Filière : Génie Énergétique & Nucléaire

Rapport de stage 2A

Étude d'une cellule d'UO2 en Burn-Up sous DRAGON5

Auteur :

GERRIER Tom
tom.gerrier@grenoble-inp.org
Année Univeritaire 2018-2019

Encadrant :

Pr. HEBERT Alain
alain.hebert@polymtl.ca
École Polytechnique de Montréal

Remerciements

Je tiens tout d'abord à remercier le professeur Alain HEBERT, pour m'avoir permis de réaliser ce stage au sein de l'École Polytechnique de Montréal. Ce stage a été très enrichissant car il m'a permis d'avoir une première approche avec la simulation numérique de système nucléaire, ainsi que de découvrir les codes Dragon et Serpent.
Je souhaite également remercier Maxime GOULOUX et Rédouane IDHIME, avec qui j'ai partagé un bureau et échangé sur nos projets respectifs pendant ces 10 semaines.

Table des matières

Glossaire

EPR : European Pressurised Reactor

REP : Réacteur à Eau Pressurisée

REB : Réacteur à Eau Bouillante

CGN : China General Nuclear Power Corporation

NEA : Nuclear Energy Agency

LPSC : Laboratoire de Physique Subatomique et de Cosmologie

Jeu de données : Ensemble de fichiers permettant le fonctionnement d'un schéma de calcul.

Burn-Up : Taux de combustion, caractérise la quantité de combustible utilisé pendant un certain temps, sous une certaine puissance.

Benchmark : Document contenant un ensemble de données concernant un système, permettant la modélisation et l'étude de ce dernier.

Tracking : Discrétisation angulaire du système par Dragon.

Table des figures

Liste des tableaux

Introduction

Aujourd'hui, les nouveaux réacteurs nucléaires se voient dotés de nombreux systèmes de sécurité et de protection face aux possibles incidents nucléaires. C'est dans cette optique que l'EPR (European Pressurised Reactor) a été conçu et développé. Actuellement, plusieurs réacteurs de type EPR sont en construction dans le monde, notamment deux réacteurs, à Taishan en Chine. Ce projet est mené par l'industriel chinois CGN (China General Nuclear Power Corporation). Afin d'assurer une sécurité maximale, CGN a besoin d'utiliser des outils précis et fiables, notamment leur code de calcul de réseau. Le constructeur chinois a donc besoin de références. Un partenariat avec l'École Polytechnique de Montréal a alors été mis en place. Cet accord consiste en la validation du code Dragon5 grâce à différents benchmarks, afin que CGN puisse utiliser Dragon5 comme référence pour pour les codes chinois.

L'une des caractéristiques importantes à vérifier, est si le code Dragon5 est fiable pour modéliser des systèmes sur de long burn-up, c'est-à-dire avec une longue utilisation du combustible.

L'un des benchmarks commandés par CGN, le benchmark "*Yamamoto*", propose l'étude de systèmes type REP/REB (Réacteur à Eau Pressurisée/Bouillante) avec un burn-up allant jusqu'à 70GWj/t. Ce projet se concentre donc sur la validation du code Dragon5, via l'étude de l'évolution de certaines grandeurs d'une cellule d'UO2 au cours du burn-up. Ces résultats seront mis en comparaison avec ceux obtenus par le code Monte Carlo Serpent2, utilisé comme référence.

Ce rapport présentera tout d'abord le benchmark "*Yamamoto*", ainsi que les différents outils numériques utilisés lors du projet. Puis les méthodes et schémas de calcul élaborés seront détaillés. Enfin, les résultats obtenus seront présentés et analysés, avant de conclure sur la validité des schémas de calcul mis en place.

Partie 1

Présentation du benchmark "*Yamamoto*"

Le benchmark "*Yamamoto*" [1] propose l'étude de système utilisant de l'eau légère comme modérateur, avec un burn-up de 70 GWj/t. Plusieurs géométries sont proposées : cellule, assemblage type REP et assemblage type REB. Plusieurs types de combustible sont également proposés : de l'UO2 et du MOX. Seule une partie du benchmark est traitée dans ce rapport : la cellule d'UO2.

1.1 Géométrie

Le système étudié étant une cellule de combustible, sa géométrie est simple. Il s'agit d'un carré de modérateur (ici de l'eau légère), dans lequel est plongé un disque de combustible (ici de l'UO2) entouré d'une gaine en Zirconium.

Afin de pouvoir appliquer les méthodes d'auto-protection sous Dragon, et traiter le burn-up, il est nécessaire de diviser le combustible [2]. Ainsi, l'UO2 est divisé en 4 couronnes, représentant respectivement 50%, 30%, 15% et 5% du volume total de celui-ci. De plus, les conditions aux limites de chaque frontières seront traitées comme de la réflexion.

La Figure 1.1 ainsi que le Tableau 1.1 ci-dessous permettent de visualiser clairement la géométrie et les dimensions de la cellule.

Abréviation	**Description**	**cm**
A	Largeur de la cellule	1.265
B	Diamètre du combustible	0.824
C	Diamètre intérieur de la gaine	0.824
D	Diamètre extérieur de la gaine	0.952

TABLE 1.1 – Tableau des dimensions de la cellule

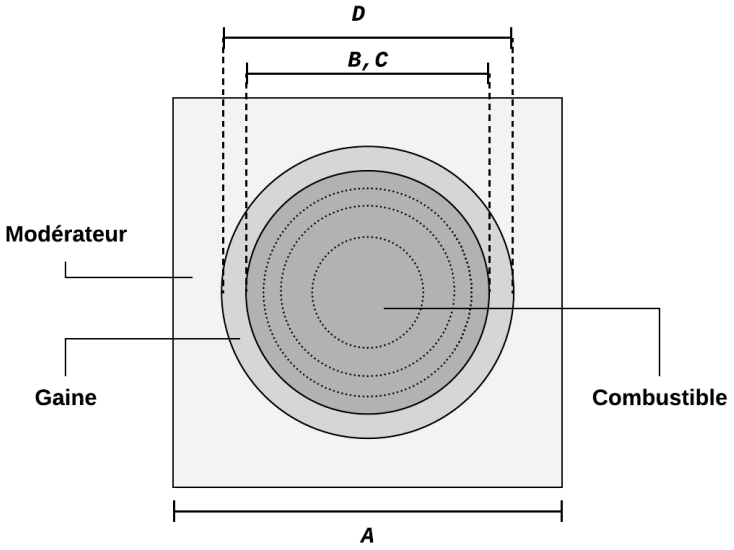

FIGURE 1.1 – Schéma de la géométrie de la cellule

1.2 Composition

La composition isotopique de la cellule est proche d'une cellule d'UO_2 classique, que l'on retrouve dans les REP. La différence majeure réside dans le pourcentage d'enrichissement en Uranium 235. Pour une cellule de REP classique, celui-ci est d'environ 3 à 4%. Ici, l'Uranium est enrichi à 6.5% en Uranium 235, afin d'assurer le fonctionnement de la cellule pour un burn-up de 70 GWj/t.

Le tableau 1.2 regroupe la composition isotopique des différents matériaux présents dans la cellule.

	Combustible (900K)		
Isotope	U235	U238	O16
Densité (barn/cm)	1.2122E-03	2.1477E-02	4.5945E-02

	Gaine (600K)
Isotope	Zr Naturel
Densité (barn/cm)	4.3107E-02

	Modérateur (600K)	
Isotope	H1	O16
Densité (barn/cm)	4.4148E-02	2.207E-02

TABLE 1.2 – Tableaux des compositions isotopiques des matériaux de la cellule

1.3 Résultats attendus

Tout d'abord, cette étude est réalisée avec une densité de puissance de **37.0W/g**. Le burn-up allant jusqu'à 70 GWj/t, cela correspond à une utilisation du combustible pendant 1891 jours, soit 5 ans et 2 mois. L'objectif de cette étude est de déterminer l'évolution de certains paramètres au cours de cette utilisation. Les paramètres dont l'évolution est à déterminer sont fournis par le benchmark [1] et sont les suivants :

- Facteur de multiplication effectif $\mathbf{k}_{eff}$.

  avec :

$$k_{eff} = \frac{\text{Nombre de neutrons produits}}{\text{Nombre de neutrons disparus}}$$

Or les conditions aux limites imposent de la réflexion sur chaque paroi de la cellule. Ainsi, un neutron disparu est nécessairement absorbé.

