Contribution to the modelization of VVER reactor cores in VERSION5

Design of VVER minicore geometries

Nicolas Weisse

September 2024

Under the supervision of:
Professor Alain Hébert - Polytechnique Montréal
Professor Shihe Xin - INSA Lyon

Contents

List of Figures

List of Tables

Acknowledgments

I would like to extend my sincerest thanks and gratitude to Prof. Alain Hébert for giving me the opportunity to work on this project and for his continued guidance at work. It has been a very exciting and interesting internship for me, which I has introduced me to the world of nuclear energy and the world of academic research. It has been very stimulating and has certainly encouraged me to look in both of these directions for my career and this is in no small part thanks to Prof. Alain Hébert.

The warmest and kindest of thoughts must also go to Pierre Fontaine who has been a colleague, a mentor and has turned into a friend as the project went on.

Thanks and appreciation to our very own SERPENT2 snake charmer Philippe Panisi who worked on the reference Monte-Carlo solution and made the office that more lively.

Special thanks also go to Raphaël Guash and Ivan Trancart our two Ph.D. debugging consultants, neutronics hotlines and caring friends who provided much needed help, external advice and great laughs whenever possible.

Abstract

Proposed subject: Contribution to the modelization of a VVER minicore in DRAGON5

VVER reactors are a family of Soviet and Russian power reactors using water as both a moderator and coolant. Although their initial design goes back to the late 1950's they have continually evolved to meet new safety standards, performance requirements and regulatory needs. With a diverse fleet of reactors worldwide, modeling VVER core performance and safety is of both academic and industrial interest for operator countries.
The DRAGON5 lattice code already has the necessary capabilities to build and simulate hexagonal VVER cells as demonstrated in previous studies. The next step is to test its minicore simulation capabilities. This proof of concept will lay the groundwork for further minicore studies which will prepare full core simulations in DONJON5. While modeling simple assembly type geometries is feasible in DRAGON5, its native geometry engine is not suited for mini-core modeling. The goal is to develop a non-native geometry program to build minicore geometries from given ALAMOS and SALOMON assembly geometric elements. Moreover, source code in CLE-2000 scripting language is to be generated by this program to correctly connect the non-native geometries with DRAGON5.

Sujet proposé : Contribution à la modélisation de mini-coeur VVER dans DRAGON5

Les réacteurs VVER sont une famille de réacteurs electrogènes soviétiques et russes à modérateur et caloporteur eau. Si leur conception initiale remonte à la fin des années 50 ils ont continuellement évolué pour répondre aux nouveaux standards de sécurité, aux besoins de performance et s'adapter à différents cadres réglementaires. Avec une flotte de réacteurs VVER aussi variée que ses opérateurs, la modélisation des performances et de la sureté des coeurs représente un intérêt académique et industriel majeur pour les pays opérateurs dans le monde.
Le code de cellule DRAGON5 possède déjà des capacités de construction et de simulations de cellules hexagonales VVER comme démontré dans des études précédentes. L'objectif actuel est de prouver ses capacités de simulation de mini-coeur. Cette démonstration permettra de poser les bases pour de plus amples études mini-coeur qui serviront de base à de futures études coeur entier dans DONJON5. Malgré la possibilité de modéliser des géométries d'assemblages simples dans DRAGON5, son moteur de géométries natives n'est pas conçu pour la description de mini-coeurs. Il est donc question de développer un logiciel permettant de construire des géométries de mini-coeur à partir d'éléments de géométries d'assemblage aux formats ALAMOS et SALOMON. De plus, du code source en langage script CLE-2000 devra être généré pour correctement interfacer ces géométries non-natives dans l'environnement DRAGON5.

Chapter 1: Introduction

1.1 Institut de Génie Nucléaire (IGN)

Polytechnique Montréal's nuclear engineering institute (IGN) is currently responsible for the training of nuclear energy students, the operation of the SLOWPOKE reactor and the development of the VERSION5 deterministic reactor physics distribution. VERSION5's foundations go back to the 1980's with the development of the DRAGON lattice code and the DONJON full core simulation tool. DONJON and DRAGON codes have become reference tools for CANDU reactor operators and are still actively developed with the first DRAGON5 version being released in 2014. It boasts both legacy and state of the art methods for reactor simulation. DRAGON5 is distributed under the LGPL license, making it the only complete open-source reactor simulation distribution in the world, a particularly interesting fact for both private actors and academic users worldwide.

1.2 VVER reactors

VVER nuclear reactors are a family of water cooled and water moderated pressurized power reactors developed in the USSR and later Russia. They have seen 1500 reactor-years of operation worldwide being known for their reliability. Operators include states from the former USSR and satellite states as well as commercial export clients. In recent decades VVER reactors have proven to be a great export success being well understood with incremental design updates and boasting moderate costs along with comprehensive financing and operating schemes.

VVER technology distinguishes itself from other pressurized water reactor designs with its hexagonal core geometry. This design element modifies its behavior compared to other designs and also requires specific hexagonal fuel assemblies. Such a peculiarity creates a dependence on Russian expertise for core design and safety assessment as well as fuel assembly supply. With some legacy VVER operators being in Moscow's crosshair, the need for alternative fuel assembly supply and core expertise has arisen. A first step in understanding VVER technology and being able to produce alternative hexagonal fuel assemblies is to produce benchmarks and simulation models of VVER cores. Said models can then be compared to experimental data measured in operational reactors to further refine their accuracy.

1.3 VVER simulations and the Poly VVER initiative

With a lack of operator data outside the Warsaw pact during the cold war, simulating VVER cores was of little interest and relevance to non user states. As the USSR dissolved and operator data became available, a vested interested was put into understanding and assessing their capabilities and safety. In these new conditions, some simulation environments implemented VVER specific codes while others defined benchmarks for these cores. As the geopolitical situation degraded between Russia and some of the former Soviet republics and Warsaw Pact states, research institutes and companies deepened their codes to be production ready and looked into designing alternative fuel for VVER reactors.

One of these VVER simulation initiatives was the European Commission's Codes And Methods Improvements for VVER comprehensive safety assessment (CAMIVVER). The project took place from 2020 to 2023 with European research institutes and companies striving to provide industrial standard simulations to eastern European VVER operators. In the wake of the CAMIVVER initiative, a parallel project was set up at the IGN. This project's goal is to demonstrate VERSION5's VVER core simulation capabilities and to provide free and open source alternatives to export controlled and heavily licensed calculation schemes developed in other proprietary environments.

In 2023, the previous team of interns laid the foundations of the Poly VVER project by proposing and running calculation schemes for VVER assemblies in DRAGON5. They validated their results with an external neutron transport code, VTT's SERPENT2, proving the relevancy of their approaches and demonstrating a first layer of DRAGON5's VVER capabilities.

The next step in the project is to go into the multi-assembly scale by running simulations on a mini-core. This step is a proof of concept for further mini-core simulations and fine tuning in DRAGON5 before going to full core simulations in DONJON5. This task requires the development of new companion tools to assist in geometry production and to automate coupling with DRAGON5. DRAGON5 being developed for assembly level simulations, its native geometric capabilities prove ill suited for mini-core sized objects. An additional challenge is found in adapting really precise assembly level calculation schemes to a much larger study domain. The SALAMANDER geometry assembler and processing tool was developed as an answer to these project specific needs.

1.4 The Version5 distribution

The Version5 distribution is an Open Source project at Polytechnique Montréal related to the development of codes DRAGON5 and DONJON5. DRAGON5 is a *lattice code* based on solutions of the Boltzmann Transport Equation (BTE) for a small component within a nuclear reactor. This component can be a single fuel pin cell, a fuel assembly, a colorset, a mini-core, or a reflector model. DONJON5 is a *full-core simulation code* based on solutions of the neutron diffusion equation and on the concept of *fuel map* for representing every assembly parameters over a 3D map. DRAGON5 can use *native geometries* based on its own geometry engine, or *non-native* geometries based on an external tool such as ALAMOS. Our contributions are related to the support of non-native mini-core geometries within DRAGON5. DONJON5 framework was not used.

1.5 Contents and layout of this report

This report provides information about the methods used to represent and process geometries in SALAMANDER and introduces some of the minicore VVER results.

Firstly, key concepts in reactor physics and transport codes are introduced so that the reader may understand the challenges and interest of the simulations. Past VVER simulation endeavors and the studied VVER elements are reviewed to then turn the attention on geometrical representations in transport codes, being this project's center of attention.

The second part exposes the minicore geometry, DRAGON5 representation and use of geometric data and then introduces the calculation scheme developed for the minicore study. This part serves to outline the requirements for SALAMANDER as a both a geometry assembler and as a companion tool. It also serves to introduce some elements of minicore calculation for later result analysis.

With an understanding of the challenges and capabilities at hand, the SALAMANDER tool is introduced. This part describes its general architecture, its methods for processing input data and the procedure to produce the minicore geometry.

Results are then analyzed, their relevance being compared with another reference tool and further improvements are discussed from the geometric perspective.

Chapter 2: Geometries in transport codes : a literature review

This literature review briefly introduces the reader to some key reactor physics concepts and the types of simulation codes. The first two sections are a synthesis of information from [19], [31] and [5] which are reference books in the field of neutron transport and reactor physics. Afterwards, past and current advances in VVER simulations are reviewed and VVER core elements are introduced. With their geometry exposed, the attention turns geometric representations in simulation codes to conclude on the interest of this work.

2.1 Fission & nuclear reactions

2.1.1 Neutron / nucleus interactions

The main focus of reactor physics is the study of neutrons and their interactions with the elements in the reactor core. All operational nuclear reactors are built around a controlled and sustained fission reaction which provides thermal energy and free neutrons to sustain the reaction. Neutron and matter interactions are characterized by the incident neutron's kinetic energy and the target nucleus' isotope. Because of the quantum nature of nuclear interactions, only probabilistic and statistical quantities can be used to describe these interactions. For instance, probabilistic properties called cross sections quantify the odds of specific interactions happening between a neutron of energy E and a given isotope.

The microscopic cross section for interaction type x, σ_x, characterizes interaction probabilities between a neutron and an individual atom. Its dimensions are that of a surface $[L^2]$ and may be understood as an equivalent target size. It is conventionally given in barns which amounts to $10^{-28}m^2$. The macroscopic cross-section $\Sigma_x[L^{-1}]$ is obtained by multiplying σ_x by N_i where N_i is the density of isotopes i $[atoms.L^{-3}]$. When multiplied by a length l it yields the probability of an x interaction after a neutron traveled l deep into the i isotope medium.

