp,mp_msh)
    n_patch = mp_msh.npatch;
    M = cell(n_patch,1);
    S_1 = cell(n_patch,1);
    S_2 = cell(n_patch,1);

    for i_patch = 1:n_patch
        spaced = sp_precompute(mp_sp.sp_patch{i_patch},mp_msh.msh_patch{i_patch},'
            gradient',true);
        mshed = msh_precompute(mp_msh.msh_patch{i_patch});
        spu = spaced;
        spv = spaced;
        M{i_patch} = spalloc(spv.ndof,spu.ndof,1);
        S_1{i_patch} = spalloc(spv.ndof,spu.ndof,1);
        S_2{i_patch} = spalloc(spv.ndof,spu.ndof,1);
        shpu = reshape (spu.shape_functions, spu.ncomp, mshed.nqn, spu.nsh_max, mshed.
            nel);
        shpv = reshape (spv.shape_functions, spv.ncomp, mshed.nqn, spv.nsh_max, mshed.
            nel);
        gradv = reshape (spv.shape_function_gradients, spv.ncomp, [], mshed.nqn, spv.
            nsh_max, mshed.nel);
```

```
18
19          for iel = 1:mshed.nel
20              jacdet_weights = mshed.jacdet(:,iel).*mshed.quad_weights(:,iel);
21              mat_loc_M = zeros(spv.nsh(iel),spu.nsh(iel));
22              mat_loc_S_1 = zeros(spv.nsh(iel),spu.nsh(iel));
23              mat_loc_S_2 = zeros(spv.nsh(iel),spu.nsh(iel));
24
25              for idof = 1:spv.nsh(iel)
26                  ishp = reshape (shpv(:,:,idof,iel), spv.ncomp, []);
27                  igrad_1 = reshape (gradv(:,1,:,idof,iel), spv.ncomp,[]);
28                  igrad_2 = reshape (gradv(:,2,:,idof,iel), spv.ncomp,[]);
29                  for jdof = 1:spu.nsh(iel)
30                      jshp = reshape (shpu(:,:,jdof,iel), spu.ncomp, []);
31                      mat_loc_M(idof, jdof) = mat_loc_M(idof, jdof) + ...
32                          sum ( jacdet_weights .* sum(ishp .* jshp, 1).');
33                      mat_loc_S_1 (idof,jdof) = mat_loc_S_1(idof,jdof) + ...
34                          sum(jacdet_weights .* sum(igrad_1.*jshp, 1).');
35                      mat_loc_S_2 (idof,jdof) = mat_loc_S_2(idof,jdof) + ...
36                          sum(jacdet_weights .* sum(igrad_2.*jshp, 1).');
37                  end
38              end
39              M{i_patch}(spv.connectivity(:, iel), spu.connectivity(:, iel)) = ...
40                  M{i_patch}(spv.connectivity(:, iel), spu.connectivity(:, iel)) +
                       mat_loc_M;
41              S_1{i_patch}(spv.connectivity(:,iel),spu.connectivity(:,iel))= ...
42                  S_1{i_patch}(spv.connectivity(:,iel),spu.connectivity(:,iel)) +
```

```
                    mat_loc_S_1;
            S_2{i_patch}(spv.connectivity(:,iel),spu.connectivity(:,iel))= ...
                S_2{i_patch}(spv.connectivity(:,iel),spu.connectivity(:,iel)) +
                    mat_loc_S_2;
        end
    end
end
```

E.3 PF_MatrixP_v3

```
function Pk = PF_MatrixP_v3 (mp_sp,bnds_s, bnds_m, OMEGA)
n_angle = numel(OMEGA);
n_patch = size(bnds_m,1);
Pk = cell(n_patch,n_angle);

for i_angle = 1:n_angle
    for i_patch = 1:n_patch
        Pk{i_patch,i_angle} = spalloc(mp_sp.sp_patch{i_patch}.ndof,mp_sp.sp_patch{
            i_patch}.ndof,1);
        for i_side = 1:4
            mshed = bnds_m{i_patch,i_side};
            spaced = bnds_s{i_patch,i_side};

            normal = mshed.normal;
            omg_n =sum(  normal.*reshape(OMEGA{i_angle}(1:2),2,1) , 1);
            if not(all(all(omg_n<0)))
```