- Abondance isotopique de certains noyaux lourds et produits de fission, en barn/cm.

Noyaux lourds : U235, U236, U238, Np237, Pu238, Pu239, Pu240, Pu241, Pu242, Am241, Am242m, Am243, Cm242, Cm243, Cm244, Cm245, Cm246

Produits de fission : Mo95, Tc99, Rh103, Cs133, Sm147, Sm149, Sm150, Sm152, Nd143, Nd145, Eu153, Gd155

- Section efficace de production de neutrons $\nu\sigma_f$, en barn, de certains noyaux lourds.

Avec ν : le nombre moyen de neutrons produits par fission et σ_f : la section efficace microscopique de fission.

Noyaux lourds : U235, U238, Pu238, Pu239, Pu240, Pu241, Pu242, Am241, Am243, Cm242, Cm244

Partie 2

Outils numériques utilisés

Afin de pouvoir étudier le benchmark présenté précédemment, il est également nécessaire de présenter les outils numériques utilisés pour ce projet. Ils peuvent être séparés en deux catégories : les codes de calcul nucléaires, et les outils plus généraux.

2.1 Codes de calcul

L'obtention des résultats importants, demandés par le benchmark, est réalisée via l'utilisation de codes de calcul. Ces outils sont utilisés pour la partie de modélisation et de simulation de la cellule de combustible. Pour ce projet, deux codes différents sont utilisés : Dragon 5 et Serpent 2.

2.1.1 Dragon 5

Dragon est un code de calcul de réseau, déterministe, permettant la simulation de systèmes variés, grâce à sa grande flexibilité. Il s'agit d'un code en sources ouvertes, développé au sein de l'École Polytechnique de Montréal et actuellement en Version 5.

Dragon fonctionne grâce à différents modules, effectuant chacun une étape de calcul, avec un large choix d'options. Ces modules manipulent des structures de données contenant des informations sur le système modélisé (géométrie, composition isotopique, sections efficaces, etc). Les échanges entre les différents modules et ces structures données est assuré par le noyau du code : *GANlib*, et son langage *CLE2000*, également développés à l'École Polytechnique de Montréal.

La Figure 2.1 ci-dessous permet de visualiser la manière dont les structures de données (ellipses) et les modules de Dragon (rectangles) s'organisent pour aboutir au fichier "*.result*".

L'objectif de l'utilisation de Dragon pour ce projet est donc la production d'un fichier "*.result*", au sein duquel se trouvent les résultats attendus par le benchmark.

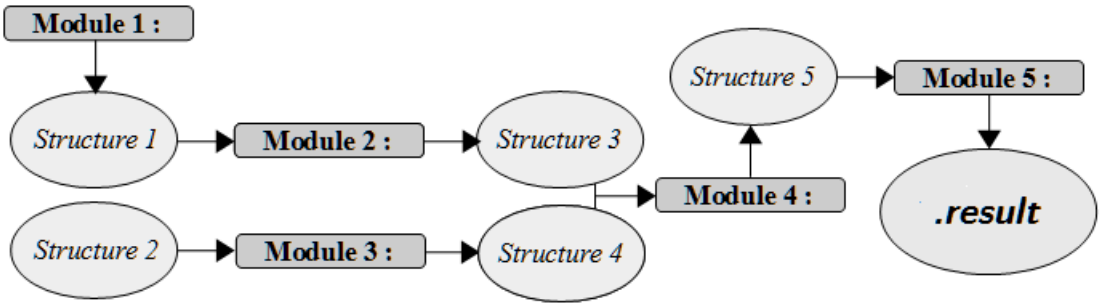

FIGURE 2.1 – Schéma d'organisation des structures de données et modules dans Dragon

2.1.2 Serpent 2

Serpent est un code de calcul Monte Carlo, développé par le Centre de recherche technique VTT de Finlande. Il est actuellement dans sa version 2 [3]. Ce code fonctionne selon le principe suivant : il prend en entrée les différents paramètres caractéristiques du système étudié (géométrie, composition) et simule, de manière individuelle, la vie d'un grand nombre neutrons évoluant au sein de ce système. Un traitement statistique est ensuite réalisé afin d'en déduire les paramètres représentatifs.

S'agissant d'un code Monte Carlo, Serpent introduit moins d'approximations et de biais dans ses méthodes qu'un code déterministe. Les résultats obtenus sont alors d'une grande précision, dès lors que la statistique, donc le nombre de neutrons simulés est suffisant. Son utilisation a donc pour objectif de produire des données de référence, afin qu'elles puissent être comparées à celles obtenues avec Dragon. Cette comparaison permet alors de valider ou de rejeter les résultats fournis par le code déterministe.

2.2 Outils de récupération et de traitement des résultats

Les codes présentés précédemment permettent d'obtenir un grand nombre d'informations sur la cellule étudiée. Ces résultats sont contenus dans des fichiers dont il faut extraire les bonnes données, afin de les traiter et de les analyser. Pour se faire, plusieurs outils numériques sont utilisés : Python et Matlab.

2.2.1 Python

Les informations utiles pour le benchmark sont contenues au sein de plusieurs centaines de milliers de lignes. Récupérer ces résultats à la main serait un travail laborieux, où des erreurs peuvent facilement être commises. L'élaboration d'un

processus de sélection et de récupération des données est donc nécessaire.

Python étant un langage de programmation polyvalent et capable de manipuler des fichiers texte (lecture et écriture), son utilisation permet la réalisation de ce processus.

2.2.2 Matlab

Mettre en forme les résultats obtenus est également une part importante du travail. Les présenter sous forme d'un tableau ne serait pas d'une grande lisibilité, compte tenu du grand nombre d'information demandées par le benchmark, ainsi que du grand nombre d'étape de burn-up réalisées. L'utilisation de graphique permet de regrouper un grand nombre d'informations, tout en assurant une bonne lecture des résultats.

Le logiciel Matlab propose un grand nombre d'options pour la création et l'édition de graphiques, que ne proposerait pas un tableur classique comme Microsoft Excel ou LibreOffice Calc. Son utilisation est donc privilégié pour la création des graphiques, afin de permettre une meilleure lisibilité, et donc une meilleure compréhension des résultats.

Partie 3

Méthodes et approches utilisées

Chaque outil numérique présenté précédemment est donc utilisé dans un but précis. Cependant, les manières d'arriver aux résultats souhaités sont multiples, et plusieurs approches sont alors possibles.

3.1 Approche directe

Une première approche consiste en l'implémentation directe et totale du benchmark dans Dragon et Serpent. C'est-à-dire, implémenter tous les paramètres (géométrie, composition) et toutes les étapes de calcul en même temps (autoprotection, calcul du flux, burn-up).

3.1.1 Dragon

Jeu de Données

Pour cette approche directe, le schéma de calcul implémenté est basé sur le travail effectué par Richard Chambon (LPSC Grenoble). En effet, les données du benchmark ont déjà été implémenté sous Dragon. Néanmoins, certaines modifications restent à être mise en place.

Le jeu de données développé par R.Chambon inclus la possibilité de choisir parmi plusieurs bibliothèques de sections efficaces (*JEFF3.1-shem285/361, ENDFB6, COSN70*) ainsi que parmi plusieurs méthodes de calcul de l'autoprotection (module **USS** :, module **SHI :**). Cependant, chacun des modules n'est pas compatible avec chacune des bibliothèques, et certaines de ces bibliothèques sont actuellement dépassées et fournissent donc de mauvais résultats. Il faut mettre à jour le jeu de données.