While there are many types of neutron-matter interactions the most prevalent in reactors are :

- Fission (n, f): the incident neutron splits a heavy nucleus into several lighter nuclei, releasing other free neutrons and energy
- Radiative capture (n, γ): the incident is captured by the target nucleus which releases γ rays to return to a stable state
- Elastic scatterings: the incident neutron and the target nucleus conserve their total energy

in what is akin to an elastic shock, it may involve the formation of a temporary compound nucleus in resonant elastic scatterings but no energy is lost through γ ray emissions

- Inelastic resonant scattering: the incident neutron and the target nucleus temporarily form a compound nucleus which releases a free neutron and γ rays to return to a stable state

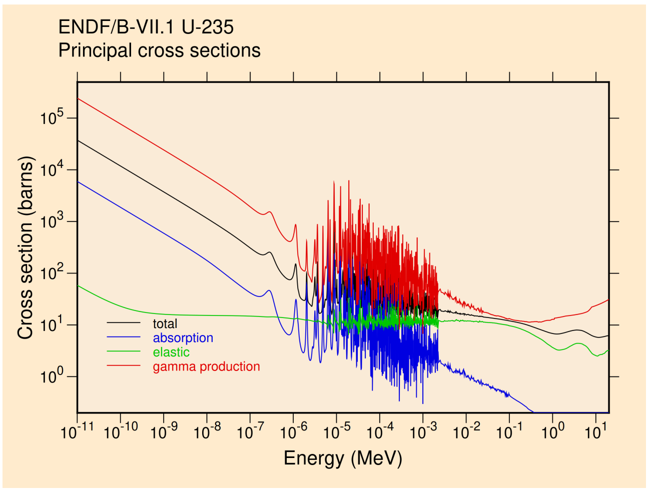


Figure 2.1: Some of the major cross section data for the ^{235}U [9]

As seen in 2.1 cross sections exhibit some erratic behavior, in that particular case in the $[10^{-6}; 10^{-3}]$ MeV range. These spikes are called resonances and are the result of quantum phenomenons. They are responsible for a more macroscopic phenomenon called self shielding 2.1.2. Cross sections also vary with temperature due to the Doppler effect. Resonances tend to diminish in amplitude but to widen in energy band with temperature. It is important to take this into account when designing reactors to operate at certain temperate ranges.

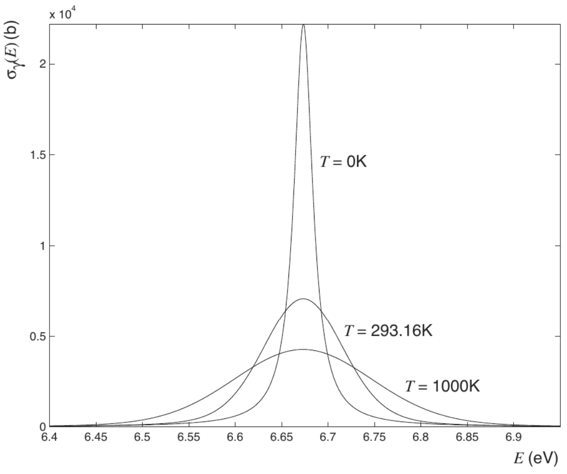


Figure 2.2: Doppler broadening for a ^{238}U resonance [19]

2.1.2 Key nuclear variables and quantities

In total, to accurately describe a neutronics problem the following variables are needed:

- 3 space variables condensed into $\vec{r}$, center of an elementary volume d^3r
- 1 speed V_N or energy E variable, V_N the velocity vector's norm, linked to energy by $E = \frac{1}{2}mV_N^2$
- 2 direction variables, usually condensed into a solid angle $\vec{\Omega}$
- 1 time variable t

Additionally, the neutron population density at position $\vec{r}$ of speed V_n going in direction $\vec{\Omega}$ is given by the $n(\vec{v}, V_n, \vec{\Omega}, t)$ variable whose dimensions are $[neutrons.L^{-3}]$. As seen before, the density of atoms in a given medium is noted $N[atoms.L^{-3}]$.

Flux and current

The neutron flux, ϕ is a shorthand notation appearing often in reactor physics used to quantify neutron population distributions. It is expressed in quantities $[L^{-2}.T^{-1}]$ and does not represent an actual flux as defined in other fields. It can be understood as the number of neutrons of speed V_n going through a surface of normal $\vec{\Omega}$ by unit of time.

$$\phi(\vec{v}, V_n, \vec{\Omega}, t) = V_n n(\vec{v}, V_n, \vec{\Omega}, t) \tag{2.1}$$

The more classical flux definition is the neutron current. For a surface element d^2S defined by the unit normal vector $\vec{N}$ the angular current $\vec{J}$ can be defined as :

$$\frac{d^3n}{d^2Sdt} = \vec{J}(\vec{v}, V_n, \vec{\Omega}, t).\vec{N} \tag{2.2}$$

Neutron multiplication factor and reactivity

The neutron multiplication factor k is an indicator of the evolution of the neutron population. It is a dimensionless number that can be understood as the ratio of neutrons between two generations or the average number of fission neutrons that will themselves cause fission reactions. Its values can describe three evolutions :

- $k < 1$: subcriticality : the neutron population is decreasing exponentially, not enough neutrons are produced to sustain the chain reaction
- $k = 1$: criticality : the neutron population stays constant, the reaction is balanced and sustained,
- $k > 1$: supercriticality : the neutron population is increasing exponentially

A distinction is to be made between the k_∞, neutron multiplication factor in an infinitely repeated environment with no leaks and the k_{eff}, effective neutron multiplication factor simulated or observed in a given bounded medium.

Reactivity $\rho = \frac{k-1}{k}$ is another way to describe a neutron population evolution, using criticality as a reference. It is relevant when weighing the contributions of different elements to core dynamics. Absorbent materials will add negative reactivity to the core while neutron sources will add positive reactivity to the core. It is measured in pcm (per cent mille, $10^{-}5$) to account for fine evolutions.

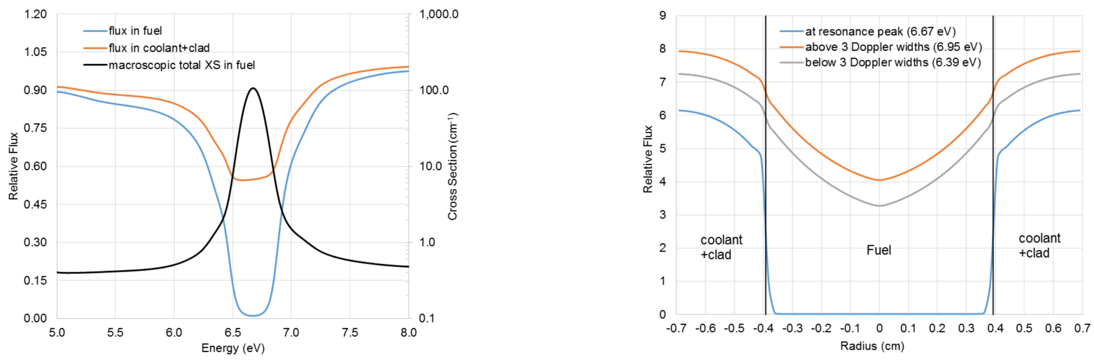
Reaction rates

The reaction rate τ_x, is the number of x type reactions, happening at E energies, per unit of volume and time. It is later mapped throughout the minicore to assess its behavior and compare its reaction and power distributions to other reference cases. It can be written as :

$$\tau_x(\vec{r}, E, t) = \Sigma_x(\vec{r}, E, t)\phi(\vec{r}, E, t) \tag{2.3}$$

Self-shielding

As seen in 2.1, some isotopes contain resonances. The increased cross sections at given neutron energies deplete the neutron flux around these energies in resonant isotopes. This phenomenon happens both on an energy and spatial level, these are respectively called energy and spatial self shielding. Isotopes are said to be self shielded from neutrons in these energy bands as their flux is depleted.



(a) An example of energy self shielding

(b) An example of spatial self shielding

Figure 2.3: Self shielding phenomenons for a ^{238}U resonance [23]

Modeling self-shielding proves challenging both analytically and computationally. In transport codes, it generally involves intermediate steps and adjusting cross sections in specific regions to account for this phenomenon. This step will be mentioned in the DRAGON5 architecture and calculation scheme part.

Burnup

Burnup is a quantity used to measure how much of the fuel has already been used. Some of the key reactor parameters mentioned before tend to be plotted against burnup rather than time. In power reactors it is the product of the reactor's nominal thermal power by its running time in days divided by its initial fuel load mass : $\frac{\dot{Q}\Delta,t}{m_i}$. Its units usually are GWd/t or MWd/t, gigawatt, or megawatt day per tonne.

2.1.3 The transport equation

The Boltzmann Transport Equation (BTE) known as both the Boltzmann Equation and the Transport Equation is a balance equation that describes the neutron flux's evolution. In a static case, in an elementary volume d^3r, it is written as follows :

$$\vec{\Omega}.\nabla\phi(\vec{r}, E, \vec{\Omega}) + \Sigma(\vec{r}, E)\phi(\vec{r}, E, \vec{\Omega}, t) = Q(\vec{r}, E, \vec{\Omega}) \tag{2.4}$$

- $\vec{\Omega}.\nabla\phi(\vec{r}, E, \vec{\Omega})$: accounts for the flux's spatial evolution, in that convention positive if neutrons leave d^3r and negative if they enter d^3r
- $\Sigma(\vec{r}, E)\phi(\vec{r}, E, \vec{\Omega})$: accounts for neutrons interacting with matter, subsequently changing energy and / or direction
- $Q(\vec{r}, E, \vec{\Omega})$: the source term in the equation, it counts two contributions :

  - neutrons being scattered, changing directions and energy
  - neutrons produced from fissions within d^3r

With reactor cores spanning a dozen meters and containing heterogeneous materials, solving the transport equation analytically proves impossible in most cases. This is why numerical methods to model neutron transport have been developed.

2.2 Numerical approaches to neutron transport

Two main approaches to neutron transport can be identified: Stochastic and Deterministic. The latter solves the BTE numerically while the former simulates individual neutron trajectories. The following part introduces some of the major elements and challenges behind transport codes. It must be noted that they are complex and highly configurable environments that provide base building blocks to users that then have to chain them and configure them properly to fit the needs of their study in what is called a calculation scheme. As such, the backbone of a transport code can only be correctly used with a sound calculation scheme.

2.2.1 The stochastic way

Stochastic codes, also referred to as Monte-Carlo codes, take a statistical approach to core simulation. These codes simulate individual neutron trajectories and interactions with random number generation. They have the disadvantage of having statistical errors, which may be reduced with higher neutron populations and more iterations. Simulating each neutron's trajectory and interactions proves more costly than a deterministic resolution as this involves simulating thousands to millions of neutrons per iteration. However, Monte-Carlo methods prove extremely relevant for reference calculations as their individual neutron simulation are closer to the actual physical phenomenons at play in the reactor than the deterministic solutions. Additionally, they are generally easier to use as their geometry engines and solvers are more flexible than deterministic codes. Some of them also have the advantage of being energy continuous, meaning they do not discretize the energy variable into groups as seen in deterministic codes, relying instead on local interpolations. In this study the SERPENT2 Monte-Carlo code will be used to validate DRAGON5's results.