```
spu = spaced; spv = spaced;
shpu = reshape (spu.shape_functions, spu.ncomp, mshed.nqn, spu.nsh_max,
   mshed.nel);
shpv = reshape (spv.shape_functions, spv.ncomp, mshed.nqn, spv.nsh_max,
   mshed.nel);
Pk_local = spalloc(spv.ndof, spu.ndof,1);

for iel = 1:mshed.nel
    jacdet_weights= mshed.jacdet(:, iel) .* mshed.quad_weights(:, iel);
    Pk_local_elem= zeros (spv.nsh(iel), spu.nsh(iel));
    omg_n_loc = omg_n(:,:,iel);
    for idof = 1:spv.nsh(iel)
        ishp = reshape(shpv(:,:,idof,iel), spv.ncomp, []);
        for jdof = 1:spu.nsh(iel)
            jshp= reshape (shpu(:,:,jdof,iel), spu.ncomp, []);
            for inqn = 1:mshed.nqn
                if omg_n_loc(inqn)>0
                    Pk_local_elem(idof, jdof) = Pk_local_elem(idof, jdof
                       )+...
                        omg_n_loc(inqn)*jacdet_weights(inqn)*ishp(inqn)*
                           jshp(inqn);
                end
            end
        end
    end
```

```
                    Pk_local(spv.connectivity(:,iel),spu.connectivity(:,iel)) = Pk_local
                       (spv.connectivity(:,iel),spu.connectivity(:,iel)) + Pk_local_elem
                       ;
                end
                Pk{i_patch,i_angle}(spv.dofs,spu.dofs) = Pk{i_patch,i_angle}(spv.dofs,
                   spu.dofs) + Pk_local;
            end

        end
    end
end
end
```

E.4 PF_Phi_adj_ppp_v4

```
function [Phi_bc,Phi_in,bnd_angle_sortant,in_voisin] = PF_Phi_adj_ppp_v4 ( mp_sp, mp_msh
   , boundaries, interface, OMEGA, bnds_m,bnds_s,bc )
n_patch = mp_msh.npatch;
n_angle = numel(OMEGA);

Phi_bc = cell(n_patch,n_angle,4);
Phi_in = cell(n_patch,n_angle,4);
bnd_angle_sortant = cell(n_patch, n_angle, 4);
in_voisin = cell(n_patch,n_angle,4);


```

```
% initialisation de Phi_bc et Phi_in
for i_patch_init =1:n_patch
    ndof = mp_sp.sp_patch{i_patch_init}.ndof;
    for i_angle_init =1:n_angle
        for i_side_init =1:4
            Phi_bc{i_patch_init,i_angle_init,i_side_init} = spalloc(ndof,ndof,1);
            Phi_in{i_patch_init,i_angle_init,i_side_init} = spalloc(ndof,ndof,1);
        end
    end
end

% calculation
for i_angle_cal = 1:n_angle

    for i_patch_cal = 1:n_patch

        for i_side_cal = 1:4

            % determiner si c'est une bordure
            bnd_mshed = bnds_m{i_patch_cal,i_side_cal};
            bnd_spaced= bnds_s{i_patch_cal,i_side_cal};
            normal = bnd_mshed.normal;
            omg_n =sum(  normal.*reshape(OMEGA{i_angle_cal}(1:2),2,1) , 1);

            if not(all(all(omg_n>=0))) % une surface entrante pour l'angle

```

```
37                  % determiner la nature de la frontiere
38                  Is_bnd = false;
39                  for bnd = boundaries
40                      if bnd.patches == i_patch_cal && bnd.faces == i_side_cal
41                          Is_bnd = true;
42                          break;
43                      end
44                  end
45
46                  % cas d'une bordure
47                  if Is_bnd
48                      Phi_bc_loc= spalloc(bnd_spaced.ndof,bnd_spaced.ndof,1);
49                      switch bc
50                          case 'void'
51
52                          case 'reflective'
53                              spu=bnd_spaced;
54                              spv=bnd_spaced;
55                              shpu = reshape (spu.shape_functions, spu.ncomp, bnd_mshed.
                                    nqn, spu.nsh_max, bnd_mshed.nel);
56                              shpv = reshape (spv.shape_functions, spv.ncomp, bnd_mshed.
                                    nqn, spv.nsh_max, bnd_mshed.nel);
57                              % determiner i_angle_sortant
58                              switch i_side_cal
59                                  case 1
60                                      for j_angle = 1:n_angle
```