Démarche suivie

La mise à jour du jeu de données consiste tout d'abord par la mise à jour des bibliothèques proposées. Ainsi, les librairies *COSN70* et *ENDFB6* ont été retirées (car trop anciennes) et les bibliothèques *JEFF3.1-shem295* et *XMAS172* ont été

ajoutées. Il est important de préciser que toutes les bibliothèques JEFF utilisées pour ce projet sont en *version 3.1*, car il s'agit de la version à privilégier pour des comparaisons entre Dragon et Serpent

Ainsi, chaque bibliothèque proposée est maintenant à jour et compatible avec une des méthodes d'autoprotection implémentées. Les correspondances entre bibliothèque et module sont explicitées dans le tableau 3.1.

Module / Bibliothèque	**SHI :**	**USS : SUBG**	**USS : PT**
SHEM281	oui	oui	non
SHEM295	non	non	oui
SHEM365	non	non	oui
XMAS172	non	oui	non

TABLE 3.1 – Tableau des compatibilités Bibliothèque/Module Autoprotection dans Dragon5

Sélection Bibliothèque/Module Auto-protection : L'utilisateur choisit une bibliothèque et un module d'autoprotection compatibles entre eux. Le tableau 3.1 récapitule les différentes combinaisons possibles.

Définition de la géométrie : A l'aide du module **GEO :**, Dragon définit la géométrie du système. Il définit donc un carré contenant 5 couronnes. La cellule est alors séparée en 6 régions, ou MIXs, correspondant au combustible divisé en 4 (MIXS 1, 4, 5 et 6), à la gaine (MIX 2) et au modérateur (MIX 3). Les différentes dimensions (rayons et largeur) sont rappelées dans le tableau 1.1. Les conditions aux limites sont également définies : il y a réflexion sur chacune des 4 frontières.

Tracking : A partir de la géométrie définies dans l'étape précédente, Dragon réalise un tracking grâce au module **SYBILT :**. Cette étape correspond à la discrétisation angulaire de la cellule. Pour cela, Dragon réalise un balayage de la géométrie, en définissant plusieurs lignes d'intégration, selon plusieurs angles et avec une densité de ligne par cm fixée.

Définition de la composition isotopique : Grâce au module **LIB :** Dragon attribue à chaque région définie une composition, c'est-à-dire la liste des isotopes présents avec leur densité respective ainsi que la référence correspondante dans la bibliothèque de sections efficaces sélectionnée. La température de chaque région est également définie. La distinction entre les méthodes d'auto-protection "USS_SUBG" et "USS_PT" se fait ici, grâce au mot clé "$SUBG$" ou "PT".

Calcul de l'auto-protection : Durant cette étape, Dragon réalise le calcul d'auto-protection des résonances. Le module utilisé dépend de la méthode sélectionnée par l'utilisateur au début du calcul. Trois méthodes sont im-

plémentée dans ce schéma de calcul :

Méthode de Stammler : Cette méthode correspond à l'utilisation du module **SHI :**. Dragon réalise une homogénéisation équivalente de la cellule pour chaque groupe d'énergie, et calcule les nouvelles sections efficaces.

Méthode des Sous-Groupes : Cette méthode correspond à l'utilisation du module **USS :**. Elle se base sur l'utilisation de tables de probabilités. Ces tables sont différentes selon si l'utilisateur sélectionne l'option "$SUBG$" ou "PT". La première correspond à des tables de probabilités physiques, et la seconde à des tables mathématiques.

Les détails de chacune de ce méthodes sont explicités dans le Guide d'utilisation de Dragon5. [4]

Assemblage des matrices : A partir des nouvelles sections efficaces autoprotégée, Dragon utilise le module **ASM :** pour créer les matrices nécessaires au calcul du flux lors de l'étape suivante.

Calcul du flux : Avec le module **FLU :**, Dragon réalise ici la résolution de l'équation du transport, à partir des matrices définies précédemment et du tracking. La méthode de résolution dépend de l'option sélectionné, ici la résolution est de "*type K*", ce qui signifie que Dragon traite un problème aux valeurs propres. Tout comme pour les méthodes d'auto-protection, les détails de la méthode de résolution sont explicités dans le Guide d'utilisation de Dragon5.[4]

Édition du fichier résultat : Grâce au module **EDI :**, Dragon réalise plusieurs opérations spécifiques, dont il inscrit le résultat dans le fichier *".result"*. Dans ce cas, il condense et homogénéise les sections efficaces de toutes les réactions nucléaires (Fission, N2N, N3N, etc), pour chaque isotope requis parle benchmark.

Partie Burn-Up : Cette partie du schéma de calcul est répété pour chaque valeur de burn-up définit par l'utilisateur (ici de 0 à 70 GWj/t, avec un total de 52 étapes).

Tout d'abord, avec le module **EVO :**, Dragon résout les équations d'évolution de la densité de chaque isotope. Ainsi de nouvelles densités et d nouvelles sections efficaces sont calculées. Ces données sont ensuite utilisées pour réaliser le calcul d'auto-protection, l'assemblage des matrices et la résolution du flux, avec les mêmes modules et options que dans la partie à burn-up 0. Il réalise ensuite les opérations indiquées par le module **EDI :** et met à jour le fichier "*.résult*".

La figure 3.1 permet de visualiser comment les différents modules s'enchaînent pour former le schéma de calcul.