2.2.2 The deterministic methods

Deterministic methods encompass a wide array of ways to solve the neutron transport equation with differing mathematical and numerical approaches. When properly configured deterministic codes prove orders of magnitude faster than stochastic codes and provide reasonably precise results relative to them [14]. This makes deterministic codes relevant for industrial users running great amounts of calculations or for academic users with limited computational resources.

A distinction is to be made between what are called lattice and full core codes. The former, like DRAGON, are for modeling neutron transport in small objects like individual cells or sets of cells forming an infinitely repeated pattern, a lattice. The latter, like DONJON, are meant to model full cores once individual assemblies have been studied with a lattice code. Full core codes make use of more approximations than lattice codes, for instance modeling neutron transport with a diffusion equation or discretizing neutron energies in way less groups than in lattice codes, usually two against at least a hundred. However, these approximations prove sound in practice on core sized objects.

This study is strictly about using DRAGON5 for multi-assembly simulations, being at the limits of what a lattice code is able to do. Calculations on the minicore geometry produced in this work made use of the Method of Characteristics (MOC) and the Collision Probability method (CP or Pij) with its Interface Current (IC) sub case. They both involve discretizing the space and energy variables.

Track generation

Track generation is the process which translates raw geometric data defined by the user into solver specific spatial data for MOC and CP calculations. The main challenge behind track generation is to accurately represent a given geometry while keeping the number of tracks as low as possible. The direction variable is broken down into a set number of discrete values and parallel tracks following these directions are then traced throughout the domain.

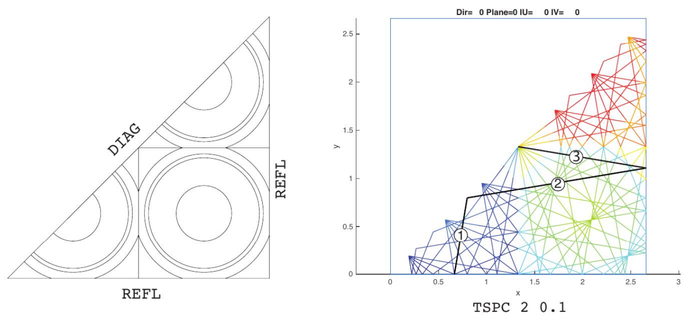


Figure 2.4: An example of cyclic track lines (or TSPC tracks) with various boundary conditions applied on a given geometry (left) [20]

An additional challenge of track generation is the handling of boundary conditions. When dealing with infinitely repeated lattice patterns, shape specific boundary conditions have to be implemented to account for tracks crossing over from neighboring cells [17], as exemplified in 2.4. This is known as cyclical tracking, as tracks exiting the lattice cycle back to it, respective of lattice shape and boundary type. For wholly defined geometries of finite extents, uniform tracking algorithms may be used. It limits the user to either void or isotropic reflection boundary conditions but allows the use of irregular geometries [35] [20] [28].

Energy discretization

As seen in 2.1.3, the transport equation is energy dependent. In deterministic codes solving the BTE also involves discretizing the continuous energy variable into smaller intervals called groups. Splits are not made at regular intervals but are weighed against cross sections to better represent the spectrum, the accurate treatment of resonances becomes a challenge here. Lattice codes like DRAGON5 typically have a great quantity of groups as the studied domains are small[19]. This work uses the legacy SHEM295 energy mesh with 295 energy groups.

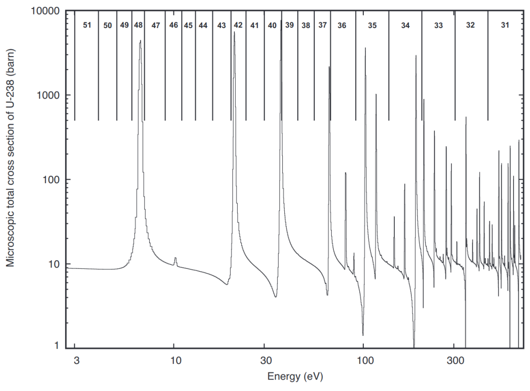


Figure 2.5: Resonant energy group splits for the ^{238}U [21]

Method of Characteristics

In MOC solvers the BTE is integrated along straight paths called characteristics. Taking $\vec{M}$ as a starting point, the space variable is simplified as $\vec{M} + s\vec{\Omega}$ where s is the length traveled along the track. This gives the characteristic form of the BTE :

$$\frac{d}{ds}\phi(\vec{M} + s\vec{\Omega}, E, \vec{\Omega}) + \Sigma(\vec{M} + s\vec{\Omega}, E)\phi(\vec{M} + s\vec{\Omega}, E, \vec{\Omega}) = Q(\vec{M} + s\vec{\Omega}, E, \vec{\Omega}) \tag{2.5}$$

This method only produces an N size result matrix for each energy group where N is the number of regions in the geometry. As such it is generally preferred when running calculations with domains containing more than a few hundred regions.

Collision probability method

Like in the MOC approach, the CP method is based on discretizing space along certain neutron paths. The domain is split into regions and flux is computed for each individual region i from the probability of a neutron coming from region j colliding into i, hence the Pij name. For a given energy group g, the reduced CP can be written as :

$$p_{i,j,g} = \frac{P_{ij,g}}{\Sigma_{j,g}} = \frac{1}{4\pi V_i}\int_{V_i^\infty} d^3r' \int_{V_j} d^3r \frac{e^{-\tau_g(s)}}{s^2} \tag{2.6}$$

The flux in region i is then computed as :

$$\phi_{i,g} = \Sigma_j Q_{j,g} p_{ij,g} \tag{2.7}$$

With $Q_{j,g}$ being the neutron production term in region j for energy group g.

This method proves particularly costly for large domains as it produces an $N \times N$ result matrix per energy group, where N is the number of regions in the domain. On top of that, the CP method assumes isotropic neutron emissions which the MOC does not. It remains relevant in few region problems with symmetries involved.

An interesting sub case of the CP method is the interface current (IC) method where cells are bundled by families. The collision probabilities are only calculated for cells of different families and coupling between families is represented using interface currents. It speeds up computation and is interesting when building self-shielding libraries.

2.3 VVER core simulation endeavors

VVER simulation is a relatively recent and active topic in transport codes as validating codes for a given reactor type requires relevant experimental data. This was only made possible after the complete fall of the Iron Curtain in the early 90's and the compilation of data by external organisms like the OECD or the EU [6]. Moreover, codes mostly optimized for conventional square PWR geometries had to be adapted to hexagonal VVER geometries. The following literature review briefly exposes some early and contemporary benchmarking efforts, DRAGON's VVER capabilities and the CAMIVVER project to then open up on the POLYVVER project.

2.3.1 Benchmarking & validation

The development of accurate computer models requires comparison to experimental datasets from defined reference cases. Comparing the relative accuracy of a computer model against real world data or other reference simulation models is called benchmarking. Running a transport code through benchmarks is a required step when adapting it to and certifying it for a new type of reactor. As such, one must separate the process of certifying a code from its actual use for a specific study case. This makes the study of VVER challenging on two levels, the actual adaptation and validation of codes to VVER cores and then the design and development of industrial calculation schemes in newly certified codes.

As the Cold War drew to a close access to VVER reactors was made easier and a variety of benchmarks have been defined and continue to be defined. As an example, the 2002 VVER Low Enriched Uranium (LEU) and Mixed OXyde (MOX) benchmark from the Nuclear Energy Agency (NEA) has been widely used as a reference case in the validation of transport codes. It originally compared computational results from Russian codes and non Russian Monte-Carlo and deterministic codes. This benchmark's legacy and relevance is justified by the challenges posed by MOX (plutonium + uranium) assemblies, as they both modify power distribution and isotopic concentrations in the core [1].

Another example would be a more recent benchmark defined from the Khmelnytskyi-2 reactor whose fuel assemblies are later studied in this report. While it was defined in 2009 it has seen a few revisions until 2020 with improvement of computer models and technical documentation being refined [25] [3]. It provides great insight into early operation of a VVER with data being gathered from the first cycles of the reactor. It has been used in conjunction with SERPENT to fill in some missing experimental data. This shows the iterative cycles benchmarks, methodologies and codes can go through before having a definitive convincing package.

These two examples illustrate the newfound wealth in VVER data and benchmarks and the ongoing efforts to improve codes that are already certified or that are being certified for VVERs. As some codes are validated for VVER studies they can then serve as references for the validation of others. Such is the case of SERPENT2, passing several real and virtual VVER benchmarks [32]. It is later used in this study as the reference model. These examples also show that while VVER technical documentation and experimental data is becoming more and more accessible, current codes continue to be improved and refined for VVERs and that industrial calculation schemes are still in the works for VVERs [16].

2.3.2 Early DRAGON VVER capabilities

During the 90's DRAGON's geometry module and the **EXCELT:** track generator were expanded with hexagonal design and simulation capabilities [29]. This first step still proved rather limited, with no major VVER studies being undertaken with it. DRAGON reached its full potential with H. P. Raghav's 2012 Ph.D. work [30] and the addition of new tracking algorithms for hexagonal assemblies in **NXT:** [17]. Since then, DRAGON has been used for various academic VVER studies comparing it to other deterministic and Monte-Carlo codes.

One of the most major and recent VVER study in DRAGON is a 2023 paper from the China Institute of Atomic Energy [36]. They ran DRAGON through 3 VVER benchmarks, one of them being the NEA's LEU and MOX benchmark, following the evolutions of k_∞ and isotopic concentrations in pin-cells and assemblies. DRAGON's relevance and reliability was illustrated by its close proximity to stochastic simulation results and experimental data, with DRAGON performing better than reference codes against experimental data. The authors deemed DRAGON suitable for pin-cell and assembly level VVER simulations.

2.3.3 The CAMIVVER initiative

The CAMIVVER (Codes And Methods Improvements for comprehensive VVER safety assessment) project was born out of a renewed effort to grant more autonomy to VVER operators in Europe with new benchmarks being defined and data being gathered. Started in 2020, its main focus is, as its name implies, safety assessment of VVER plants and improvements of current codes for these studies. As such it outlines guidelines and improvement for VVER calculation schemes [33] as well as new safety benchmarks with coupled thermal hydraulics codes [15]. This shows the very ongoing nature of VVER modeling from neutronic, thermal hydraulics and safety perspectives. CAMIVVER highlights the need to readapt and requalify existing tool chains for this type of reactors and the current challenges and perspectives for VVER modeling.