```
61                          if OMEGA{j_angle}(1) == -OMEGA{i_angle_cal}(1)
                               && OMEGA{j_angle}(2) == OMEGA{i_angle_cal}(2)
62                              i_angle_sortant = j_angle;
63                              break;
64                          end
65                      end
66                  case 2
67                      for j_angle = 1:n_angle
68                          if OMEGA{j_angle}(1) == -OMEGA{i_angle_cal}(1)
                               && OMEGA{j_angle}(2) == OMEGA{i_angle_cal}(2)
69                              i_angle_sortant = j_angle;
70                              break;
71                          end
72                      end
73                  case 3
74                      for j_angle = 1:n_angle
75                          if OMEGA{j_angle}(1) == OMEGA{i_angle_cal}(1) &&
                                OMEGA{j_angle}(2) == -OMEGA{i_angle_cal}(2)
76                              i_angle_sortant = j_angle;
77                              break;
78                          end
79                      end
80                  case 4
81                      for j_angle = 1:n_angle
82                          if OMEGA{j_angle}(1) == OMEGA{i_angle_cal}(1) &&
                                OMEGA{j_angle}(2) == -OMEGA{i_angle_cal}(2)
```

```
                    i_angle_sortant = j_angle;
                    break;
                end
            end
        end

        bnd_angle_sortant{i_patch_cal,i_angle_cal,i_side_cal} =
           i_angle_sortant;

        % calcul
        for iel =1:bnd_mshed.nel
            jacdet_weights= bnd_mshed.jacdet(:, iel) .* bnd_mshed.
               quad_weights(:, iel);
            mat_loc_mass= zeros (spv.nsh(iel), spu.nsh(iel));
            omg_n_loc = omg_n(:,:,iel);

            for idof = 1:spv.nsh(iel)
                ishp = reshape (shpv(:,:,idof,iel), spv.ncomp, []);
                for jdof = 1:spu.nsh(iel)
                    jshp = reshape (shpu(:,:,jdof,iel), spu.ncomp,
                       []);
                    for inqn =1:bnd_mshed.nqn
                        if omg_n_loc(:,inqn)<0
                            mat_loc_mass(idof, jdof) = mat_loc_mass(
                               idof, jdof)+...
```

```
104                                                 omg_n_loc(:,inqn)*jacdet_weights(
                                                       inqn)*ishp(inqn)*jshp(inqn);
105                                         end
106                                     end
107                                 end
108                             end
109                             Phi_bc_loc(spv.connectivity(:, iel), spu.connectivity(:,
                                    iel)) = ...
110                                 Phi_bc_loc(spv.connectivity(:, iel), spu.
                                       connectivity(:, iel)) +  mat_loc_mass;
111
112                         end
113                         Phi_bc{i_patch_cal,i_angle_cal,i_side_cal}(bnd_spaced.dofs,
                               bnd_spaced.dofs) = Phi_bc{i_patch_cal,i_angle_cal,
                               i_side_cal}(bnd_spaced.dofs,bnd_spaced.dofs) + Phi_bc_loc
                               ;
114
115                 end
116             else % cas d'une interface
117
118                 % trouver voisin, format de in_voisin [patch, side,orientation]
119                 for int = interface
120                     if(int.patch1 == i_patch_cal) && (int.side1 == i_side_cal)
121                         in_voisin{i_patch_cal,i_angle_cal,i_side_cal} =[int.patch2,
                               int.side2, int.ornt];
122                         break;
```

```
123
124                     elseif (int.patch2 == i_patch_cal) && (int.side2 == i_side_cal)
125                         in_voisin{i_patch_cal,i_angle_cal,i_side_cal} =[int.patch1,
                               int.side1, int.ornt];
126                         break;
127                     end
128                 end
129
130                 % checker orientation
131
132                 spv=bnd_spaced;
133                 shpv = reshape (spv.shape_functions, spv.ncomp, bnd_mshed.nqn, spv.
                       nsh_max, bnd_mshed.nel);
134
135                 adj_patch = in_voisin{i_patch_cal,i_angle_cal,i_side_cal}(1);
136                 adj_side = in_voisin{i_patch_cal,i_angle_cal,i_side_cal}(2);
137                 ornt = in_voisin{i_patch_cal,i_angle_cal,i_side_cal}(3);
138
139                 adj_bnd_mshed = bnds_m{adj_patch,adj_side};
140                 adj_bnd_spaced= bnds_s{adj_patch,adj_side};
141
142                 Phi_in_loc = spalloc(bnd_spaced.ndof,adj_bnd_spaced.ndof,1);
143
144                 spu = adj_bnd_spaced;
145                 shpu = reshape (spu.shape_functions, spu.ncomp, adj_bnd_mshed.nqn,
                       spu.nsh_max, adj_bnd_mshed.nel);
```

