Grenoble INP - Ensimag
École nationale supérieure d'informatique et de mathématiques appliquées

Pré Rapport

Laboratoire Jacques-Louis Lions

Interpolation polynomiale en grande dimension pour un problème de construction de bibliothèque neutronique

SIALA Rafik

3A - Specialization MMIS /M2 MSIAM

du 03/04/2017 au 01/09/2017 (22 semaines)

Laboratoire Jacques-Louis Lions
UPMC, 4 place Jussieu
75005 Paris, France

Tuteur entreprise
COHEN Albert, cohen@ann.jussieu.fr
MADAY Yvon, maday@ann.jussieu.fr
Tuteur Ensimag:
MAITRE Emmanuel,
emmanuel.maitre@imag.fr

Contents

1 Introduction

Un nouveau cadre pour les calculs neutroniques de base développé à EDF R&D, dans le département de SINETICS (SImulation NEutronique, Technologie de l'Information, Calcul Scientifique) est basé sur la résolution de l'équation de Boltzmann (ou une approximation) pour les neutrons.
Cette équation nécessite en entrée des sections efficaces qui modélisent les interactions entre les neutrons induits par la fission et les noyaux provenant soit du combustible soit du modérateur.

Ces sections efficaces (au nombre de plusieurs milliers) dépendent de d paramètres locaux (dits paramètres de rétroaction), tels que la densité de l'eau, la concentration en bore, la température du carburant, le brûlage du combustible, etc. Elles sont stockées dans des «bibliothèque nucléaire» et doivent être consultées régulièrement tout au long de la simulation quand les paramètres de rétroaction évoluent.

Dans le cadre d'une thèse CIFRE [3] qui vient d'être soutenue au Laboratoire J.-L. Lions, on a développé une nouvelle méthode [4] basée sur l'utilisation de bases tensorielles utilisant des fonctions directionnelles adaptées aux fonctions qu'on veut représenter et qui permet de diminuer le stockage, le coût des calculs et la rapidité pour reconstruire les sections efficaces avec une très grande précision.

L'objectif du stage est de compléter cette étude et de la comparer avec une approche alternative (méthodes parcimonieuses) proposée récemment au Laboratoire J.-L. Lions par Albert Cohen [2]. Le stage porte sur la mise en œuvre de ces méthodes parcimonieuses pour le cas particulier de l'approximation des sections efficaces. Il s'agit pour certaines d'entre elles de méthodes non-intrusives (donc basées sur des évaluations obtenues par un code qui peut être vu comme une boite noire, ce qui est bien adapté à la situation). Elles font appel à des techniques de représentations parcimonieuses dans le but de tirer parti des anisotropies potentielles dans la fonction qu'on veut capturer (certaines variables pouvant en particulier être plus sensibles que d'autres) et des propriétés de régularité en fonction des différentes variables.

Mots clés: Sections efficaces, méthodes parcimonieuses, ...

2 Présentation de la structure d'accueil

Le laboratoire, créé en 1969, porte le nom de son fondateur Jacques-Louis Lions. Il s'agit maintenant d'une unité de recherche conjointe à l'Université Pierre et Marie Curie, à l'université Paris Diderot et au Centre National de la Recherche Scientifique.

Le Laboratoire Jacques-Louis Lions constitue le plus grand laboratoire de France et l'un des principaux au monde pour la formation et la recherche en mathématiques appliquées.

Il accueille l'activité de deux masters deuxième année ce qui représente une centaine d'étudiants. Ses axes de recherche recouvrent l'analyse, la modélisation et le calcul scientifique haute performance de phénomènes représentés par des équations aux dérivées partielles. Fort d'environ 100 enseignants-chercheurs, chercheurs, ingénieurs, personnels administratifs permanents ou émérites, et d'autant de doctorants ou post-doctorants, le LJLL collabore avec le monde économique et avec d'autres domaines scientifiques à travers un large spectre d'applications: dynamique des fluides; physique, mécanique et chimie théoriques; contrôle, optimisation et finance; médecine et biologie; traitement du signal et des données.

Figure 1: Localisation de la structure d'accueil

3 Présentation de la problématique du projet

3.1 Présentation du contexte

Afin d'exploiter au mieux son parc nucléaire, la R&D d'EFD est en train de développer une nouvelle chaîne de calcul appellée ANDROMEDE, pour simuler le cœur des réacteurs nucléaires avec des outils à l'état de l'art. L'un des éléments de cette chaine est le code neutronique COCAGNE, dont l'un des objectifs est de résoudre numériquement l'équation du **transport neutronique** (ou l'une de ses approximations), qui nous permet d'obtenir des grandeurs physiques d'intérêt pour décrire le comportement du réacteur, telle que: le flux neutronique, le facteur de multiplication effectif, la réactivité ...
La résolution de cette équation nécessite une grande quantité de données physiques, en particulier les **sections efficaces**.

En neutronique, les sections efficaces représentent la probabilité d'interaction d'un neutron incident avec les noyaux cibles, pour différents types d'interaction. Dans une simulation neutronique, les sections efficaces peuvent être représentées comme des fonctions dépendant de plusieurs paramètres physiques. Ces paramètres sont utilisés pour décrire les conditions thermo-hydroliques et la configuration du cœur du réacteur, tel que: la température du combustible, concentration en bore, niveau de xénon, burnup, ...
Ainsi les sections efficaces sont des fonctions multivariées définies sur un espace appelé l'**espace de phase des paramètres**.

Dans la simulation d'un cœur complet, le nombre de valeurs des sections efficaces est de l'ordre de plusieurs milliards. Leur détermination demande un calcul complexe et lent. Pour résoudre ce problème, un schéma en deux étapes est proposé afin de réduire la complexité des calculs et d'accomplir la simulation du cœur. Ce schéma est basé sur la structure multi-échelle du noyau du réacteur: noyau contenant des assemblages, assemblages contenant des tiges/cellules (voir figure 2).