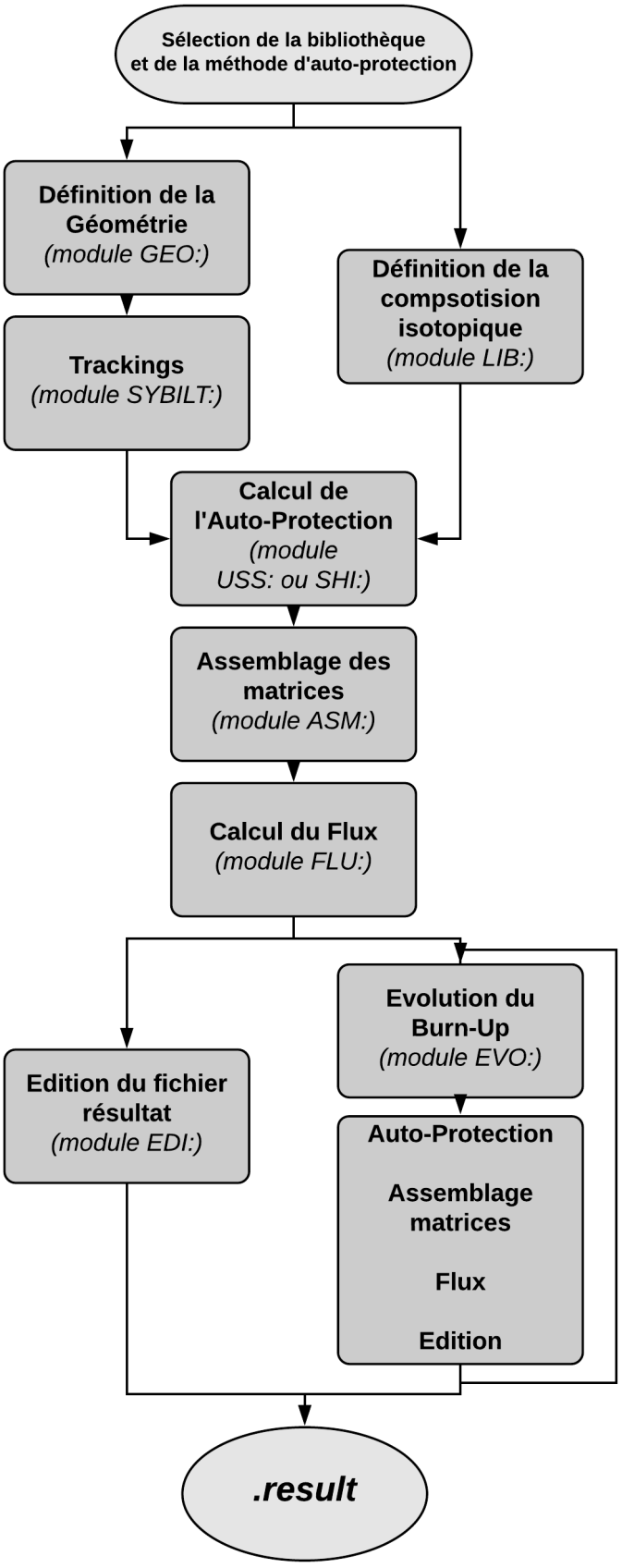

FIGURE 3.1 – Schéma de calcul de l'approche directe dans Dragon5

3.1.2 Serpent

Jeu de Données

De même que pour le schéma de calcul sous Dragon, celui implémenté sous Serpent est basé sur un jeu de données déjà existant. Il s'agit de l'étude d'une cellule de combustible en burn-up. Ce cas est donc proche du système étudié pour ce projet, et ne nécessite que peu de modifications.

Démarche suivie

Le système étudié dans ce jeu de données étant très similaire au cas proposé par le benchmark, seuls les paramètres de géométrie et de composition sont à modifier.

Tout d'abord, les bibliothèques sont modifiées pour correspondre aux bibliothèques utilisées par Dragon (voir tableau 3.1, elles sont maintenant toutes en *version 3.1*. Les valeurs de géométrie et de compositions sont ensuite modifiées, tout comme les valeurs liées au burn-up (densité de puissance, liste des isotopes à considérer, liste des étapes de burn-up à réaliser).

Les différentes étapes du schéma mis en place sont alors les suivantes :

Définition de la géométrie & des matériaux : Serpent définit la géométrie du système et associe à chaque matériaux sa composition isotopiques, sa température et indique si le matériaux s'épuise ou non lors du burn-up.

Définition des paramètres de simulation : L'utilisateur indique à Serpent les paramètres liés à la simulation, ainsi ce schéma de calcul comporte *400 cycles de 400 neutrons* chacun. Les options de maillages sont également renseignées : un maillage de 500 par 500 est alors mise en place par Serpent.

Définition des paramètres de détection : Serpent permet d'obtenir des valeurs de sections efficaces pour différentes réactions. Pour cela, il faut indiquer à Serpent les matériaux dont il doit déterminer les sections efficaces, mais aussi les énergie auxquelles elles doivent être mesurée, ainsi que le type de réaction en question. Pour ce projet, seules les sections efficaces de production de neutrons sont requises. Les matériaux renseignés sont les isotopes listés dans la section 1.3.

Définition des paramètres de Burn-Up : Afin de pouvoir traiter l'évolution du combustible, Serpent a besoin de connaître la liste des isotopes concernés par l'évolution, les différentes étapes de cette évolution, ainsi que la densité de puissance.

La figure 3.2 reprend l'organisation du schéma de calcul.

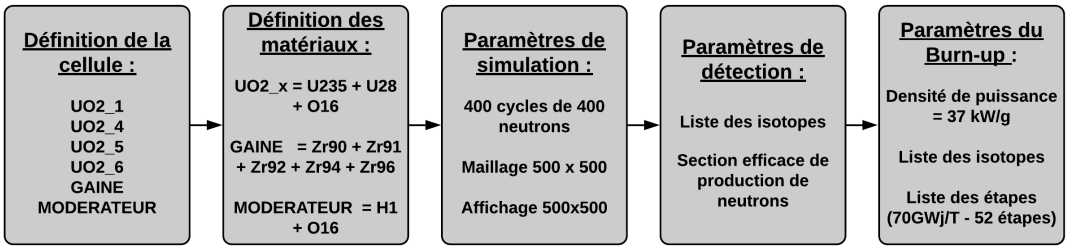

FIGURE 3.2 – Schéma de calcul de l'approche directe dans Serpent2

3.2 Approche progressive

Une seconde approche pour l'étude de la cellule consiste en une implémentation progressive du benchmark. Pour se faire, un ensemble de jeux de données est utilisé comme base de cette méthode.

3.2.1 Jeux de Donnée

Le jeu de données utilisé est un benchmark pour cellule de REP, développé par la Nuclear Energy Agency (NEA). Il a été développé afin de comparer différents codes entre eux. Il s'agit donc d'une excellente base pour la comparaison de Dragon5 et Serpent2 sur l'étude du benchmark Yamamoto.

Dragon

Sous Dragon, le benchmark est implémenté sous la forme d'un test de non-régression. C'est-à-dire un cas test permettant de vérifier si Dragon obtient toujours les mêmes résultats sur un cas test, lors d'une mise à jour. Il est composé de 10 cas différents, 5 utilisant un combustible UOx et 5 utilisant un combustible MOx, pour différentes températures et densités.

Seul un de ces 10 cas est sélectionné comme base de l'étude. Il s'agit du cas UOx numéro 3, car il correspond à une cellule avec un combustible à 900K, tout comme le benchmark Yamamoto.

La méthode d'auto-protection utilisée est la méthode "USS_PT" et le flux est calculé grâce à la méthode de *type K*, tout comme dans l'approche directe.

Serpent

De manière équivalente, les 10 cas précédents sont également implémenté sous Serpent 2, la configuration utilisée comme base est la configuration "uo2_3".

3.2.2 Démarche suivie

Comme mentionné précédemment, les modifications des schémas existants sont faites de manière progressive. Une vérification de la cohérence des résultats entre Dragon et Serpent est effectuée entre chaque étape de modification. Cette vérification se fait en comparant la valeur du k_{eff} fourni par les deux codes. Si l'écart entre les deux valeurs est cohérents, alors la modification suivante peur être implémentée.

La démarche mise en place est alors la suivante :

Vérification de départ : Avant d'entreprendre une quelconque modification, il faut vérifier que les schémas de calcul de base fournissent toujours des valeurs cohérentes. Les bibliothèques de sections efficaces utilisées sont les bibliothèques *JEFF 2.2 à 295 groupes*.

JEFF2.2 à JEFF3.1 La première modification consiste à passer toutes les bibliothèques utilisées en ***JEFF 3.1 à 295 groupes***. Il s'agit de la meilleure version pour les comparaisons entre Dragon et Serpent.

Implémentation de la géométrie La géométrie des deux schémas est modifiées pour correspondre à celle du benchmark Yamamoto.

Sous Dragon, le combustible est séparé en 6 et le modérateur en 2. On ramène alors le nombre de couronne dans l'UO2 à 4, avec les bons rayons, et le modérateur est réduit à une seule région. Les densités restent inchangées.

Sous Serpent, le combustible n'est pas divisé, on implémente alors les 4 couronnes. Les densités restent inchangées.

Modifications individuelles des densités : Les densités des différents isotopes de la cellule étudiée sont implémentées de manière progressive elles aussi. Cela est réalisé en 3 étapes : les densités de chaque matériaux (combustible, gaine, modérateur) sont modifiées individuellement afin de déterminer si l'une de ces 3 modifications engendre un problème.

Modification globale des densités : Une fois les vérifications individuelles réalisées, les densités des 3 matériaux sont implémentées sous Dragon et Serpent. Ainsi la géométrie et la composition du benchmark Yamamoto sont entièrement implémentée sous Dragon, comme sous Serpent.