```
146
147                 if ornt == 1
148                     for iel =1:bnd_mshed.nel
149                         jacdet_weights= bnd_mshed.jacdet(:, iel) .* bnd_mshed.
                               quad_weights(:, iel);
150                         mat_loc_mass= zeros (spv.nsh(iel), spu.nsh(iel));
151                         omg_n_loc = omg_n(:,:,iel);
152                         for idof = 1:spv.nsh(iel)
153                             ishp = reshape (shpv(:,:,idof,iel), spv.ncomp, []);
154                             for jdof = 1:spu.nsh(iel)
155                                 jshp = reshape (shpu(:,:,jdof,iel), spu.ncomp, []);
156                                 for inqn =1:bnd_mshed.nqn
157                                     if omg_n_loc(:,inqn)<0
158                                         mat_loc_mass(idof, jdof) = mat_loc_mass(idof
                                               , jdof)+...
159                                           omg_n_loc(:,inqn)*jacdet_weights(inqn)*
                                                 ishp(inqn)*jshp(inqn);
160                                     end
161                                 end
162                             end
163                         end
164                         Phi_in_loc(spv.connectivity(:, iel), spu.connectivity(:, iel
                               )) = ...
165                           Phi_in_loc(spv.connectivity(:, iel), spu.connectivity(:,
                                  iel)) +  mat_loc_mass;
166
```

```
167                     end
168                 else
169                     for iel =1:bnd_mshed.nel
170                         jacdet_weights= bnd_mshed.jacdet(:, iel) .* bnd_mshed.
                               quad_weights(:, iel);
171                         mat_loc_mass= zeros (spv.nsh(iel), spu.nsh(iel));
172                         omg_n_loc = omg_n(:,:,iel);
173                         for idof = 1:spv.nsh(iel)
174                             ishp = reshape (shpv(:,:,idof,iel), spv.ncomp, []);
175
176                             jel = bnd_mshed.nel+1-iel;
177
178                             for jdof = 1:spu.nsh(jel) % il faut trouver l'element
                                   correspondant dans le patch voisin, donc aussi
                                   trouver les qn correspondants
179                                 jshp = reshape (shpu(:,:,jdof,jel), spu.ncomp, []);
180                                 for inqn =1:bnd_mshed.nqn
181
182                                     jnqn = bnd_mshed.nqn+1 - inqn;
183
184                                     if sum(omg_n_loc(:,inqn))<0
185                                         mat_loc_mass(idof, jdof) = mat_loc_mass(idof
                                               , jdof)+...
186                                             sum(omg_n_loc(:,inqn))*jacdet_weights(
                                                   inqn)*ishp(inqn)*jshp(jnqn);
187                                     end
```

```
                                    end
                                end
                            end
                            Phi_in_loc(spv.connectivity(:, iel), spu.connectivity(:, jel
                               )) = ...
                                Phi_in_loc(spv.connectivity(:, iel), spu.connectivity(:,
                                    jel)) +  mat_loc_mass;

                        end
                    end
                    Phi_in{i_patch_cal,i_angle_cal,i_side_cal}(bnd_spaced.dofs,
                       adj_bnd_spaced.dofs) = Phi_in{i_patch_cal,i_angle_cal,i_side_cal
                       }(bnd_spaced.dofs,adj_bnd_spaced.dofs) + Phi_in_loc;
                end
            end

        end
    end
end

end
```

E.5 PF_ComputeSCAT_up

```
function SCAT = PF_ComputeSCAT_up(mp_sp,MIX,region_mix,OMEGA,ANISO,M,Phi)
n_patch = mp_sp.npatch;
```

```
n_group = size(Phi,1);
n_angle = size(OMEGA,2);