A 2024 paper published by Karlsruhe Institute of Technology (KIT) as part of the CAMIVVER project serves as the basis for this project's geometry definition [26]. The paper itself outlines methods to compare theoretical examples of transient control rod ejection in minicores. It is concerned with the reactivity weight of control rods and coupled thermal hydraulics study of the impact of rod ejection. These studies are conducted to compare deterministic schemes developed during CAMIVVER to reference stochastic solutions. This paper could prove useful for further minicore studies and could be used to benchmark a DRAGON VVER minicore against reference test cases and data.

2.3.4 POLYVVER: current state and future perspectives

In the wake of the CAMIVVER project, the 2023 POLYVVER project focused on lattice and assembly level comparisons between DRAGON5 and SERPENT2. Their aim was to adapt several classical PWR calculation schemes to VVER reactors in the DRAGON5 environment. Their results proved the already demonstrated quality of DRAGON's VVER simulation and the feasibility of a REL2005 two layer calculation scheme for VVER assemblies [11]. Adapting a REL2005 scheme to VVER cores in DRAGON5 means that it can now be used to reliably output production level results for industrials using standard schemes and not just project specific tweaks and adjustments. Their DRAGON calculation scheme also serves as a reference architecture for the continuation of the POLYVVER project. Additionally, they developed post-processing tools to compare data from SERPENT2 and DRAGON5 [7]. This first step was major, providing the current minicore project valuable glue tools for data manipulation and a base DRAGON code architecture to work from and adapt.

With assembly elements proven to work in DRAGON, the POLYVVER minicore project starts with the goal of adapting and optimizing the calculation schemes to a larger scale. This part of the project is about building a first technology demonstrator for VVER minicore studies in DRAGON. It must noted that minicore studies are relatively rare in lattice codes and that DRAGON was never used with minicores this size. But as seen during the CAMIVVER project, studying minicores can prove interesting for benchmarking and localized safety assessment studies. Minicore calculations also act as a stepping stone before full core calculations to certify the whole VERSION5 toolchain for reliable VVER simulations.

It must be noted that at the time of writing, transport codes validated for VVER core modeling and simulation exist but are entirely proprietary. Continuous efforts are being made to further extend their capabilities whether on the software side or calculation scheme part. Their proprietary nature poses problems for potential users who may fall under budget constraints, export control regulations or who could see their license agreement severely restrict their use of a given code. Proprietary software just maintains users in a state of dependence relative to the parent institution or company. This is why VERSION5 and its open-source model proves an interesting alternative for VVER core simulation to other transport codes, as it already is for other PWR designs.

2.4 VVER core elements

VVER reactor cores are made from a set of hexagonal Fuel Assemblies (FA) enclosed in a pressure vessel where cooling water is circulated in a closed loop. FA are themselves made up of fuel rods, control rods and instrument tubes. This study is set at a multi-assembly scale. The core elements studied in this report are from Khmelnytskyi-2 (KML2), in Ukraine. It is of type V-320 and has been built in the early 1980's. In standard conditions the KML2 reactor operates as follows:

Number of assemblies	163
Reactor power (MWt)	3000
Reactor pressure (MPa)	15.7
Moderator temperature (K)	560
Fuel temperature (K)	900
Gross power output (MWe)	1000
Net power output (MWe)	950

Table 2.1: KML2 nominal operating conditions from [2]

2.4.1 Moderating water

Water is circulated through the core to cool it and moderate the neutrons. Moderation is the process of slowing down high energy neutrons to lower energies with scattering interactions. Here, hydrogen atoms in the water have a high scattering cross section and slow the fission neutrons in the reactor. The neutrons emitted after fissions typically have energies of a few MeV while U_{235} fission is more prevalent for neutrons at energies of a few eV. Water may also be injected with soluble boron, a neutron absorber, to add negative reactivity to the core and modulate its power output on a global scale.

2.4.2 Fuel rods

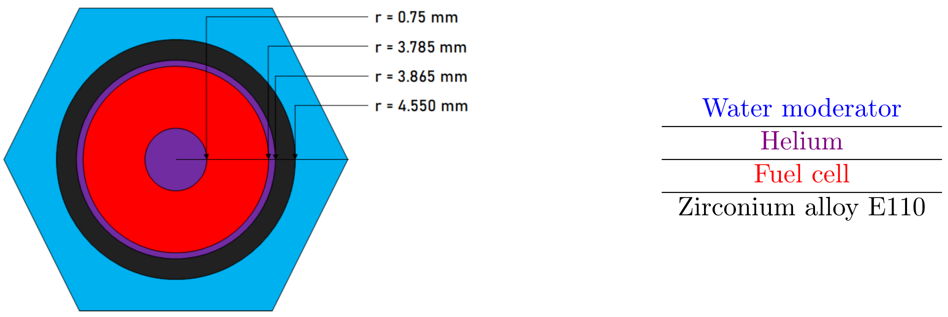


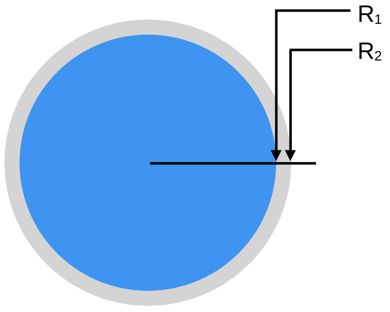
Figure 2.6: A VVER fuel rod taken from [11]

VVER fuel rods are made up of 3 components. An external zirconium alloy cladding which serves as a containment layer in case of accidents, otherwise it is virtually transparent to neutrons, having a very low total cross section. Two gap layers filled with helium which are located both within and without the fuel cell. And the actual fuel cell, made up of UO_2 pellets or UO_2 and Gd_2O_3 pellets in burnable absorbers rods. These rods contain a set mass percentage of fissile U_{235}, this is called enrichment and it governs the core's dynamics. VVER reactors run on 3% to 5% enriched uranium fuel [27].

As mentioned before their fuel assemblies may contain what are called burnable absorbers, i.e. isotopes with high absorption cross sections. As these burnable absorbers capture neutrons they turn into isotopes with lower absorption cross sections. They provide local negative reactivity in early stages of reactor operation and as time goes on and they have transformed into stabler isotopes, their absorbing effect vanishes. When properly placed, burnable absorbers are a way of locally smoothing reactivity over time [10].

2.4.3 Control rods & instrument tubes

VVER control rods are made from a given enrichment of B_4C or Dy_2O_3 TiO_2, two potent neutron absorbers. Depending on the location of the fuel assembly within the core, its rods may be used for power modulation, partially inserted and dynamically readjusted, or just safety purposes, all in / all out configurations. Instrument guides tubes are located at the center of each VVER FA. Both these guide tubes are made up of zirconium alloy E635.



	Control rods	Instrument tubes
R_1 (mm)	5.45	5.50
R_2 (mm)	6.30	6.50

Figure 2.7: VVER control rod and instrument tube dimensions

2.4.4 TVSA fuel assemblies

The Ukrainian VVER fleet uses TVSA fuel assemblies. The KML2 assemblies have the peculiarity of having stiffener blades around their outer edges, it is represented in 2.8, they are meant to improve mechanical stability. Their presence does not modify the actual lattice pitch but may cause some parasitic neutron absorption.

Number of fuel rods	312
Number of guide tubes	18
Number of central guide tube	1
Hexagonal lattice pitch (cm)	23.6
Active height (cm)	353.0
Fuel mass (kg)	491.4 ± 4.5

Table 2.2: KML2 TVSA fuel assembly data from [2]

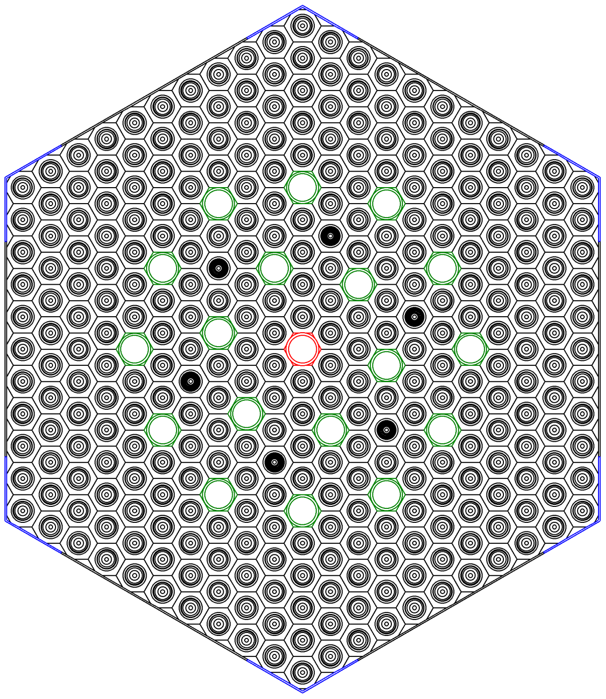


Figure 2.8: A TVSA fuel assembly

2.5 Geometries and transport codes

Before diving into how transport codes represent geometries, some definitions are needed to lift confusions. The words geometry and mesh are sometimes used interchangeably while they designate two different concepts [4].

- A geometry is the abstract structure describing an object with sets of elements and operations. The quality of the modeled object is tied to the underlying geometric representation model and its capabilities.
- A mesh is the discretization of the geometry into smaller parts, typically for rendering or computing purposes. Meshes are a form of geometry themselves, they are a simpler representation of the model with coarser geometric elements.

The line between geometries and meshes is often blurred in reactor simulations as users manually discretize regions to better account for phenomenons such as spatial self shielding. With this side note out of the way, this section is only about geometries.

Another important note is the platform dependent nature of geometric representation models in transport codes. With various approaches to solving the transport equation, the multiple ways of discretizing space and their relative computational costs, geometries are often constrained by the solvers. This makes it so that no definitive, universal models can be described for all transport codes of a given type. Studying individual models also requires having access to technical documentation, which is often restricted for proprietary software. The following subsections gloss over a few documented examples encountered during the project.

2.5.1 Geometries in deterministic codes : the DONJON & DRAGON example

An example of platform specific geometry can be found in DRAGON's gigogne geometries which shares some commonalities with the CEA's APOLLO 1 and 2 geometries. The gigogne representation model is highly structured and is specifically aimed at modeling reactors with lattices built from elementary predefined shapes. Individual geometries describing individual cells can then be layered into one another with recursion to produce larger geometries such as fuel assemblies [20] [13].