Les différents résultats obtenus après chaque étape de modification sont regroupé dans le tableau 3.2 :

	k_eff Dragon 5	k_eff Serpent 2	Ecart (pcm)
Vérification de départ	1,303833	1,307170	-333,7
JEFF 2.2 à JEFF 3.1	1,298978	1,301810	-283,2000
Implémentation de la géométrie	1,307713	1,308880	-116,7000
Densité Combustible	1,432716	1,434010	-129,4000
Densité Gaine	1,307768	1,309360	-159,2000
Densité Modérateur	1,296000	1,296500	-50,0000
Burn-Up 0	1,419247	1,420670	-142,3000

TABLE 3.2 – Tableau des résultats intermédiaires de l'approche progressive

Les écarts obtenus sont compris entre 330 et 50 pcm selon l'étape, ce qui correspond à un écart cohérent, compte tenu de la grande sensibilité du facteur de multiplication effectif.

Partie 4

Résultats obtenus

4.1 Récupération des données importantes

Le traitement et l'analyse des résultats obtenus passent avant-tout par la récupération de ces derniers.

4.1.1 Fichiers de sortie

Afin de mettre au point un algorithme de récupération des résultats, il est nécessaire de comprendre quels sont les fichiers contenants les résultats et où se trouvent les informations utiles au projet au sein de ces fichiers.

Dragon

Les résultats importants fournis par Dragon sont situés dans plusieurs fichiers différents. Les informations sur le facteur de multiplication effectif k_{eff} ainsi que les valeurs de sections efficaces sont situées dans le fichier "*.result*" généré puis sauvegardé à la fin de l'exécution du code. Les différentes valeurs de densités sont stockées dans un fichier appelé "*_BURN_ rowlands.txt*", dans la section "*ISOTOPEDENS*" [5].

Serpent

Serpent crée plusieurs fichiers contentant les informations importantes. Les valeurs du facteur de multiplication k_{eff} sont stockées dans le fichier ".m", il s'agit du fichier de résultat principal.

Pour chaque étape du burn-up, Serpent crée un fichier "_det*xx*.m"' et un fichier "_bumat*xx*.m" avec *xx* le numéro de l'étape. Le premier fichier contient les valeurs de sections efficaces, et le deuxième les valeurs de densités isotopiques.

4.1.2 Méthode d'extraction des données

La méthode utilisée pour extraire les données des fichiers s'appliquent à chacun des fichiers mentionnés précédemment. Elle comporte les étapes suivantes :

Parcours des lignes du fichier : Python ouvre le fichier contenant les résultats et le parcours ligne par ligne.

Sauvegarde des données importantes : Lorsqu'une des informations importantes (valeur de k_{eff}, de densité ou de section efficace), Python la lit et la stocke dans une liste comportant tous les résultats lus et sauvegardés, mais sans qu'ils soient triés.

Regroupement des résultats par isotope : Cette partie ne concerne pas la récupération des données du k_{eff}, puisqu'il ne dépend pas l'isotope, mais est global. Les différentes données de la liste précédente sont ensuite triées, en étant répartie dans un liste 2D, chaque élément de cette liste est une liste contenant le nom de l'isotope et les valeurs associées.

Homogénéisation des résultats : Pour chaque isotope et chaque étape du burn-up, Python réalise une moyenne des4 valeurs correspondant chacun à une couronne du combustible, avec les proportions suivante : 50%, 30%, 15% et 5%.

Écriture des résultats : Python sauvegarde les valeurs récupérées et homogénéisées en les écrivant dans un fichier texte, qui pourra ensuite être exploité sous MatLab.

La figure 4.1 schématise les différentes étapes de l'algorithme.

FIGURE 4.1 – Schéma de l'algorithme de récupération des données sous Python

4.2 Burn-Up 0

Avant d'étudier les résultats obtenus lors de l'évolution du Burn-Up, il est nécessaire d'étudier les résultats "*à Burn-Up 0*", c'est-à-dire avant que le combustible ne commence à s'user. Cette vérification est le point de départ de l'analyse des résultats.

4.2.1 Facteur de multiplication

Le paramètre permettant de valider ou non une approche est le facteur de multiplication effectif k_{eff}. Il s'agit du paramètre décrivant le système dans son ensemble, c'est donc un résultat important qui doit être cohérent avant d'étudier le système en évolution.

Le tableau 4.1 regroupe les résultats obtenues grâce aux deux approches :

	k_eff Dragon 5	**k_eff Serpent 2**	**Ecart (pcm)**
Approche Directe SHI_281	1,412113	1,508670	-9 655,7
Approche Directe USS_SUBG_281	1,413943	1,508670	-9 472,7
Approche Directe USS_SUBG_172	1,413581	1,508670	-9 508,9
Approche Directe USS_PT_295	1,415150	1,508670	-9 352,0
Approche Directe USS_PT_361	1,414784	1,508670	-9 388,6
Approche Progressive USS_PT_295	1,419247	1,420670	-142,3

TABLE 4.1 – Tableau comparatif des facteurs de multiplication effectif à Burn-Up 0

4.2.2 Rejet de l'approche directe

D'après le tableau 4.1, l'approche directe donne des écarts supérieurs à 9000 pcm, et ceux pour chaque combinaison Bibliothèque/Module auto-proctection. Il s'agit là d'écarts bien trop importants pour que les résultats obtenus par Dragon puissent être validés. Le schéma de calcul développé avec l'approche directe est donc rejetée.

En revanche, le schéma de calcul mis en place par l'approche progressive donne un écart de 142 pcm. Les résultats présentés par la suite seront ceux obtenus avec ce schéma de calcul.

4.3 Évolution des résultats en fonction du Burn-Up

Les résultats à Burn-Up 0 étant cohérents, ceux obtenus lors de l'évolution du combustible peuvent être analysés.

4.3.1 Facteur multiplication effectif

Tout comme précédemment, le premier paramètre à étudier est le facteur de multiplication effectif k_{eff}, car il décrit le système dans son ensemble. Des résultats cohérents sur son évolution sont donc nécessaires pour valider l'étude. Les résultats obtenus, ainsi que les divergences ($k_{eff/Dragon}$ - $k_{eff/Serpent}$) sont présentés sur la figures 4.2 et la figures 4.3.

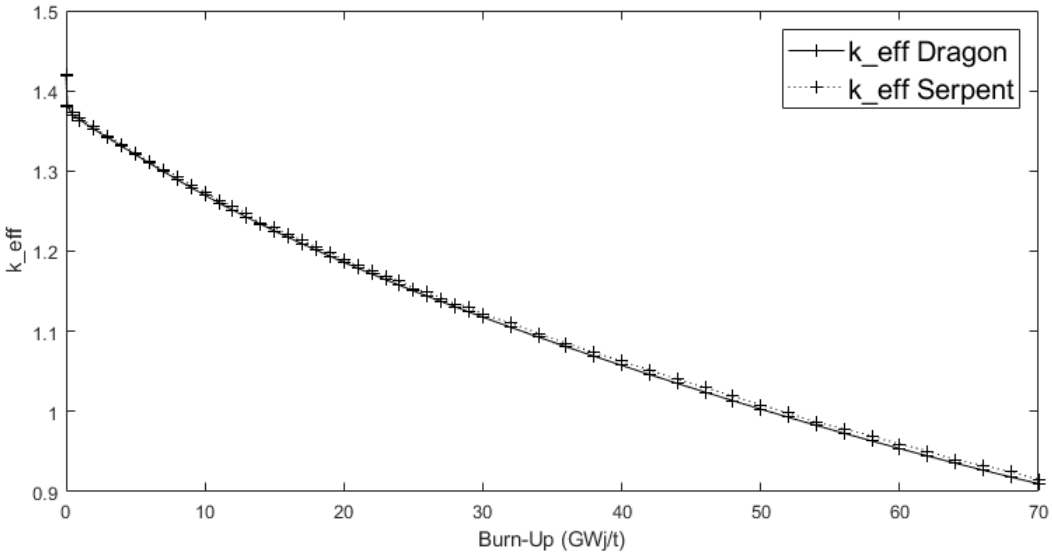

FIGURE 4.2 – Evolution du Facteur de Multiplication Effectif k_{eff} en fonction du Burn-Up