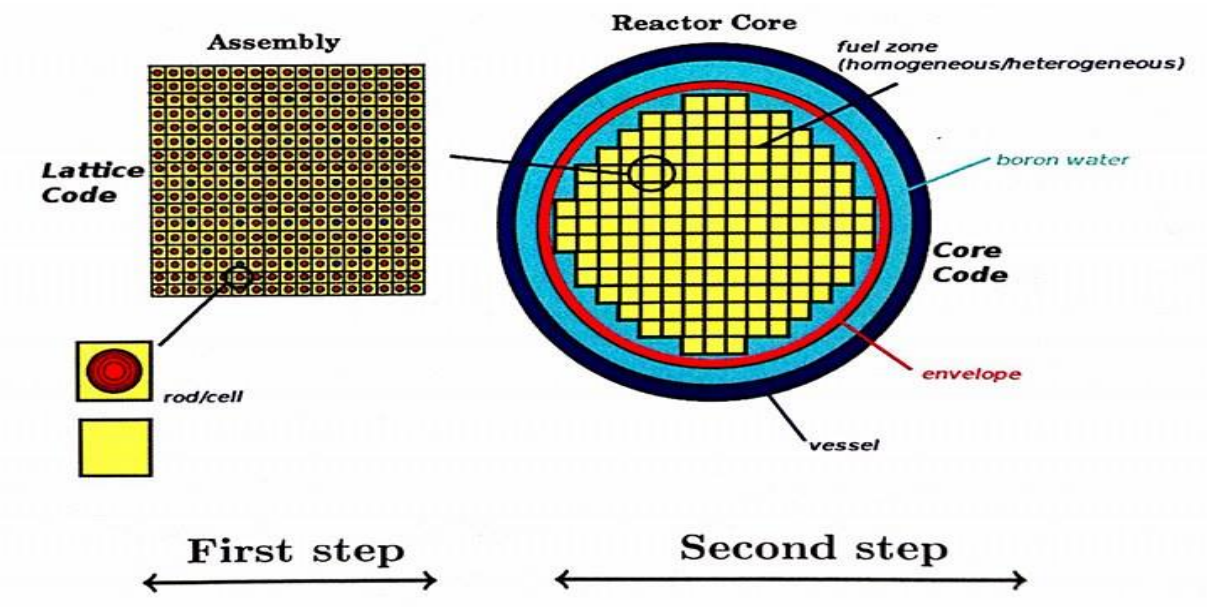

Figure 2: Schéma à 2 étapes basé sur la structure multi-échelle du noyau du réacteur

Par conséquent, 2 étapes sont effectuées séparément par deux codes:

- **Code réseau:** permet de précalculer les sections efficaces sur des points présélectionnés dans l'espace de phase des paramètres (sur chaque assemblage). L'équation du transport neutronique est résolue de manière précise grâce à des discrétisations spatiales et énergétique (calcul hors ligne). L'information obtenue sur les sections efficaces est stockée dans des fichiers (appelés les **bibliothèques neutroniques**).
- **Code de cœur:** permet d'évaluer les sections efficaces en n'importe quel point du cœur en utilisant les informations stockées dans les biblothèques neutroniques. Ces valeurs sont ensuite utilisées comme données (points d'interpolations) pour résoudre l'équation du transport neutronique au niveau du cœur du réacteur.

Le schéma ci-dessous (figure 3) illustre bien le modèle de reconstruction des sections efficaces.

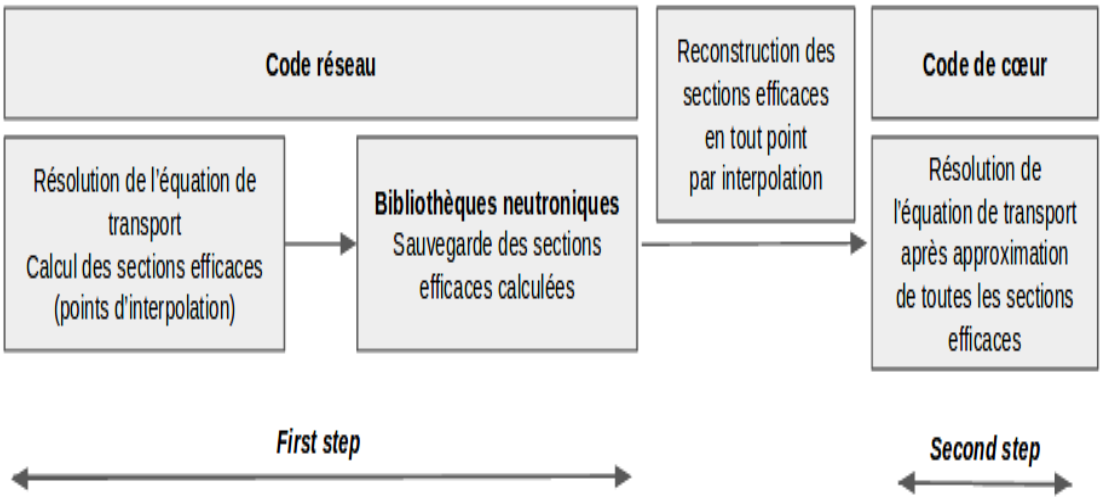

Figure 3: Schéma à 2 étapes pour la simulation du cœur du réacteur

Avec ce modèle, le nombre de points de discrétisation augmente exponentiellement en fonction du nombre de paramètres ou de manière considérable quand on ajoute des points sur un des axes. En effet, si on suppose que les sections efficaces dépendent de d paramètres et que chaque paramètre est discrétisé par n points sur l'axe correspondant, dans ce cas le nombre de calculs hors ligne et la taille des bibliothèques neutroniques sont de l'ordre de $O(n^d)$.

3.2 Objectifs attendus, problème à résoudre

Avec les contraintes industrielles imposées à EDF, le modèle de l'interpolation multilinèaire devient très coûteux en mémoire et en temps de calcul pour des situations complexes à cause de l'augmentation rapide et exponentielle du nombre de points de calcul. Ces situations peuvent arriver, par exemple, dans le cas accidentel où de nouveaux paramètres s'ajoutent (température de l'eau, taille des lames d'eau) et/ou quand le domaine de calcul s'étend (avec des domaines de définition plus larges).