        SCAT = cell(n_group,n_patch,n_angle);
        for i_patch = 1:n_patch
            ndof = mp_sp.sp_patch{i_patch}.ndof;
            mix = MIX(region_mix(i_patch));
            for i_group = 1:n_group
                for i_angle = 1:n_angle
                    SCAT{i_group, i_patch,i_angle} = zeros(ndof,1);
                    for j_group = i_group+1 : n_group
                        for j_angle = 1: n_angle
                            SCAT{i_group,i_patch,i_angle} = SCAT{i_group,i_patch,i_angle
                               } + 1/(4*pi)*(  mix.SCAT0(i_group,j_group)+3*(ANISO-1)*
                               mix.SCAT1(i_group,j_group)*sum(OMEGA{i_angle}(1:2).*OMEGA
                               {j_angle}(1:2),'all')  )*OMEGA{j_angle}(4)*M{i_patch}*Phi
                               {j_group,i_patch,j_angle};
                        end
                    end
                end
            end
        end
end
```

E.6 PF_ComputeFIS

```
function FIS = PF_ComputeFIS (mp_sp,MIX,region_mix,Keff,M,PHI_ext)
n_group = size(PHI_ext,1);
FIS = cell(n_group, mp_sp.npatch);
for i_patch = 1:mp_sp.npatch
    ndof = mp_sp.sp_patch{i_patch}.ndof;
    mix = MIX(region_mix(i_patch));
    for i_group = 1:n_group
        FIS{i_group, i_patch} = zeros(ndof,1);
        for j_group = 1:n_group
            FIS{i_group, i_patch} = FIS{i_group,i_patch} + 1/(4*pi*Keff)*mix.CHI(i_group
               )*mix.NUSIGF(j_group)*M{i_patch}*PHI_ext{j_group,i_patch};
        end; clear j_group
    end; clear i_group
end; clear i_patch
end
```

E.7 Boucle d'itérations : Itération de puissance et source itérée

```
Keff=1;
K_err=1;
PHI_ext_err =1; % init

Phi_ext_temp = cell(n_group,n_patch,n_angle); % initialiser les flux angulaires
   intermediaires de la boucle externe
```

```
6  PHI_ext_temp = cell(n_group,n_patch); % initialiser les flux integres intermediaires de
       la boucle externe
7
8  % boucle externe
9  while K_err>1E-5 || PHI_ext_err > 1E-5
10
11     % calculer FIS a partir du flux integre PHI_ext (maj en fin du boucle) et SCAT_up a
           partir de du flux angulaire Phi_ext
12     FIS = PF_ComputeFIS (mp_sp,MIX,region_mix,Keff,M,PHI_ext); % (n_group,n_patch,
           n_angle)
13     SCAT_up = PF_ComputeSCAT_up(mp_sp,MIX,region_mix,OMEGA,ANISO,M,Phi_ext); % (n_group,
           n_patch,n_angle)
14
15     Phi_int_temp = Phi_int; % initialiser le flux angulaire intermediaire
16     PHI_int_temp = PHI_int; % initialiser le flux integre intermediaire
17     % boucle interne
18
19     for i_group =1:n_group
20         % calculer SCAT_down a partir de Phi_ext_temp (i-1 group ont ete maj dans
               Phi_ext_temp)
21         SCAT_down = PF_ComputeSCAT_down(mp_sp,MIX,region_mix,OMEGA,ANISO,i_group,M,
               Phi_ext_temp); %(n_patch,n_angle)
22
23         PHI_err_int=1;
24         % calcul de Phi_int
25         while PHI_err_int > 1E-5
```

```
26          % computing self scattering a partir de Phi_i
27          SCAT_self = PF_ComputeSCAT_self(mp_sp,MIX,region_mix,OMEGA,ANISO,i_group,M,
               Phi_int);%(n_patch,n_angle)
28          for i_angle = 1:n_angle
29              if OMEGA{i_angle}(1)>0 % ordering
30                  if OMEGA{i_angle}(2)>0
31                      patch_order = [10 12 6 8 2 4 1 3 5 7 9 11 13];
32                  else
33                      patch_order = [13 10 9 6 5 2 1 3 4 7 8 11 12];
34                  end
35              else
36                  if OMEGA{i_angle}(2)>0
37                      patch_order = [12 11 8 7 4 3 1 2 5 6 9 10 13];
38                  else
39                      patch_order = [11 13 7 9 3 5 1 4 2 8 6 12 10];
40                  end
41              end
42              for i_patch = patch_order
43                  mat = - OMEGA{i_angle}(1)*S_x{i_patch} - OMEGA{i_angle}(2)*S_y{
                       i_patch} + MIX(region_mix(i_patch)).TOTAL(i_group)*M{i_patch} +
                       P_k{i_patch,i_angle};
44
45                  Phi_in_eval = zeros(ndof,1);
46                  Phi_bc_eval = zeros(ndof,1);
47
48                  for i_side = 1:4
```