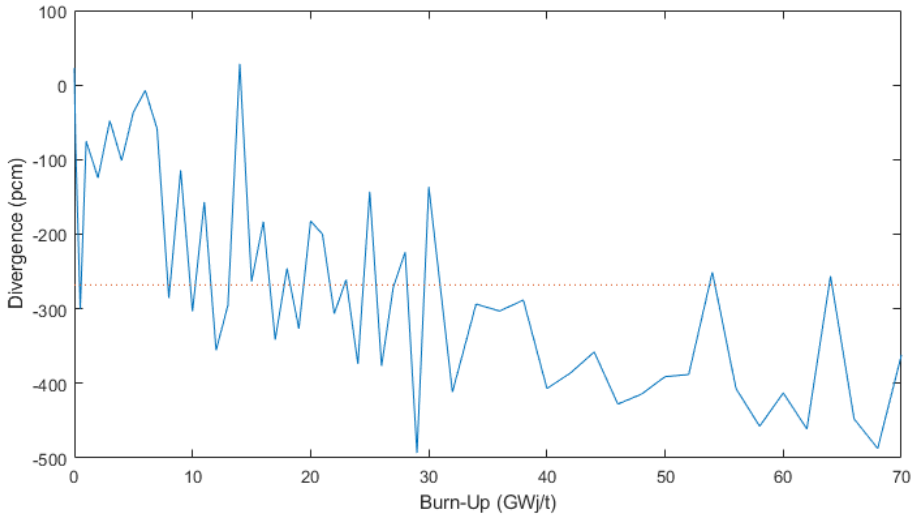

FIGURE 4.3 – Divergences en fonction du Burn-Up

Sur la figure 4.2, est représentée l'allure de l'évolution du facteur de multiplication en fonction du burn-up. On observe une décroissance d'environ 1,42 à 0,91

pour les deux codes, avec un passage du régime surcritique à sous-critique pour un burn-up de 50 GWj/t. La décroissance observée s'explique par le fait que plus le combustible s'épuise, moins il y a de noyaux produisant des neutrons. Il y a donc de moins en moins de neutrons produits par rapports aux neutrons absorbés : le facteur de multiplication effectif diminue.

La figure 4.3 présente les divergences, c'est-à-dire l'écart absolu entre la valeur calculée par Dragon et celle calculée par Serpent, en pcm. On observe que Dragon a tendance à sous-évaluer par rapport à Serpent, un écart moyen de 200 pcm au cours de l'évolution. Il s'agit d'un écart convenable pour une comparaison entre un code déterministe et un code Monte Carlo.

4.3.2 Densités isotopiques

L'étude de l'évolution du facteur de multiplication effectif étant validée, les autres données peuvent être étudiées.

Les figures 4.4 à 4.7 représentent l'évolution des densités isotopiques des noyaux lourds. Les figures 4.8 et 4.9 correspondent à l'évolution de la densité des différents produits de fissions. Les valeurs représentées sont les densités normalisées, c'est-à-dire divisées par leur valeur maximale, afin qu'elles puissent être représentées sur un même graphique, sans problèmes d'échelle.

Tout d'abord, on observe sur la figure 4.4 une diminution de la quantité en *U235* et *U238*, ainsi qu'une augmentation de la quantité des autres noyaux lourds et produits de fission sur les autres figures. Ceci qui montre bien une usure du combustible, et une production d'autres isotopes par différentes réactions nucléaires (fission, bêta, alpha) au cours de l'évolution.

Les deux codes donnent des résultats presque identiques, avec un écart absolu variant de -0.7 pcm à 0.4 pcm. Chaque valeur calculée par Dragon est donc égale à la valeur de référence au pcm près. Le code Dragon 5 donne donc de très bons résultats concernant l'évolution des densités isotopiques au cours du burn-up, même long.

4.3.3 Sections efficaces de production de neutrons

Les figures 4.10 et 4.11 représentent l'évolution des sections efficaces de production de neutrons, condensées pour chaque isotope entre 0.11meV et 19.6 MeV, au cours du burn-up.

Concernant les sections efficaces, les résultats sont différent de 13 ordres de grandeurs. Cet écart est évidement trop importants pour valider les données, mais également trop importants pour rejeter l'approche progressive, compte tenu des

résultats obtenus concernant les densités et valeurs du k_{eff}.

Les valeurs obtenues par Dragon correspondent à des ordres de grandeur cohérents (Pu239 et U235 largement supérieurs aux autres, U238 avec la section la plus faible) avec des sections efficaces comprises entre 0,5 et 100 barns. Serpent fournit des sections efficaces comprises entre 10^{12} et $10^{1}5$ barns. Il s'agit d'ordres de grandeurs beaucoup trop importants pour correspondre à des valeurs de sections efficaces microscopiques. On peut donc penser que ces écarts sont dûs à des erreurs dans le paramétrage de la partie "Détecteurs" sous Serpent2, comme une mauvaise plage d'énergie, ou alors une erreur dans l'homogénéisation durant l'algorithme de récupération Python. Cette étape du schéma de calcul doit donc être reprise afin d'obtenir des valeurs cohérentes, qui puissent être comparées aux sections efficaces calculées par Dragon.

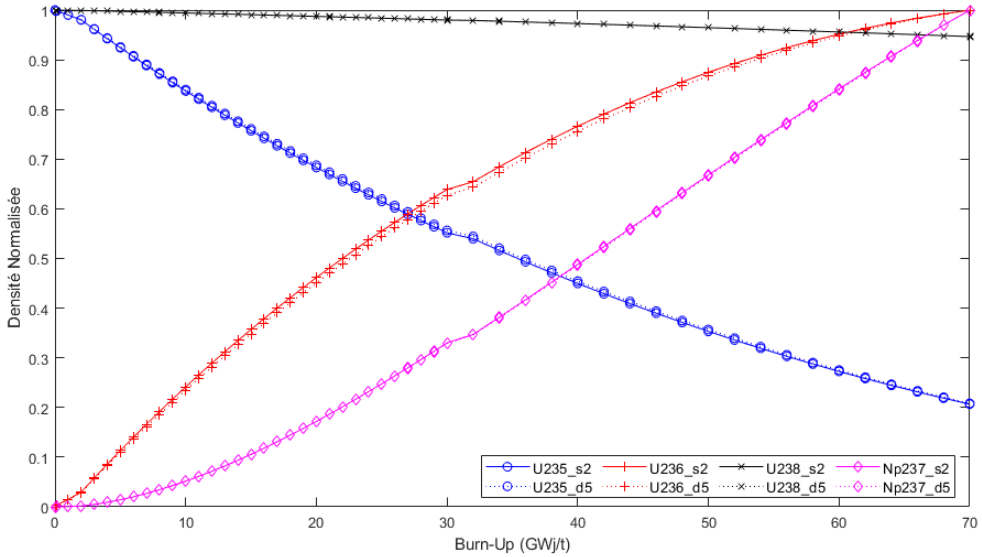

FIGURE 4.4 – Évolution de la densité isotopique de l'Uranium et du Neptuninum en fonction du Burn-Up