Afin de dépasser les limitations de ce modèle, l'objectif de ce stage est de développer un modèle de reconstitution des sections efficaces tout en répondant aux exigences suivantes:

- **Calculs hors ligne:**
  - utiliser moins de précalculs et ceci en choisissant les points de discrétisation de manière astucieuse.
  - stocker moins de données pour la reconstruction des sections efficaces.
- **Calculs en ligne:**
  - avoir une bonne précision pour l'évaluation des sections efficaces.

4 Solution technique mise en œuvre

Le but de cette partie est de présenter une méthode d'interpolation polynomiale adaptative à haute dimension. Cette méthode permet d'approximer une fonction multivariée avec une forte précision tout en utilisant moins de données pour la reconstruction des sections efficaces.

Dans la section précédente, on a vu que les sections efficaces sont des fonctions multivariées définies sur un espace appelé l'espace de phase des paramètres qu'on note $\mathcal{P} = P^d$. On peut donc modéliser une section efficace par une fonction f tel que:
$f : \mathcal{P} \to V_\Lambda$ avec P un compact de $\mathbb{R}$ (ou $\mathbb{C}$) et V l'espace des valeurs prises par les sections efficaces.

En pratique, pour $y \in \mathcal{P}$, on peut approcher $f(y)$ avec un code très couteux. Le but est d'éviter ce lent calcul, tout simplement, en considérant une approximation de f en y, qu'on note $\tilde{f}(y)$. Pour cela, il faut commencer par choisir un certain nombre de point $y^i \in \mathcal{P}, i \in 1..m$. Ensuite on évalue $f_i = f(y^i) = f(y^i_1, .., y^i_d)$ en faisant m appel au code couteux. Ensuite, on utilise $y^1, .., y^m, f_1, .., f_m$ pour fabriquer $\tilde{f}$ par interpolation.

L'objectif est donc de construire un opérateur d'interpolation I_Λ qui permet de reconstruire une fonction f définie sur $\mathcal{P}$ à valeurs dans V_Λ.
Les méthodes d'interpolation polynomiale d'ordre supérieur construisent des approximations de la forme $u_\Lambda(y) = \sum_{\nu \in \Lambda} u_\nu y^\nu$ avec $\Lambda \in \mathcal{F}$ un ensemble fini de multi-indices $\nu = (\nu_j)_{j \geq 1} \in \mathcal{F}$ et $y^\nu = \prod_{j \geq 1} y_j^{\nu_j}$.

Remarque: En dimension finie ($d < \infty$), l'ensemble d'indices $\mathcal{F}$ coincide avec $\mathbb{N}_0^d$.
Les u_Λ sont choisis dans l'espace V_Λ, défini par $V_\Lambda = vect\left\{\sum_{\nu \in \Lambda} v_\nu y^\nu : v_\nu \in V\right\}$

4.1 Description de la solution envisagée

Afin d'approximer f en tout point, on va procéder par interpolation multivariée. Dans cette partie, on va revoir le principe de l'interpolation de base en $1D$, puis en dimension quelconque. Par la suite, on évaluera le coût d'une telle opération et son évolution en fonction du nombre de points d'interpolation et de la dimension de l'espace de phase des paramètres. Ensuite, on présentera une meilleure méthode d'interpolation adaptative en grande dimension (moins couteuse) basée sur une construction hiérarchique de l'opérateur d'interpolation.

4.1.1 Interpolation monovariée et tensorisation:

Cas d=1:
Soit $(y_k)_{k \geq 0}$ une séquence de points deux à deux distincts. On note I_k l'opérateur d'interpolation polynomiale associé à la séquence $\{y_0, .., y_k\}$.
Supposons que la fonction f qu'on veut interpoler est à valeurs dans $\mathbb{R}$ (ou $\mathbb{C}$).
On a alors,

$$I_k(f) = \sum_{i=0}^{k} f(y_i) l_i^k \tag{1}$$

avec $l_i^k(y) = \prod_{j=0 i\neq j}^{k} \frac{(y-y_j)}{(y_i-y_j)}$, polynômes d'interpolation de Lagrange de degrès $(k-1)$ associé à la séquence $\{y_0, .., y_k\}$ (voir figure 4).

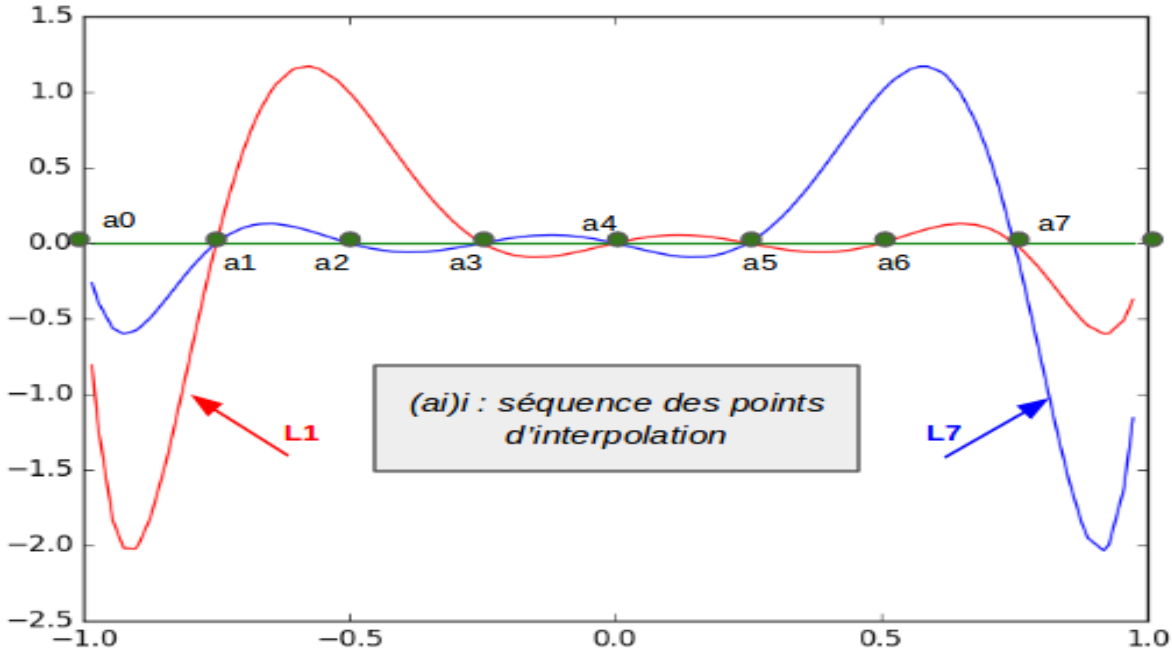

Figure 4: Exemples de polynômes d'interpolation de Lagrange

Cas d>1:
On procède par tensorisation. C'est à dire, pour $\alpha_1, .., \alpha_d \in \{0, .., k-1\}$