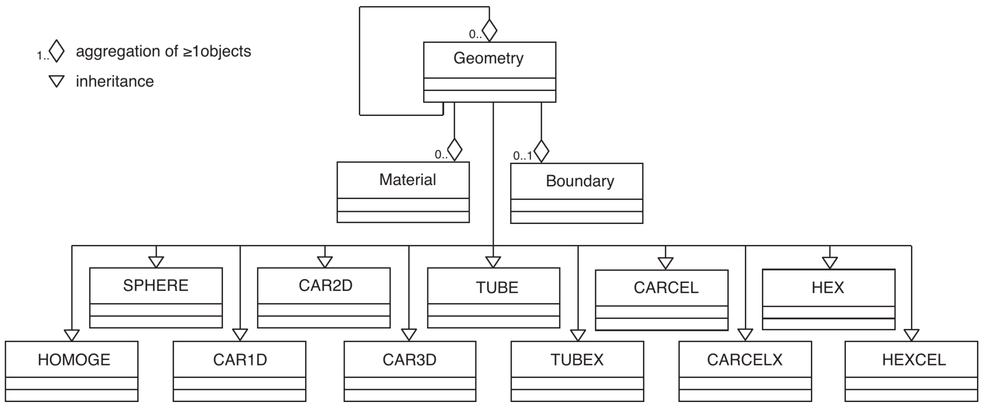


Figure 2.9: The gigogne geometry architecture [20]

Such a structure simplifies tracking to a certain degree as it limits geometries to a few regular shapes with set attributes. These particular shapes can be identified and proper boundary conditions may be applied to them. The layered structure also means that geometries can be represented as trees, which makes boundary and collision detection easier when generating tracks. However, users are bound to simplify their models to fit the native geometric capabilities. With conventional PWR cores being made up of square or hexagonal assemblies these limitations have only been a minor hurdle in core simulation. They may prove limiting for more irregular and complex core designs like the ones found in fast breeder reactors or small modular reactors.

2.5.2 Constructive solid geometries

A more flexible approach to core geometry design is found in Constructive Solid Geometry (CSG). It allows users to define surfaces and volumes from a set of primitive shapes which are then combined with boolean geometric operations (intersection, union and exclusion). Like before, the primitives remain a limiting factor although the possibilities are extended by the boolean operators.

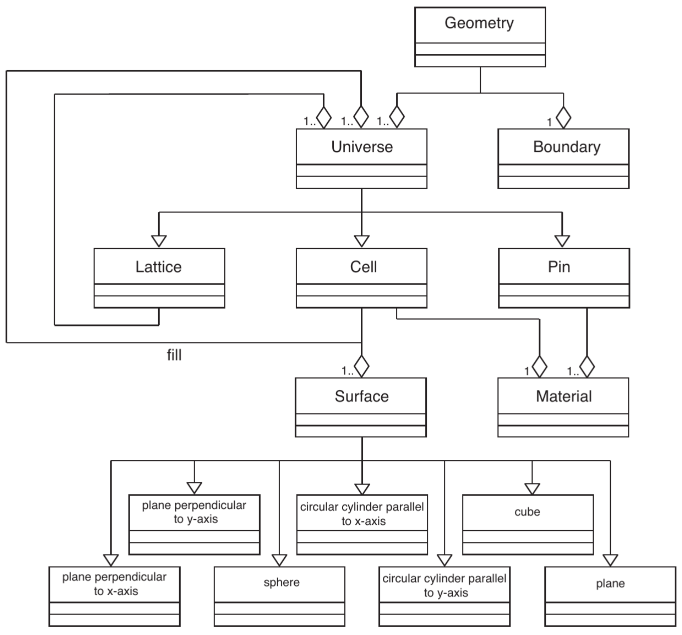

Geometry
1..
1..
1..
1
Universe
Boundary
Lattice
Cell
Pin
fill
1..
1
1..
Surface
Material
plane perpendicular to y-axis
circular cylinder parallel to x-axis
cube
plane perpendicular to x-axis
sphere
circular cylinder parallel to y-axis
plane

Figure 2.10: The SERPENT2 constructive geometry model [20]

As illustrated in 2.10, CSG models, such as the one in SERPENT2 [24], may involve some complex and layered structures, grouping up pins in cells, cells in lattices and lattices in universes. This nested structure proves powerful for describing complex core or assembly patterns. Hierarchical constructs like these allow the use of tree data structures to represent a given geometry. This is efficient when testing collisions with the ray tracing algorithms used to model neutron trajectories in stochastic codes, hence the historical of CSG in Monte-Carlo codes.

This approach is still less structured than gigogne geometries and is fairly rare in deterministic transport codes. As examples, GTRAN2 and OpenMOC use CSG as their base geometric representation model [28]. Information about both of these codes is scarce and it must be noted that no reference deterministic code, like DRAGON, uses CSG.

2.5.3 Boundary representation

An answer to the platform specific and primitive specific limitations mentioned before is boundary representation or BRep. These geometries are made up of elementary shapes which describe the boundaries of surfaces or volumes. An example of a BRep format is the CEA's and EDF's ALAMOS format. With segments, circles and arc circles most of the complex geometries can be constructed or at least faithfully approximated.

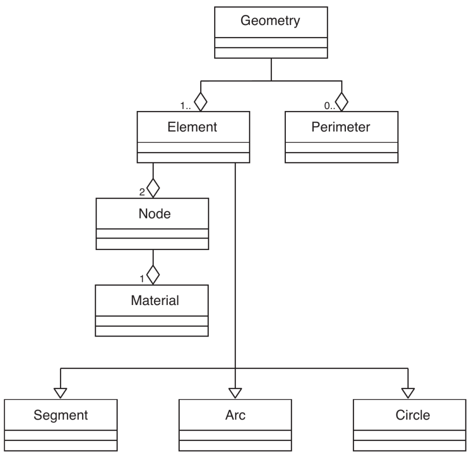


Figure 2.11: The SALOMON geometry architecture, a precursor to ALAMOS [20]

As see in 2.11, the geometry is now a set of completely unstructured elements, defining possibly incoherent shapes. It is important to perform consistency checks and structure healing procedures to ensure the overall soundness of a structure. With the added complexity of manipulating simpler elements than in gigogne models, the design of such geometries requires a dedicated companion tool with a GUI [8] [22]. Track generation also becomes more challenging as regular and irregular shapes may coexist in the same geometry. However with the implementation of more general tracking algorithms some of these problems have been alleviated, allowing DRAGON to import ALAMOS and SALOMON formatted BRep geometries [35].

2.5.4 Conclusion & future perspectives

As core geometries become more complex and irregular with next generation core designs, the need for advanced geometric models grows. Contemporary efforts are focused on providing GUI tools to design models usually by way of interfacing with external CAD platforms. These new interfaces are also meant to be used for result visualization and for data fusion from other multi-physics simulation software. One such example can be found in the SALOME platform, whose development is lead by EDF and the CEA among other institutions and industrials. Its provides a graphical and programmable CAD interface which may be extended with modules like the CEA's ALAMOS [8]. SALOME's modular architecture opens up new possibilities for advanced calculation schemes, result analysis and deeper coupling between DRAGON and codes from other domains in reactor simulation

[22]. Such platforms also make core design for comparative deterministic / stochastic studies easy with BRep to CSG conversion tool kits such as McCAD [18].

Further developments of integrated platforms like SALOME and individual software interfaces provide a compelling answer to the problems posed by core geometry design and analysis. With this in mind, challenges still remain in deterministic codes to develop track generators with appropriate cyclical tracking algorithms for given lattice shapes. It must be noted that core design modules like ALAMOS in SALOME are for the most part proprietary. This still proves a hurdle for export controlled or budget restricted projects. Efforts are being made to circumvent such restrictions at the IGN with open source tools being in the works but they remain mid term endeavors. This project is a short term solution to bridge the gap between DRAGON and already designed ALAMOS geometries.

Chapter 3: Study domain & methodology

This chapter outlines the studied FAs, some of the inner workings of DRAGON as a platform and how it was configured for a first proof of concept minicore study. It must be understood that this work is not strictly about reactor physics, as such elements mentioned here will be looked at from a geometry point of view. It will also be a way to sketch SALAMANDER's requirements as a companion tool. For reactor physics analysis as well as in depths DRAGON5 and SERPENT2 configurations please refer to P. Fontaine's and P. Panisi's report [12].

3.1 Proposed minicore geometry

The proposed mini-core geometry is taken from the configuration outlined in the Karlsruhe Institute of Technology (KIT) paper and as part of the CAMIVVER project [26]. It is made up of one central 30AV5 FA and 6 390GO FAs, surrounded by a water reflector in an overall hexagonal pattern and the boundaries beyond the water reflectors are considered to be void (albedo of 0). Boron concentration in water is set to 600*ppm* to mirror KIT's configuration.

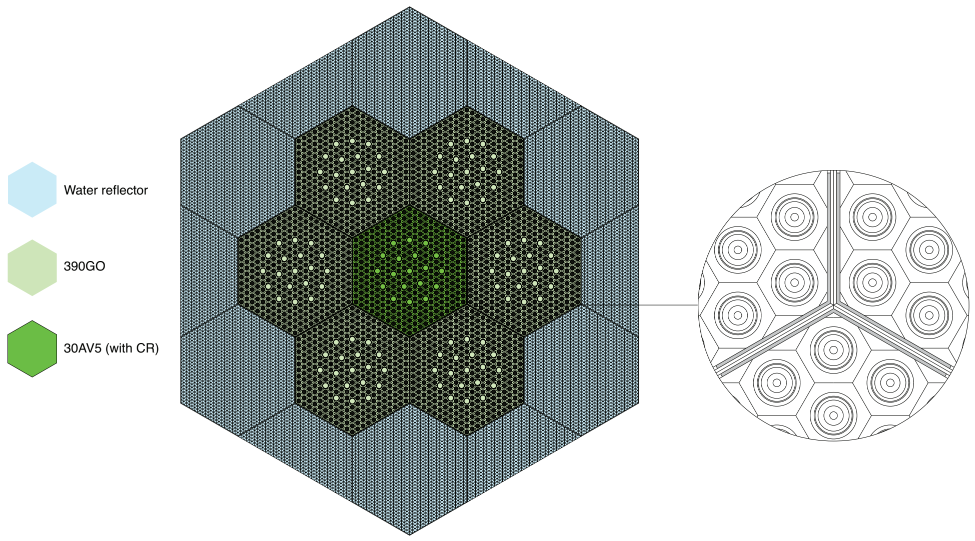
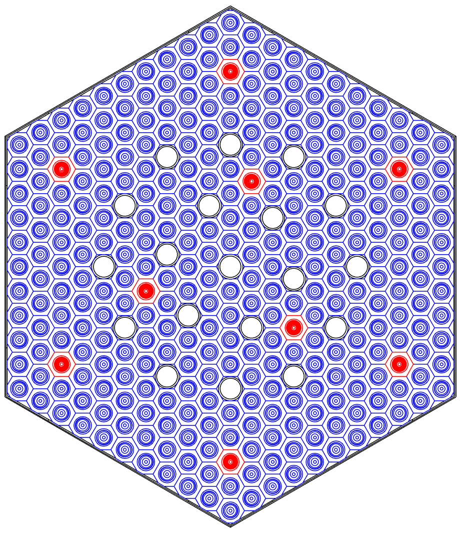


Figure 3.1: The minicore geometry as described in [26]

3.1.1 Fuel assemblies

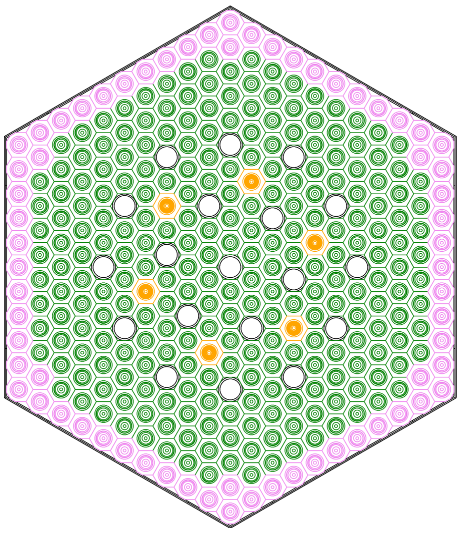
The 30AV5 fuel assembly only contains one type of fuel pin with one type of burnable absorber. Its burnable absorbers are located more in its periphery and the enrichment of its fuel is relatively low compared to a 390GO FA.