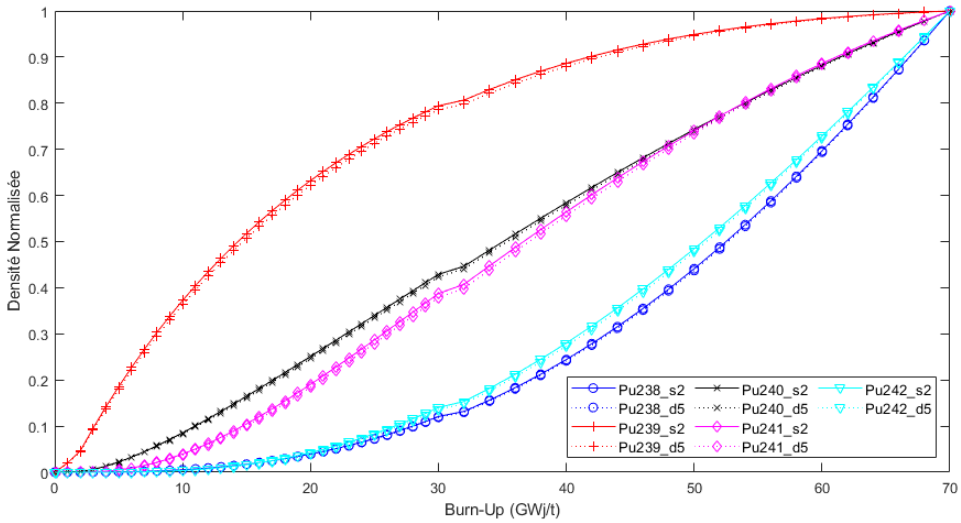

FIGURE 4.5 – Évolution de la densité isotopique du Plutonium en fonction du Burn-Up

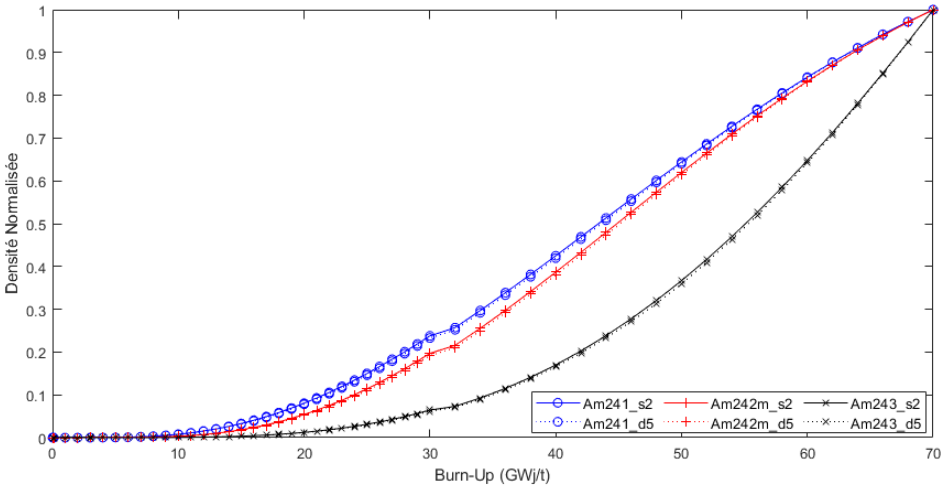

FIGURE 4.6 – Évolution de la densité isotopique de l'Américium en fonction du Burn-Up

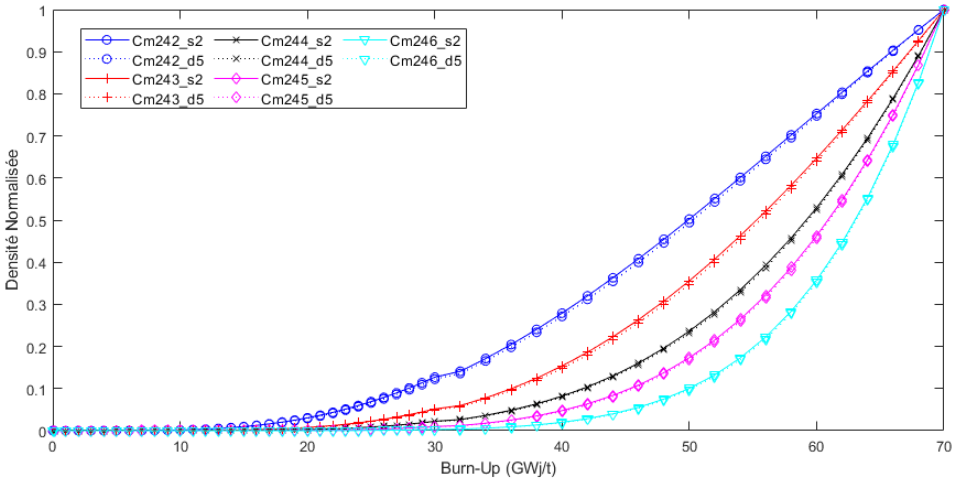

FIGURE 4.7 – Évolution de la densité isotopique du Curium en fonction du Burn-Up

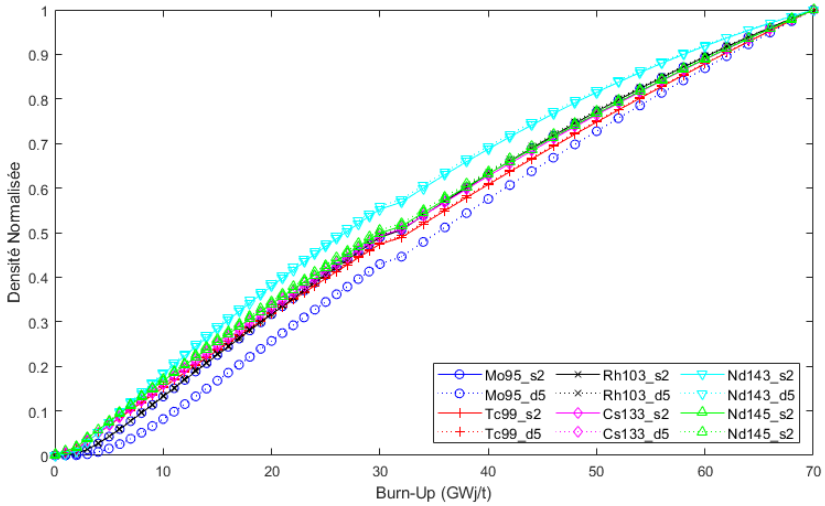

FIGURE 4.8 – Évolution de la densité isotopique de certains produits de fission en fonction du Burn-Up (1/2)

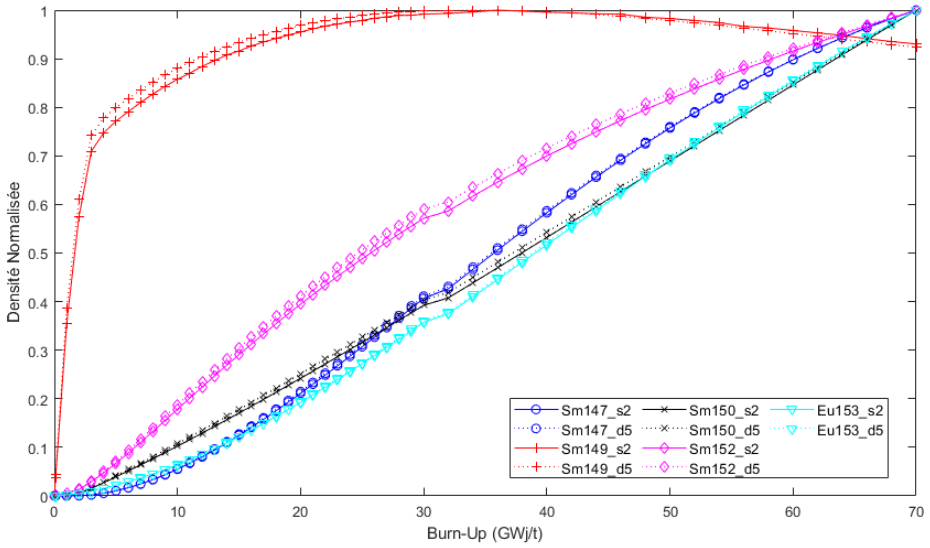

FIGURE 4.9 – Évolution de la densité isotopique de certains produits de fission en fonction du Burn-Up (2/2)