$$\tilde{f}(y_1, .., y_d) = \bigotimes_{i=1}^{d} I_k(f)(y_1, .., y_d) \in \mathbb{Q}_{k-1} = vect\{y \to y_1^{\alpha_1}..y_d^{\alpha_d}\} \quad (2)$$

$$= \sum_{k1=1}^{k} .. \sum_{kd=1}^{k} (f(y_{k_1}, .., y_{k_d}) l_{k_1}(y_1)..l_{k_d}(y_d)) \quad (3)$$

Pour avoir une bonne précision, il est préférable de choisir un nombre relativement grand de points d'interpolation. Si par exemple, on prend k_j points suivant la direction j, on obtient un nombre total de points égal à $\prod_{i=1}^{d} k_j$. Ceci pose problème, bien évidemment, lorsque d est grand. En effet, non seulement le calcul de $\tilde{f}$ sera couteux, mais aussi, si on souhaite ajouter n points suivant la variable i ça revient à ajouter $n \prod_{j=1, j\neq i}^{d} k_j$ noeuds dans la grille (voir figure 5).

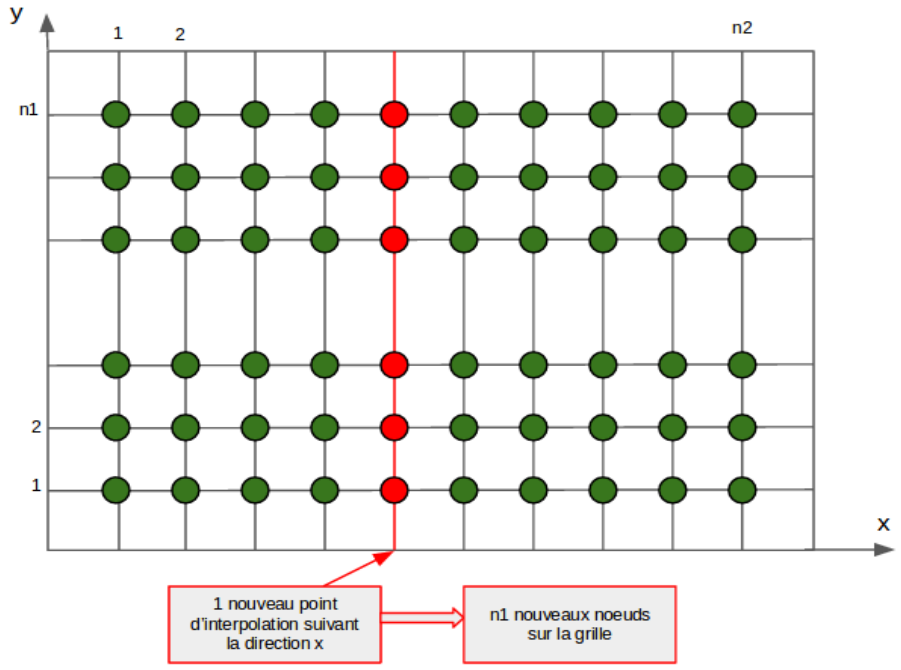

Figure 5: Influence de l'ajout d'un point d'interpolation sur une seule direction

Cette méthode directe d'interpolation devient très couteuse lorsqu'il s'agit de fonctions multivariées en grande dimension.
Une solution serait de calculer l'opérateur d'interpolation par une méthode hiérarchique. C'est à dire qu'on va utiliser les approximations $I_j(f),\ j < k$ de f (en utilisant j points d'interpolation) pour approcher $I_k(f)$.

4.1.2 Construction hiérarchique de l'opérateur d'interpolation:

Cas d=1:
Soit $k \geq 0$, on définit l'opérateur différence Δ_k par:

$$\Delta_k = I_k - I_{k-1} \tag{4}$$

On suppose, par convention, que Δ_{-1} est l'opérateur nul. De cette façon, Δ_0 correspond à l'opérateur qui à f associe le polynôme constant $f(y_0)$.
On a alors,

$$I_n = \sum_{k=0}^{n} \Delta_k \tag{5}$$

On définit ensuite, les polynômes hiérarchiques de degré k associés à la séquence $\{y_0, .., y_k\}$ par

$$h_k(y) = \prod_{j=0}^{k-1} \frac{(y - y_j)}{(y_k - y_j)}, k > 0,\ et\ h_0(y) = 1, \tag{6}$$

On a alors,

$$\Delta_k(f) = \alpha_k(f) h_k, \quad \alpha_k(g) = f(y_k) - I_{k-1} f(y_k) \tag{7}$$

On obtient alors, par un processus itératif, la représentation suivante:

$$I_n(f) = \sum_{k=0}^{n} \alpha_k(f) h_k \tag{8}$$

Ainsi pour calculer l'interpolant de f avec n points, il suffit de connaitre les interpolants de f d'ordre i, $i \in \{1, .., n-1\}$, et ceci d'une manière relativement rapide.

Cas d>1:
On veut effectuer une interpolation polynomiale pour une séquence imbriquée d'ensemble $(\Lambda_n)_{n\geq 1}$ avec $n = \sharp(\Lambda_n)$.
Pour la suite, on définit :

- Le point multivarié $\mathbf{y}_\nu$ par $\mathbf{y}_\nu = (y_{\nu_j})_{j\geq 1} \in \mathcal{P}$
- La fonction polynomiale hiérarchique tensorisée $\mathbf{H}_\nu$ par $\mathbf{H}_\nu(y) = \prod_{j\geq 1} h_{\nu_j}(y_j)$

Pour que la méthode décrite précedament en 1D fonctionne en dimension d quelconque, il faut s'assurer que pour tout $n \geq 1$, Λ_n est monotone.