Cell type	Number
Non fuel	19
$3.0\%^{235}U$ fuel pin	303
$2.4\%^{235}U$ and $5.0\% Gd_2O_3$ absorber pin	9

Figure 3.2: The 30AV5 fuel assembly

The 390GO FA is made up of two types of fuel pins, with the lowest enriched ones being placed on its outer shell. This is done to prevent power peaks and neutron leaks at its boundary. Its burnable absorbers are set closer to its center.

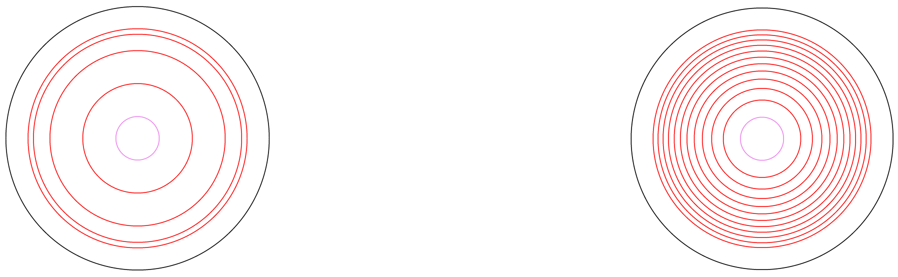


Cell type	Number
Non fuel	19
$4\%^{235}U$ fuel pin	240
$3.6\%^{235}U$ fuel pin	66
$3.3\%^{235}U$ and $5.0\% Gd_2O_3$ absorber pin	6

Figure 3.3: A surrounding 390GO fuel assembly

3.1.2 Fuel rod discretization

The fuel rod geometries have been modified with the removal of the outer helium void region to speed up calculations. The fuel cell part has also been radially split into increasingly smaller annular regions as radius increases. This is done to better model the effects of spatial self-shielding and the evolution of isotopes at different radii in the cells. Fuel cells have been split in 4 annular regions and burnable absorber cells in 12 annular regions.

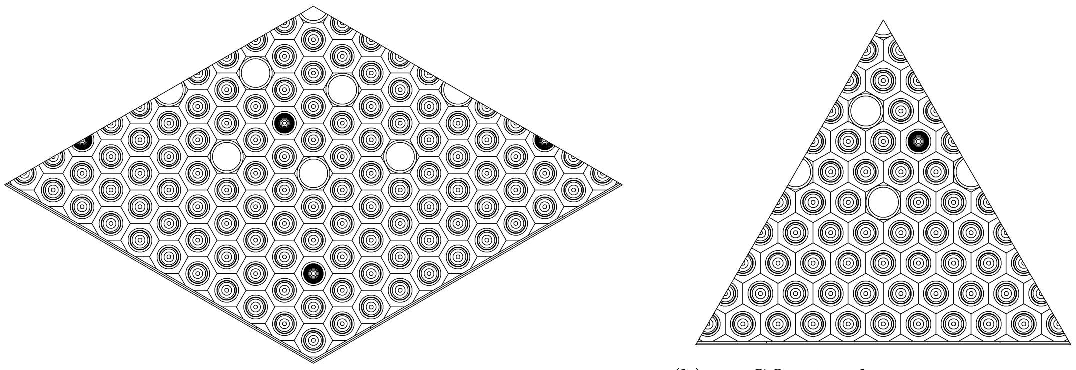

(a) Discretization of fuel cells

(b) Discretization of burnable absorber cells

Figure 3.4: Cell geometry discretization

3.1.3 Input geometries and construction challenges

The geometries of individual assemblies come in the form of smaller symmetry geometries which were split to save disk space and computation time when adequate boundary conditions are applied. The 30AV5 comes as a diamond geometry while the 390GO is split in an equilateral triangle, as seen in 3.5. Building a complete minicore geometry requires unfolding those input files, removing internal boundaries and then fusing them together. Additionally, as reflector geometries were not given they have to be produced and added onto the final design. Answering those needs constitute the core requirements for the SALAMANDER tool.


(a) 30AV5 diamond symmetry geometry

(b) 390GO triangle symmetry geometry

Figure 3.5: Input geometries

3.2 DRAGON5: setup & geometric data

DRAGON's module and data are accessed via the CLE-2000 language. It is used to control data flow, define routines and develop calculation schemes. In the following subsections the link between geometric data and nuclear data are explained and the configuration of modules used during the project is outlined.

3.2.1 Region representation

DRAGON has its own geometry module **GEO:** and also has the capability to import non-native ALAMOS / SALOMON surface geometries using the module **G2S:**. While these two types of geometries are defined and structured differently, they both divide space in what are called regions. Regions represent closed surfaces or volumes, they have a unique identifier index and a medium index. The medium index is used when building libraries and accessing nuclear data for this given region. It may be shared or be unique, depending on the type of medium and evolution observed in that medium's material. Regions bearing fuel or burnable absorber materials will be assigned a unique medium number for accurate self-shielding and evolution calculations. Otherwise, regions made up of structural materials will be assigned shared medium indexes. In this study the following medium indexes were shared across multiple region:

Medium	**Medium index**
Void / helium	1
Water	2
Stiffener material	3
30AV5 instrument guide tube material	4
390GO instrument guide tube material	5
30AV5 control guide tube material	6
390GO control guide tube material	7
30AV5 fuel cladding material	8
390GO fuel cladding material	9

Table 3.1: Shared minicore medium indexes

3.2.2 Library generation & Self-shielding

The data structure that links medium indexes to actual property tables and material definition is called a library, in DRAGON it is defined with the **LIB:** module. It builds a database of relevant nuclear data, such as cross-sections, from the materials defined by the user. The library may be further refined with calls to self-shielding modules on specific mediums and their isotopes to correct cross-sectional data to account for the flux depletion caused by self-shielding. Self-shielding modules run an extra flux calculation step to reevaluate some of the data associated with materials. In this project the **USS:** (Universal Self-Shielding) module was used with the P_{ij} method on native DRAGON assembly geometries. This simplifies the problem from a whole minicore calculation to two individual assembly calculations. However medium indexes in the native DRAGON self-shielding geometries and the minicore flux calculation geometry must match for data to be coherently written to the library.

Isotopes	Self-shielding parameter
^{235}U	Uniform self-shielding in all mixtures
^{238}U	Self-shielded by fuel region
^{239}Pu	Self-shielded by fuel region
$^{240}Pu,^{241}Pu,^{242}Pu$	Uniform self-shielding in all mediums
$^{154}Gd,^{155}Gd,^{156}Gd,^{157}Gd,^{158}Gd$	Self-shielded by fuel region
$^{90}Zr,^{91}Zr,^{92}Zr,^{94}Zr,^{96}Zr$	Uniform self-shielding in all mediums

Table 3.2: Self-shielded isotopes and associated local or global parameters from [12]

To accurately model the effects of self shielding at the assembly level, cells in the native geometries were defined by families. These families are meant to group cells with relatively similar behaviors and environments, so families are defined from a cell's composition and neighborhood. The 30AV5 being at the center of the minicore, having one fuel composition, one burnable absorber type and being exposed to 390GO FAs on all sides, its self-shielding families are discretized radially and around peculiar elements like control rod guide tubes and fuel absorber cells. The 390GO FAs are more challenging because of their varied surrounding and their two fuel compositions. As such their boundaries are discretized by side and angle cells are placed in to account for discontinuities between two sides.