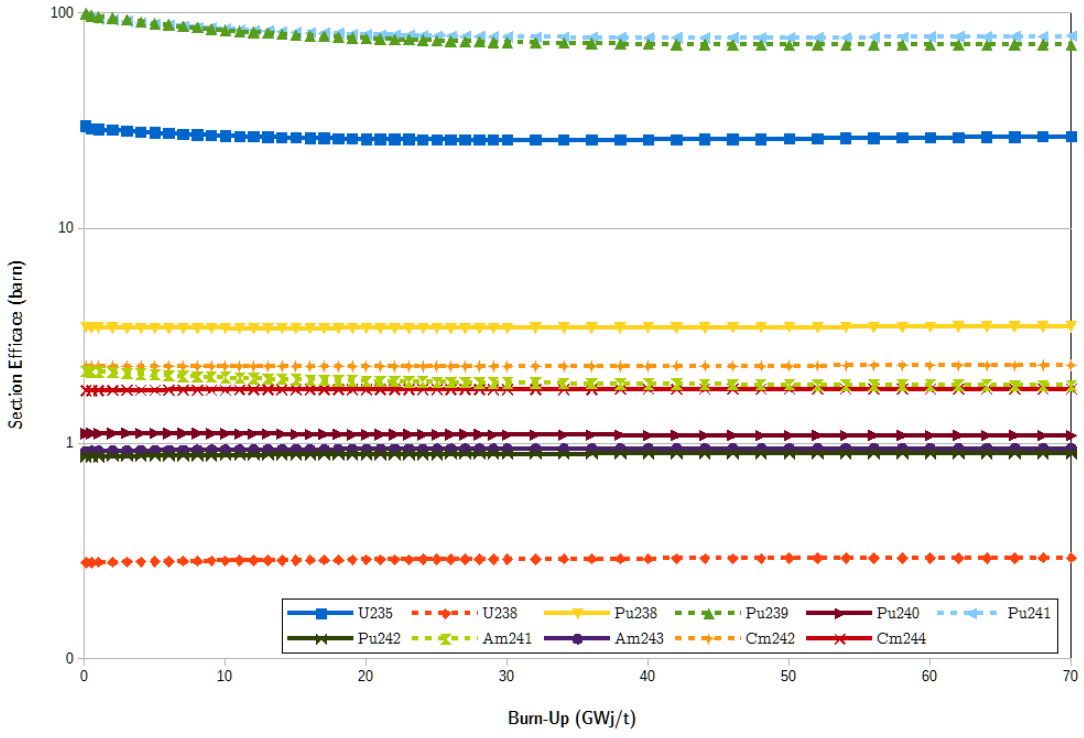

FIGURE 4.10 – Évolution des sections efficaces de production de neutrons en fonction du Burn-Up (Dragon5)

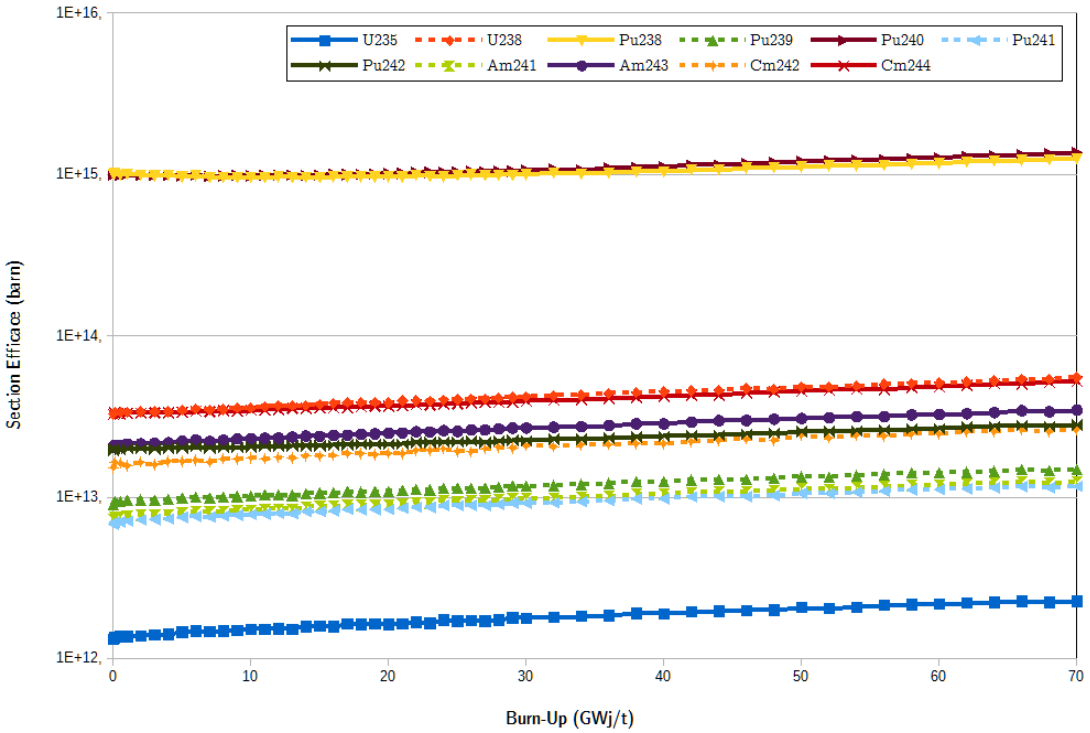

FIGURE 4.11 – Évolution des sections efficaces de production de neutrons en fonction du Burn-Up (Serpent2)

Conclusion

L'objectif de ce projet était donc l'étude de l'évolution d'une cellule de combustible basé sur le benchmark Yamamoto, afin de vérifier la validité du code Dragon 5 sur de long burn-up. Pour ce faire, différents résultats devaient être produits et comparés avec le code Monte Carlo Serpent 2, utilisé comme référence. Plusieurs outils et méthodes ont ainsi été mises en place. Cependant, la première méthode avec approche directe a été rejetée car elle n'obtient pas de résultats cohérents permettant la suite de l'étude. La seconde quant à elle, avec approche progressive, produit des résultats d'une bonne fiabilité, très proches de ceux obtenus par le code référence. Cependant, le calcul des sections efficaces sont à reprendre afin do'btenir des données cohérentes.

Par conséquent, cette étude permet de valider, en partie, la fiabilité du code déterministe Dragon 5 lors de long burn-up. Les résultats produits qui ont été validés peuvent donc être utilisés par le groupe chinois CGN comme références pour leur code déterministe. Cependant, cette étude ne représente qu'une partie du benchmark Yamamoto. Elle peut donc être complétée en reprenant la partie servant à calculer les sections efficaces. Mais elle peut également être prolongée en traitant les cas utilisant du combustible MOx, en cellule ou en assemblage, ou encore en implémentant d'autres méthodes d'autoprotection. Ceci permettrait alors de produire d'avantage de références, pour s'assurer de la fiabilité des codes déterministes chinois.

Bibliographie

[1] Akio YAMAMOTO et Tadashi IKEHARA et Takuya ITO et Etsuro SA JI. Benchmark problem suite for reactor physics study of lwr next generation fuels. *Journal of Nuclear Science and Technology, Vol. 39, No. 8*, pages 900–912, Feb 2012.

[2] Alain HEBERT. *Libraries and resonance self-shielding calculations.* Ecole Polytechnique de Montr, 2016.

[3] Jaakko LEPPEN. *Serpent a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code , User's Manual.* VTT Technical Research Centre of Finland, 2012.

[4] G. MARLEAU et A. HEBERT et R. ROY. *A USER GUIDE FOR DRAGON VERSION5.* Ecole Polytechnique de Montr, 2019.

[5] G. MARLEAU et A. HEBERT et R. ROY. *A Description of the DRAGON and TRIVAC Version5 Data Structures.* Ecole Polytechnique de Montr, 2019.