Définition: Un ensemble $\Lambda \subset \mathcal{F}$, non vide, est dit monotone, si

$$\nu \in \Lambda \; et \; \mu < \nu \Rightarrow \mu \in \Lambda \tag{9}$$

avec $\mu \leq \nu$ signifie que $\mu_j \leq \nu_j$ pour tout j.

Dans cette configuration, Λ_n peut être vu comme une section $\{\nu^1, .., \nu^n\}$ d'une séquence $(\nu^k)_{k\geq 1} \in \Lambda^{\mathbb{N}}$. Cette observation mène à un algorithme efficace pour le calcul de $I_{\Lambda_n} g$ à partir de $I_{\Lambda_{n-1}} g$ et de la valeur de f en le nouveau point $\mathbf{y}_{\nu^n}$. En effet, par temsorisation, on observe que Δ_{ν^n} est multiple de la fonction hiérarchique tensorisée $\mathbf{H}_{\nu^n}$ défini précedement.
Puisque $\mathbf{H}_{\nu^n}(\mathbf{y}_{\nu^n}) = 1$, alors

$$\begin{aligned} \Delta_{\nu^n} g &= \Delta_{\nu^n} f(\mathbf{y}_{\nu^n}) \mathbf{H}_{\nu^n} && (10)\\ &= (I_{\Lambda_n} f(\mathbf{y}_{\nu^n}) - I_{\Lambda_{n-1}} f(\mathbf{y}_{\nu^n})) \mathbf{H}_{\nu^n} && (11)\\ &= (f(\mathbf{y}_{\nu^n}) - I_{\Lambda_{n-1}} f(\mathbf{y}_{\nu^n})) \mathbf{H}_{\nu^n} && (12) \end{aligned}$$

Donc

$$I_{\Lambda_n} g = I_{\Lambda_{n-1}} g + (f(\mathbf{y}_{\nu^n}) - I_{\Lambda_{n-1}} f(\mathbf{y}_{\nu^n})) \mathbf{H}_{\nu^n} \tag{13}$$

Par conséquent, les polynômes $I_{\Lambda_n} g$ sont donnés par:

$$I_{\Lambda_n} g = \sum_{k=0}^{n} g_{\nu^k} \mathbf{H}_{\nu^k} \tag{14}$$

avec g_{ν^k} définis récursivement par:

$$g_{\nu^1} = f(y_0), \quad g_{\nu^{k+1}} = f(\mathbf{y}_{\nu^{k+1}}) - I_{\Lambda_k} f(\mathbf{y}_{\nu^{k+1}}) = f(\mathbf{y}_{\nu^{k+1}}) - \sum_{i=1}^{k} g_{\nu^i} \mathbf{H}_{\nu^i}(\mathbf{y}_{\nu^{k+1}}) \tag{15}$$

Interpolation adaptative et séquence de Leja: Les ensembles Λ_n peuvent être choisis préalablement de sorte qu'ils forment une séquence imbriquée d'ensembles monotones. Ils peuvent aussi être construits d'une manière astucieuse lors du calcul de l'interpolant en construisant un chemin d'indices qui correspond à un ordre entre les points d'interpolations.

Soit l'analogie suivante: si $(\mathbf{H}_\nu)_{\nu\in\mathcal{F}}$ est une base orthoromée de $L^2(\mathcal{P})$ alors le choix d'un ensemble d'indices Λ_n qui minimise l'erreur serait de prendre les n plus grands $a_\nu \left|g_\nu\right|$ tel que $a_\nu = \left\|\mathbf{H}_\nu\right\|_{L^\infty(\mathcal{P})} = \prod_{j\geq 1} \left\|h_{\nu_j}\right\|_{L^\infty(\mathcal{P})}$.
Cette stratégie permet d'avoir une séquence imbriquée $(\Lambda_n)_{n\geq 1}$, cependant il n'est pas certain que les ensembles Λ_n soient monotones.
Afin de résoudre ce problème, on définit la notion de voisins pour n'importe quel ensemble monotone Λ par,

$$\mathcal{N}(\Lambda) = \{\nu \notin \Lambda : R_\nu \in \Lambda \cup \{\nu\}\}, R_\nu = \{\mu \in \Lambda : \mu < \nu\} \tag{16}$$

Ainsi, avec cette définition, l'algorithme ci-dessous mène à une séquence imbriquée d'ensembles monotones.
Algorithme d'Interpolation Adaptative:

- Commencer par $\Lambda_1 = \{0_\Lambda\}$
- En supposant Λ_{n-1} calculé, trouver $\nu^n = argmax\ \{a_\nu \left|g_\nu\right| : \nu \in \mathcal{N}(\Lambda_{n-1})\}$, et définir $\Lambda_n = \Lambda_{n-1} \cup \{\nu\}$

Il est important de noter que le choix de la séquence de points d'interpolation joue un rôle critique pour la stabilité de l'opérateur d'interpolation multivariée définie par la constante de Lebesgue:

$$\mathbb{L}_\Lambda = sup_{f\in B(\mathcal{P})-0} \frac{\left\|I_\Lambda f\right\|_{L^\infty(\mathcal{P})}}{\left\|f\right\|_{L^\infty(\mathcal{P})}} \tag{17}$$

avec $B(\mathcal{P})$ est l'espace des fonction bornées définis sur $\mathcal{P}$
L'objectif est de choisir la séquence $(y_k)_{k\geq 0}$ tel que les constantes univariées de Lebesgue

$$\lambda_k = max_{f\in B(\mathcal{P})-0} \frac{\left\|I_k f\right\|_{L^\infty(\mathcal{P})}}{\left\|f\right\|_{L^\infty(\mathcal{P})}} \tag{18}$$

associées à l'opérateur d'interpolation univariée I_k croient modérément par rapport à k.
Une telle séquence univariée peut être construite en fixant $y_0 \in P$ et en définissant inductivement

$$y_k = argmax_{y\in P} \prod_{j=0}^{k-1} |y - y_j| \tag{19}$$

Cette séquence $(y_k)_{k\geq 0}$ est appellée séquence de Leja [1] sur P. Elle modère la croissance des constantes de Lebesgue et a une implication intéressante sur le choix adaptatif des ensembles Λ_n.