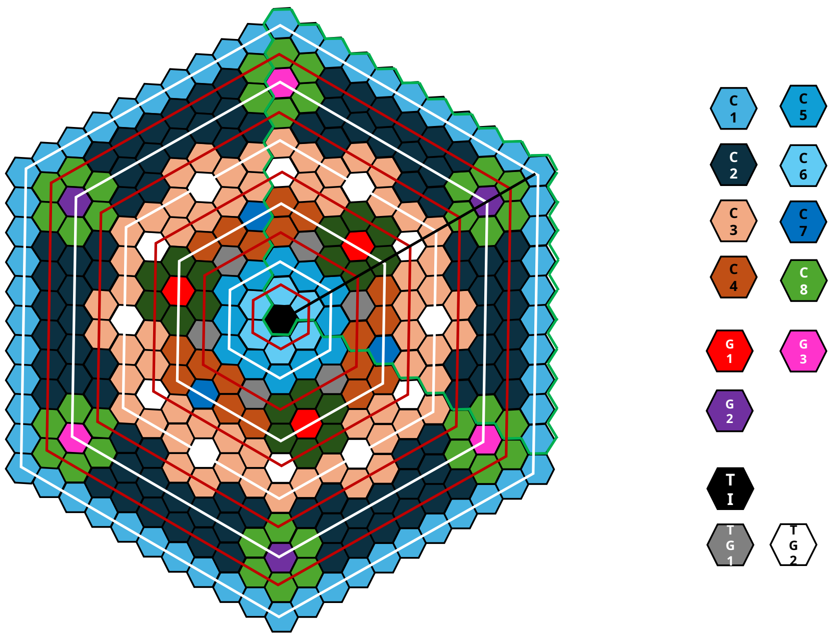


Figure 3.6: 30AV5 cell families

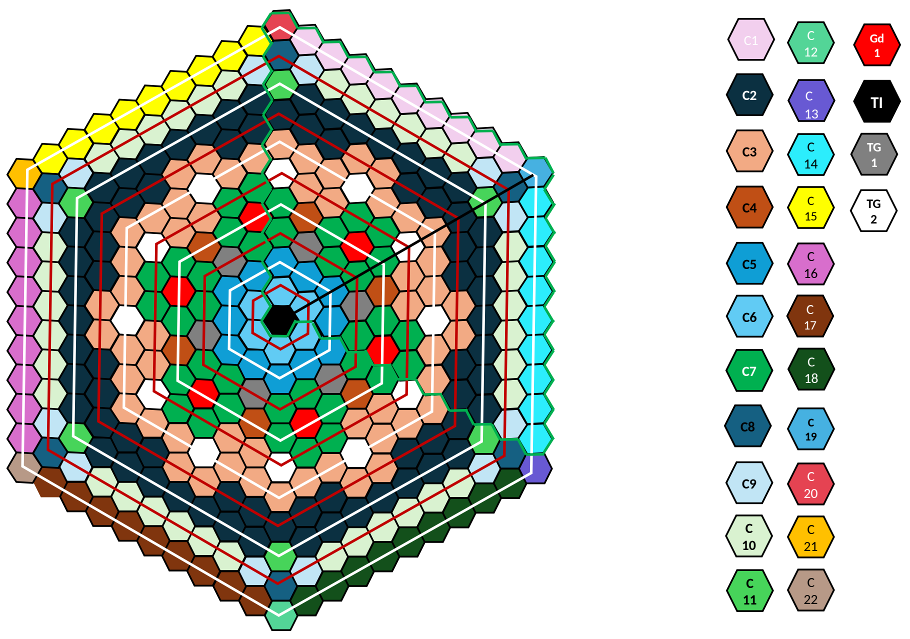


Figure 3.7: 390GO cell families

3.2.3 Tracking & flux calculation

The tracking process, as seen in 2.2.2, runs neutron trajectories through the study domain, splitting each tracks by regions. Tracking data is then passed to BTE solvers which calculate the flux in each region. Inside a region flux is assumed to be uniformly distributed, as such, it is important to keep regions small and sometimes split them for more precise results. During flux calculation a region's medium data is looked up in the library to use appropriate nuclear data. The surfacic minicore geometry is processed by the **SALT:** module whose track data is fed into the **MCCGT:** MOC tracking module. After flux calculation the **EVO:** module is called to calculate burnup, the evolution of isotopes in the minicore and other relevant key of reactor quantities.

Track density (cm^{-1})	25
Number of polar angles	4
Distribution of polar angles	CACB (uniform)
Boundary condition	TISO (isotropic reflection) and void boundary

Table 3.3: MOC tracking parameters for the whole minicore geometry

3.2.4 Data manipulation & tracked parameters

At last, data computed during the flux calculation and evolution procedure can be manipulated and processed by the **EDI:** module. In this project it was used to group fuel region data by cells and to condense the 295 energy group data into 2 energy groups. This processed output can then be compared to similarly formatted data from SERPENT2 with post-processing tools. The data manipulation step requires precise knowledge of the geometry and its medium indexes to coherently bundle mediums together.

Studied isotopes	$^{235}U,^{238}U,^{239}Pu,^{157}Gd,^{135}Xe,^{149}Sm$
Condensed energy groups	2 (slow and fast neutrons)
Energy group split	0.625 MeV
Output data	NWTO (flux), H-FACTOR (power factor) NFTOT (fission cross section), NG (radiative capture cross section)

Table 3.4: Tracked study parameters

3.2.5 Conclusion: Interfacing with DRAGON

With the VVER minicore being made up of 6 FAs, each consisting of 321 cells, themselves split in at least 6 regions, the total region count grows to numbers too great for human operators to handle when inputting or manipulating data in DRAGON. As such, the SALAMANDER tool has to provide a coherent region and medium indexing solution as well code generators to automate library generation and data manipulation. SALAMANDER must be able to make sense of abstract structures like cells and assemblies to properly number regions and split them for assembly by assembly self-shielding calculations. On top of that, geometric and numbering operations must be rigorously conducted to produce coherent closing shapes for the track generators to work with.

3.3 Minicore calculation scheme

The minicore's calculation scheme takes it roots in the previous POLYVVER project which succeeded in adapting a REL2005 calculation scheme to VVER assemblies in DRAGON. However, as the minicore is a large and complex object to simulate, concessions had to be made. The REL2005 scheme is introduced to clarify some notions and terms used in the demonstrator minicore scheme.

3.3.1 POLYVVER's REL2005 basis

REL2005 is a PWR FA calculation scheme developed at the CEA. It is said to be a two layer scheme because of its use of two distinct geometries for flux calculation, transformed into *Track_ 1L* and *Track_ 2L* in the illustration below. The nuclear data library and region mixes are generated and the first flux calculation takes place with a coarse geometry and the P_{ij} method. It is used to reevaluate the data in the library for a second flux calculation, this time with a finer geometry and the MOC method. The result of this flux calculation is then used for data edition or evolution of isotopic concentrations. Self-shielding calculations take place only at the initial step and at set burnup points to save computation time, it is computed by P_{ij} method on a coarse geometry. [34] The scheme may be hybrid, using two different types of geometry, 100% ALAMOS or 100% native, using only ALAMOS or native GIGOGNE geometries respectively.

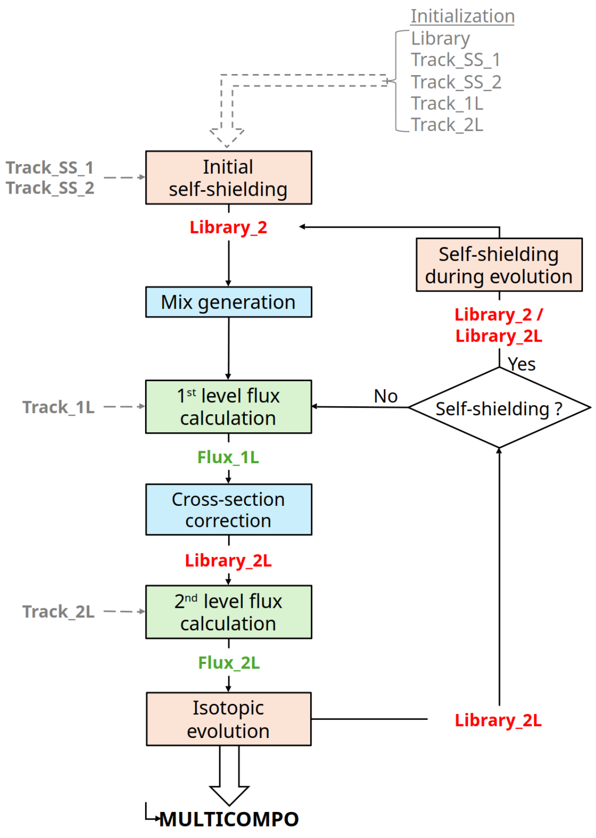


Figure 3.8: REL2005 calculation scheme flowchart

3.3.2 Minicore adaptation

With technical limitations encountered during the project, notably in terms of computation time and library size, a simpler scheme was developed. The two layer REL-2005 approach was dropped for just one MOC flux calculation which is more appropriate for large objects. The geometry used for flux calculation is a coarse one provided by SALAMANDER. Self-shielding calculations take place on smaller assembly geometries and not on the whole minicore geometry. While those compromises degrade the quality of the results they are sufficient to prove the feasibility of a minicore and DRAGON. These results are only meant to be compared with SERPENT2 to identify the points to be improved in the calculation scheme and, if really limited, in DRAGON.

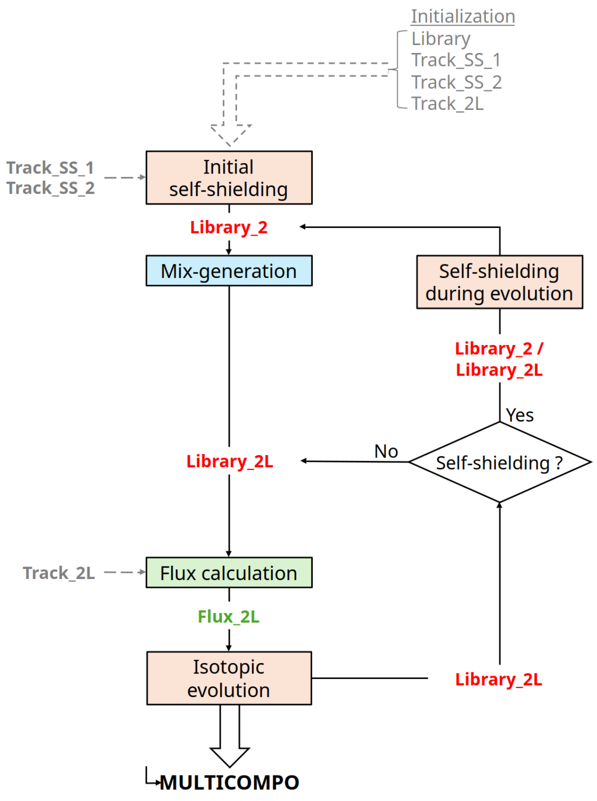


Figure 3.9: Minicore calculation scheme flowchart

Chapter 4: The Salamander geometry assembler & processing tool

Salamander was developed to process multiple mother assembly geometries and output a multi-assembly child geometry with project specific CLE-2000 companion scripting code for easy plug in with DRAGON5. Its main interest lies in its code generation and multi-layer geometric identification abilities. This chapter goes over SALAMANDER's architecture and functionalities. It is done so that external users may grasp the inner workings of SALAMANDER to work with it, to expand it or to build new codes from the experience acquired when developing SALAMANDER. Attention is first given to its global architecture and the basic **Element** class it manipulates. Then abstract containers, geometric operations and indexing operations are introduced. Finally, focus is set on the minicore design procedure.

4.1 General architecture

SALAMANDER works with ALAMOS and SALOMON surfacic formats, describing 2D BRep geometries, it can manipulate them with basic geometric operations. BRep's design flexibility on the user side also hides layers of complexity for later processing as geometries prove completely unstructured 2.5. This lack of direction and hierarchy makes analyzing a BRep geometry and identifying its superstructures costly and complex. SALAMANDER's role is to provide and build abstract data structures for these unstructured geometries. Its key guiding principle is to build appropriate containers for both internal geometric operations and the user's needs. SALAMANDER was developed in PYTHON3 using an object oriented approach to quicken access to and modification of data structures. It also has a TKinter graphical user interface for visual inspection of output geometries although this module remains completely optional.

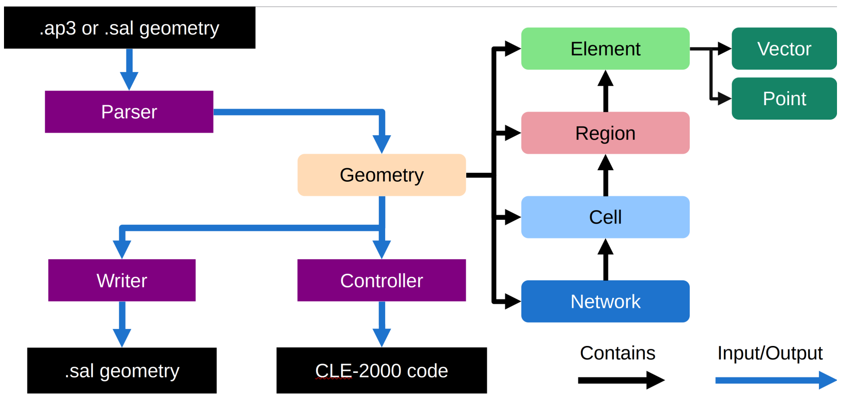


Figure 4.1: SALAMANDER's data flow and architecture

SALAMANDER is built around a master container class named **Geometry** which stores **Element** and **Region** objects as well as other meta information regarding the geometry's precision and boundary conditions. It is used to apply any type of geometric operation on the whole set of elements and regions as some require intricate coordination between several layers of the geometry. **Geometry** classes can be copied (*copy()* method) or added (*add(geom)* or *batchAdd([geom])* methods) together as well, their elements and regions are just renumbered and appended to the mother geometry's lists.