```
                    if not(isempty(in_voisin{i_patch,i_angle,i_side}))
                            Phi_in_eval = Phi_in_eval + Phi_in{i_patch, i_angle,
                               i_side}*Phi_int_temp{i_group,in_voisin{i_patch,
                               i_angle,i_side}(1),i_angle};
                    end

                    if not(isempty(bnd_angle_sortant{i_patch,i_angle,i_side}))
                            Phi_bc_eval = Phi_bc_eval + Phi_bc{i_patch, i_angle,
                               i_side}*Phi_int{i_group,i_patch, bnd_angle_sortant{
                               i_patch,i_angle,i_side}};
                    end
                end; clear i_side

                Phi_adj = Phi_in_eval + Phi_bc_eval;

                RHS = SCAT_up{i_group,i_patch,i_angle} + SCAT_down{i_patch,i_angle}
                   + SCAT_self{i_patch,i_angle} + FIS{i_group,i_patch} - Phi_adj;
                Phi_int_temp{i_group,i_patch,i_angle}= mat\RHS;
            end; clear i_patch
        end; clear i_angle


        for i_patch = 1:n_patch
            PHI_int_temp{i_group,i_patch} = zeros(mp_sp.sp_patch{i_patch}.ndof,1);
            for i_angle = 1:n_angle
```

```
                PHI_int_temp{i_group,i_patch} = PHI_int_temp{i_group,i_patch} +
                    OMEGA{i_angle}(4)*Phi_int_temp{i_group,i_patch,i_angle};
            end; clear i_angle
        end; clear i_patch

        S = 0;
        for i_patch = 1:n_patch
            S = S + sqrt(  sum( (PHI_int_temp{i_group,i_patch}-PHI_int{i_group,
                i_patch}).^2,'all' ) / sum( PHI_int{i_group,i_patch}.^2,'all' ) );
        end; clear i_patch
        PHI_err_int = S; clear S

        % heredite
        Phi_int = Phi_int_temp;
        PHI_int = PHI_int_temp;

    end % sortie boucle interne pour i_group

    for i_patch = 1:n_patch
        PHI_ext_temp{i_group,i_patch} = PHI_int_temp{i_group,i_patch};
        for i_angle= 1:n_angle
            Phi_ext_temp{i_group,i_patch,i_angle} = Phi_int{i_group,i_patch,i_angle
                };
        end; clear i_angle
    end; clear i_patch

```

```
92      end; clear i_group % sortie du boucle interne pour tous les groupes d'energie
93
94      S1=0;
95      S2=0;
96      for i_patch = 1:n_patch
97          for j_group = 1:n_group
98              mix = MIX(region_mix(i_patch));
99              CHI = mix.CHI;
100             if CHI(j_group)>0
101                 NUSIGF = mix.NUSIGF;
102                 if all(not(NUSIGF<=0))
103                     for i_group = 1:n_group
104                         S1 = S1 + CHI(j_group)*NUSIGF(i_group)*vec{i_patch}*PHI_ext{
                               i_group,i_patch};
105                         S2 = S2 + CHI(j_group)*NUSIGF(i_group)*vec{i_patch}*PHI_ext_temp
                               {i_group,i_patch};
106                     end; clear i_group
107                 end
108             end
109         end; clear j_group
110     end; clear i_patch
111
112     Keff_temp = Keff*S2/S1;
113     K_err = sqrt( (Keff_temp - Keff)^2 )/abs(Keff);
114     Keff=Keff_temp;
115     if K_err < 1E-5
```

```
116         PHI_ext_err = 0;
117         for i_group = 1:n_group
118             for i_patch = 1:n_patch
119                 PHI_ext_err = PHI_ext_err + sqrt(  sum( (PHI_ext_temp{i_group,i_patch}-
                        PHI_ext{i_group,i_patch}).^2,'all' )/ sum( PHI_ext{i_group,i_patch
                        }.^2,'all' ) );
120             end
121         end
122     end
123     PHI_ext = PHI_ext_temp;
124     Phi_ext = Phi_ext_temp;
125 end
```