4.2 Implèmentation de la solution

Dans cette partie, on va mettre en pratique cette méthode et on va présenter certains détails de l'implémentation de cette solution (notamment la construction de la séquence de Leja).

Tout d'abord cette solution a été implémenté en C++. Elle a été, par la suite, téstée sur des fonctions définies à l'avance en des points choisis d'une manière aléatoire. Le code est constitué principalement de trois classes:

- **Classe Interpolation:** elle contient les structures de données contenants les différentes quantités qui entrent en jeu dans la construction de l'opérateur d'interpolation, notamment les points d'interpolation, les points de test, les g_ν et le chemin d'indices vus dans la section précédente. De plus, l'algorithme AI ainsi que certaines fonctions intermédiaires, (notamment celle responsable de la recherche des voisins dans une grille) sont implémentés dans la classe *Interpolation*

- **Classe MultivaraintPoint:** il s'agit d'une classe générique qui modélise à la fois les multi-indices ainsi que les points réels multivariés. C'est à dire qu'une instance de cette classe peut renfermer soit des valeurs réelles ou alors entières (multi-indice). Les opérateurs d'affectation, d'addition, d'affichage ainsi que de comparaison de points multivariés sont implémentés dans cette classe.

- **Classe Utils:** il s'agit d'une classe statique renfermant certaines fonctions utiles notamment les fonctions d'affichages, les fonctions qui gèrent la création des séquences d'interpolations (notamment la séquence uniforme, les zéros de Tchebychev ainsi que la séquence de Leja).

Le choix de la séquence d'interpolation joue un rôle important dans la stabilité de l'opérateur d'interpolation. Dans la suite, on verra quelques détails sur la construction de la séquence des points de Leja.

Séquence de Leja: L'idée est de discrétiser l'intervalle de définition U de la fonction f en un grand nombre de points $(a_i)_{i=1..n(=100000)}$. Supposons qu'on a construit à l'étape i la séquence $\{y_0, .., y_i\}$, alors pour avoir le point y_{i+1}, on compare le produit des distances de chaque point de discrétisation aux points de Leja déja calculés, puis on choisit le max. La façon la plus naturelle de choisir les points a_i est de les choisir uniformément répartis dans l'intervalle de définition de f.
On obtient alors une suite de polynômes qui interpole f en de plus en plus de points. On pourrait s'attendre à ce que la suite converge uniformément vers f lorsque le nombre de points d'interpolation augmente.
Malheureusement, ce n'est pas le cas, ce phénomène est connu sous le nom de phénomène de Runge.
Une solution est, étonnamment, de ne pas choisir les points uniformément répartis. Une raison pour cela est l'inégalité suivante : si f est de classe C^N sur l'intervalle U et si L est le polynôme d'interpolation de Lagrange en les points $a_1, .., a_N$, alors on a

$$\forall x \in U, |f(x) - L(x)| \leq sup_{x \in U}(\prod_{i=1}^{n} |x - a_i|) \frac{\left\| f^{(n)} \right\|_\infty}{n!} \tag{20}$$

L'idée est donc de choisir les a_i de sorte que $sup_{x\in U}\prod_{i=1}^{n}|x-a_i|$ soit le plus petit possible. On démontre que ceci est réalisé lorsque les a_i sont les zéros du polynôme de Tchebychev de degré n, à savoir $a_i = cos(\frac{(2i-1)\pi}{n}), i = 1, .., n$.
La figure 6 montre une grille 2D formés par des points de Leja construits dans le segment $[-1, 1]$.

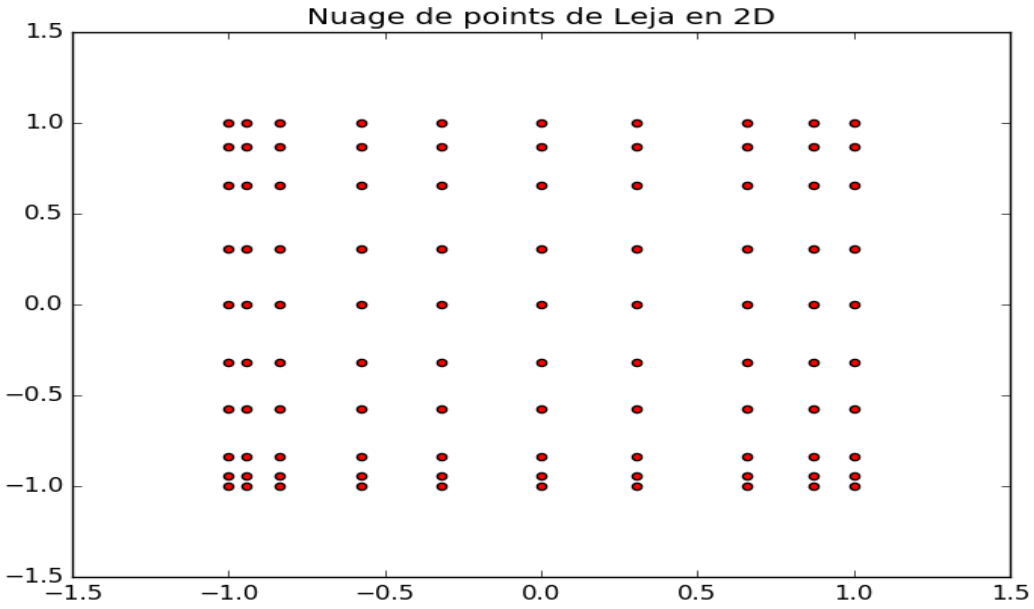

Figure 6: Grille 2D des points de Leja

5 Évaluation de l'efficacité de la méthode

5.1 Résultats expérimentaux:

Afin d'évaluer les performances de l'algorithme d'interpolation adaptative multivariée, des tests ont été faits sur différentes fonctions, séquences d'interpolation et en plusieurs dimensions. Les résultats en matière de temps de calcul et d'erreur d'interpolation sont résumés dans le tableau ci-dessous. La fonction interpolée dans ce test est la fonction f définie par:

$$f : [-1, 1]^d \to \mathbb{R} \tag{21}$$

$$x = (x_1, .., x_d) \to sin(\|x\|^2) = sin(\sum_{i=1}^{d} x_i^2) \tag{22}$$

Nombre de points d'interpolation (d=2)	Erreur d'interpolation M_0 | M_1	Temps de calcul (s) M_0 | M_1
100	$1.5\ 10^{-4}$ | $1.5\ 10^{-4}$	0.172 | 0.003
500	$8.0\ 10^{-4}$ | $8.7\ 10^{-5}$	21.24 | 0.036
1000	$1.2\ 10^{-6}$ | $2.4\ 10^{-5}$	173.2 | 0.157
5000	$3.2\ 10^{-7}$ | $2.4\ 10^{-7}$	$> 1h$ | 6.529
10000	——— | $5.3\ 10^{-7}$	$+\infty$| 32.21
50000	——— | $1.0\ 10^{-8}$	$+\infty$| 2270.

La méthode M_0 correspond à une première version de l'algorithme AI. Une fois téstée et validée, on est passé à une version M_1 apportant certaines améliorations. Les critères de validation sont les suivants:

- L'interpolation donne un résultat exact (l'erreur est nulle), si la fonction interpolée est de nature polynomiale
- Le résultat d'interpolation ne dépend pas du chemin suivi. C'est-à-dire que si on fixe au préalable un ordre différent pour les points d'interpolation que celui obtenu par l'algorithme AI (tout en gardant le caractère monotone de l'ensemble d'indices), on obtient le même résultat.
- Les coefficients α_ν ou g_ν ne dépendent pas du chemin suivi. Ils ne dépendent que de la fonction interpolée et de la séquence d'interpolation.

Le tableau ci-dessous montre les résultats de l'interpolation de f pour différentes valeurs de d. Les temps de calcul présentés dans le tableau correspondent au temps que met l'algorithme AI pour approximer f avec une erreur $e \leq seuil = 10^{-4}$.
On note t le temps de calcul, et n_iter le nombre d'itérations effectuées par l'algorithme avant de s'arrêter.

Remarque: Le nombre maximal d'itérations correspond au nombre total de points d'interpolation construit initialement. L'algorithme AI sélectionne des points à partir de cette séquence initiale. Au cours de son exécution, chaque fois que l'algorithme effectue 10% du nombre maximal d'itérations, il évalue l'erreur d'interpolation. Si elle dépasse un certain seuil, l'algorithme s'arrête.

Nombre de points d'interpolation	d=2 n_iter | t	d=3 n_iter | t	d=4 n_iter | t
1000	600 | 0.06	900 | 0.16	1000 | 0.24
5000	1000 | 0.22	1500 | 0.46	3000 | 2.24
10000	1000 | 0.24	4000 | 4.83	7000 | 13.04
20000	4000 | 5.47	8000 | 19.53	12000 | 40.92

5.2 Optimisations apportées:

- **Recherche du voisin courant:** Dans M_0, on recherchait le nouveau voisin parmi tous les points de la grille. Ce n'est pas évidemment la meilleure idée car la grille contient non seulement les voisins potentiels mais aussi les points déjà sélectionnés et traités. Donc, il est inutile de la parcourir en totalité à chaque étape car ça devient très couteux.
  Dans la version M_1, une meilleure stratégie a été adopté. En effet, on ne cherche plus le nouveau meilleur voisin parmi tous les points de la grille (100 si on est en dimension 2 et qu'on a choisi 10 points de discrétisation sur chaque direction), mais plutôt parmi un petit ensemble de points correspondant aux points voisins. (voir figure 7: à gauche la version M_0 et à droite, la version M_1. La recherche du nouveau voisin s'effectue dans l'ensemble délimité par un coutour bleu).

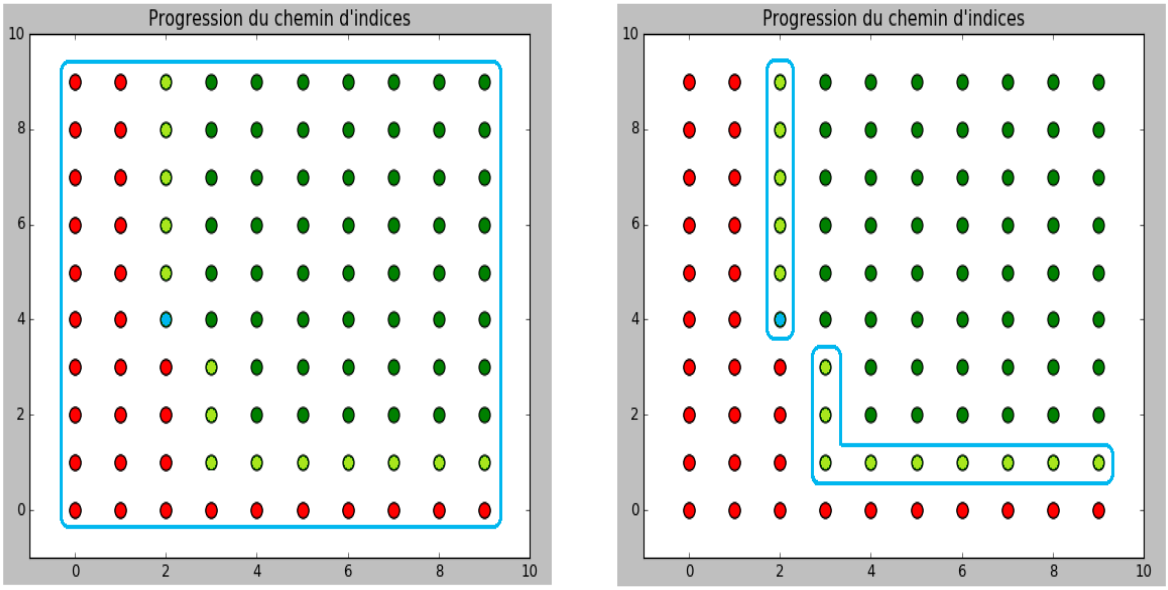

Figure 7: Recherche du voisin courant

- **Calcul des α_ν (ou g_ν):** Dans M_0, à chaque étape de l'algorithme AI, on construit un sous-ensemble de points qui correspondent à tous les voisins courants. Ensuite on calcule tous les α_ν (ν étant les multi-indices correspondants aux voisins courants) et on choisi le voisin ayant le plus grand α_ν. Il est possible qu'au cours d'une certaine itération (voir toutes les itérations) de recalculer un α_ν déja calculé à l'itération précédente. Ceci est, bien évidemment, couteux.
  Dans M_1 on procède d'une manière plus astucieuse. En effet, on peut éviter cette redondance, tout simplement, en utilisant une structure de données qui donne un coût de recherche en $O(log(n))$. Par exemple, on peut utiliser le conteneur map de la STL. Dans ce cas, dès qu'on calcule un α_ν pour un ν donné on ajoute à la map le couple (ν, α_ν). Ainsi, si on a besoin, dans une itération ultérieure, de calculer ce même α_ν, on irait le chercher directement dans cette map avec un coût en temps logarithmique.

5.3 Importance du choix de la séquence d'interpolation:

Le choix de la séquence des points d'interpolation joue un rôle critique dans la stabilité de l'opérateur d'interpolation. En effet, des tests ont été fait en utilisant une séquence uniforme puis les points de Leja, et les résultats le confirment. Avec une séquence uniforme, dès qu'on a un nombre de points assez grand, l'erreur d'interpolation devient très grande et l'algorithme diverge. Par ailleurs avec la séquence de Leja, le résultat de l'interpolation devient plus précis et l'erreur moins importante dès qu'on augmente la taille de la séquence. On obtient de ce fait un algorithme assez performant et stable.
Le tableau ci-dessous montre l'évolution de l'erreur d'interpolation en fonction de la nature de la séquence d'interpolation ainsi que du nombre de points. L'erreur d'interpolation est calculée en dix points de test et la fonction interpolée est la même utilisée précédemment.

Nombre de points d'interpolation	Séquence uniforme	Séquence de Leja
100	0.3	$1.5\ 10^{-4}$
500	$6.1\ 10^{4}$	$8.7\ 10^{-5}$
1000	$6.8\ 10^{5}$	$2.4\ 10^{-5}$

6 Etat d'avancement

L'algorithme de l'Interpolation Adaptative a été implémenté et testé sur des fonctions choisies arbitrairement et en des points sélectionnés aléatoirement.
Dans la suite, des tests seront effectués sur des données réelles fournies par le département SINETICS d'EDF.

On verra par la suite d'autres méthodes et optimisations pouvant améliorer les performances de l'interpolation multivariée. Certaines méthodes mettent en jeu des fonctions polynomiales par morceaux.
D'autres moyens performants en relation avec le choix de la séquence de points d'interpolations seront étudiés.

7 Impressions personnelles

Au début du stage et notamment au cours des premières présentations du sujet, ce dernier m'a paru plutôt difficile. Il m'a fallu une certaine période de documentation et de recherche avant de pouvoir entamer la partie pratique. Ainsi, au bout des premières semaines de mon stage, j'ai réussi à bien assimiler la partie théorique du sujet. Ce sujet m'a paru alors encore plus intéressant car j'ai pu constater sa richesse en matière de spectre d'applications. De plus j'ai eu l'occasion de beaucoup apprendre notamment sur l'interpolation en grande dimension en général, mais aussi au niveau de la programmation informatique. En effet, j'ai pu améliorer mes connaissances en python et mettre en application tout ce que j'ai appris en algorithmique et en C++.

De plus, étant donnée que mes encadrants ont participé à l'étude et la réalisation de la partie théorique, il m'était facile d'évoluer et d'avancer dans ma compréhension des différents points-clés de mon sujet de stage. En effet, le fait qu'ils soient souvent disponibles pour discuter des points potentiellement imprécis et de l'organisation du travail m'a évité une importante perte de temps.
Par ailleurs, j'ai été agréablement surpris par l'environnement et l'esprit de travail au sein d'un laboratoire de recherche.
J'ai, de plus, apprécié la liberté qu'on m'a laissée pour le choix de l'environnement de travail et l'organisation des différentes taches et l'ordre de leurs réalisations. Ceci m'a permit non seulement d'apprendre à être autonome mais aussi d'améliorer mes compétences en analyse et en programmation.

8 Prise en compte de l'impact environnemental et sociétal

Dans un monde de plus en plus impacté par l'évolution humaine, l'écologie est un sujet souvent pris en compte de nos jours. On en parle beaucoup dans les domaines de transport, de la nutrition, de l'économie et du marketing, ...
Qu'en est-il alors des nouvelles technologies et de la recherche mathématique?

Actuellement étudiant à l'Ensimag et stagiaire dans un grand laboratoire de mathématiques appliquées, ce sujet m'a intéressé et a aussi suscité l'attention des membres de mon équipe qui m'ont aidé à enquêter là-dessus
Je travaille quotidiennement avec 4 personnes dans un bureau éclairé au néon. Nous utilisons chacun un ordinateur de bureau pendant 7h par jours.
Actuellement, le laboratoire met, à la disposition des employés, de plus en plus de moyens écologiques. Par exemple, l'éclairage s'éteint automatiquement en absence prolongée de mouvement dans la piéce, de plus les déchets papiers sont recyclés.
Quant au sujet de mon stage, les méthodes mathématique proposées qui y sont proposées permettent de réduire considérablement le coût des calculs des sections efficaces en neutronique, à précision égale, et par conséquent de limiter la consommation énergétique de gros serveurs de calcul.

References

[1] Abdellah Chkifa, Albert Cohen, and Christoph Schwab. Multidimensional interpolation and construction for Leja and RLeja points. 2013.

[2] Abdellah Chkifa, Albert Cohen, and Christoph Schwab. High-Dimensional Adaptive Sparse Polynomial Interpolation and Applications to Parametric PDEs. *Foundations of Computational Mathematics*, 14(4):601–633, 2014.

[3] Thi Hieu Luu. *Amélioration du modèle de sections efficaces dans le code de coeur COCAGNE de la chaîne de calculs d'EDF*. PhD thesis, 2017.

[4] Thi Hieu Luu, Yvon Maday, Matthieu Guillo, and Pierre Guérin. A new method for reconstruction of cross-sections using Tucker decomposition. working paper or preprint, March 2017.