Additionally, the **Geometry** class can contain two abstract data structures that appear as part of the analysis process:

- Cells: a list of **Cell** objects defined from its **Region** objects
- Networks: a list of **Networks** objects, which contain and operate on **Cell** objects

These structures are then read by the **Controller** class which is used to renumber the region indexes and mediums according to user input and the underlying structure of the geometry. It must be called just before saving the geometry to disk and it is tasked with 2 main functions:

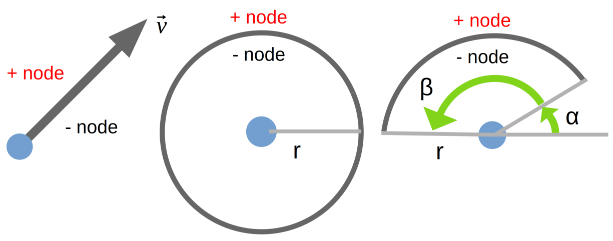
- Renumbering regions for export according to user input, region material and earlier analysis
- Generating CLE-2000 code and correspondence tables for the user

4.1.1 Element representation

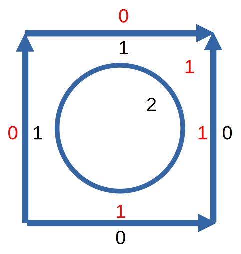
As seen in 2.11 during the literature review, ALAMOS / SALAOMON geometries have 3 elementary oriented shapes:

- Lines: defined by a starting point and vector $\vec{v}$
- Circles: defined by a center point and a radius r
- Arcs: defined by center point, a radius r, a starting angle α and an opening angle β

Elements are oriented, they have a positive and negative side each bearing a region, or node, index. The positive direction for ALAMOS / SALOMON files is defined as clockwise and the limits of the study domain are marked by a 0 region index.



(a) Orientation of base elements



(b) A simple surface geometry

Figure 4.2: Geometry representation in SALAMANDER

4.1.2 Data formats

SALAMANDER can manipulate two data formats with one being preferred for input, .ap3 ALAMOS files and one for output .sal SALOMON files. They have the exact same geometric representations and basic elements, making converting from one format to another a lossless operation. The use of two file formats is due to the region data they contain.

ALAMOS format

ALAMOS geometries come in two ASCII files:

- **.ap3** file : sets of points and orientated elements linking those points, these elements have set regions on their positive and negative sides, it also defines region materials
- **.za** file : material, temperature and property maps

The .ap3 file is sufficient to build the geometry and is easily parsed. The region material information is key when later analyzing the core and is not easily found in SALOMON files, hence the use of .ap3 files as inputs. SALAMANDER only parses .ap3 files but doesn't output them, SALOMON files are selected for DRAGON exports.

SALOMON format

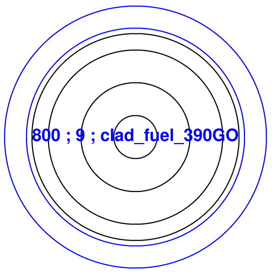
SALOMON files are used as output for their explicit medium numbering, which is later used when coupling with DRAGON. They also pack more data with them such as boundary conditions, perimeter information and some meta information about the geometry's precision. For now SALAMANDER only outputs SALOMON files with default meta values taken from DRAGON's ALAMOS to SALOMON export module **G2S:**. SALAMANDER may both read and write SALOMON files, however a dedicated material map is needed to make sense of medium numbering in SALOMON files.

4.2 Multi-layer containers

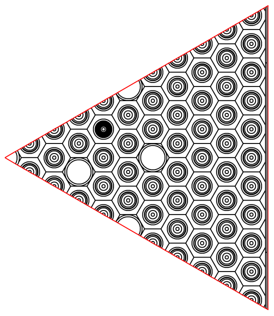
SALAMANDER's analysis and code generation abilities arise from its variety of object containers which may be nested into one another. It fills the void left by the unstructured, unordered BREP elements. It translates needs and data structures from the DRAGON such as cells and assemblies.

4.2.1 Regions

Regions are defined by a unique region index, a medium index, a material and the set of **Element** objects they contain. The material string is later required when assigning medium indexes in the **Controller** class. **Region** objects are automatically constructed when parsing a geometry file and they are assigned to the corresponding nodes of the **Element** objects they contain. Region 0 does not actually exist but is created and reserved for quick access to boundary **Element** objects.



(a) A simple region in blue with (index ; region_index ; material)



(b) Region 0 in red

Figure 4.3: Two example of regions as represented in SALAMANDER

4.2.2 Cells

Cells are sets of concentric regions defined by a unique index, a composition type and a finer arbitrary type. The composition type is a list of the **Cell**'s **Region** materials. The type is a more refined indicator used for self-shielding groups, by default the type is set to the composition type, the user needs to define its own types later.

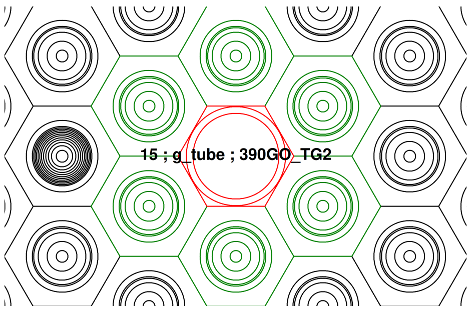


Figure 4.4: An example of a cell in red with its characteristics (index ; composition_type ; type) and its physical neighbors in green

Cells are identified and built with **Cell**'s *identify([regions])* which returns a list of cell objects. This process automatically builds the cells indices and defines composition types and also assigns the corresponding mother **Cell** object to children **Region** objects within itself. Regions are grouped by centers and then sorted by radius for simplified access to data. **Cell** objects may contain several regions in one radius group in the case of ultra fine wedged geometries.

4.2.3 Networks

The **Network** class is the highest level container, representing a fuel assembly. They are made up of **Cell** objects and defined from a reference center **Cell**, radii groups are then identified and **Cell** objects are assigned in corresponding radii group. When a **Cell** has less than 6 neighbors it is considered on the outer limit. **Network** objects allow simple access to these radii of **Cell** objects for quicker type assignment by users based on geometric position and not arbitrary cell indexes. They are also used to identify a the different assembly types in a given **Geometry**.

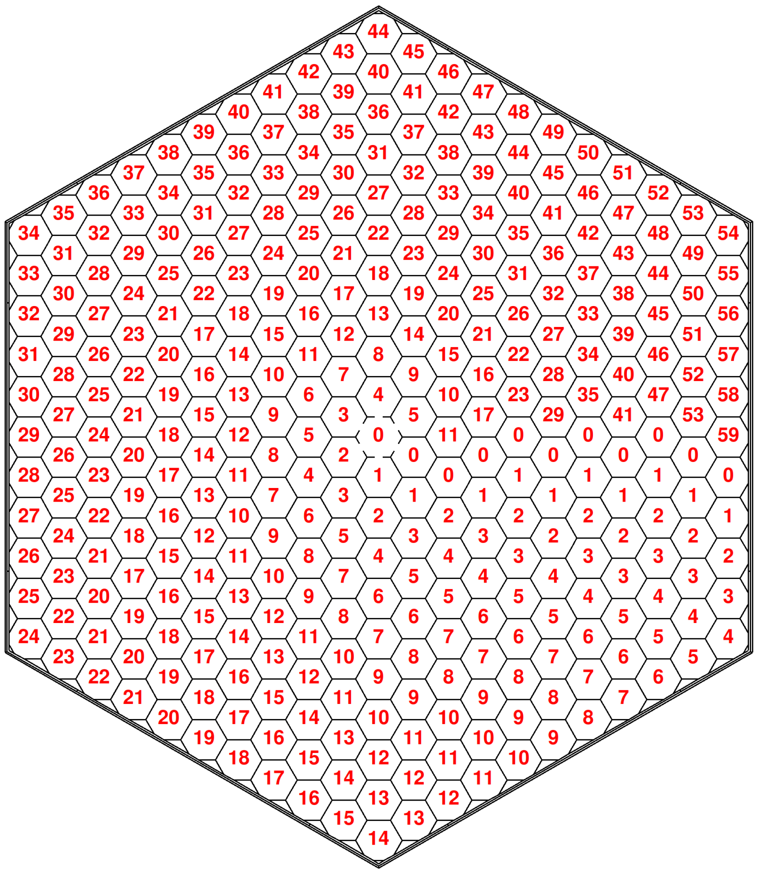


Figure 4.5: Cell numbering by radius and position within the radius in **Network** objects

4.3 Geometric operations

Geometric operations can be divided into two main types, elementary movement operations only applied on **Element** objects and structural changes applied on **Element** object nodes and **Region** objects.

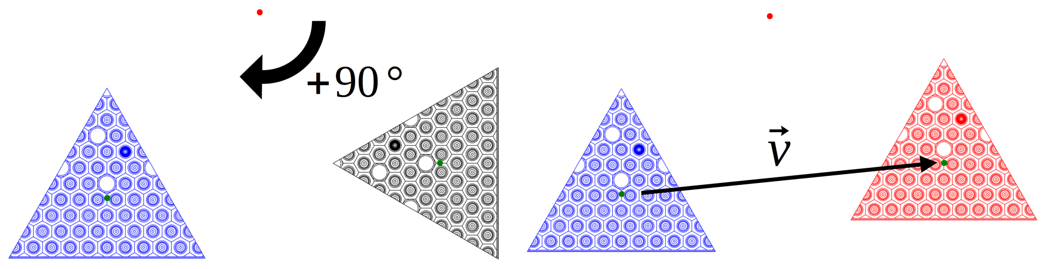
4.3.1 Elementary operations

Translation proves trivial in the data representation used in SALAMANDER, the element's origin point or center just sees the translation vector's coordinates added or subtracted to its own coordinates. This is done using the *translate(vector)* method of the **Geometry** class.

Rotations are done with the help of a rotation matrix, it is applied to each element's point. For arc circles the rotation angle is also added to their opening angle and for lines their vectors are rotated as well. A θ rotation of point $\vec{r}$ to point $\vec{r'}$ can be written as :

$$\vec{r} = R(\theta)\vec{r} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} xcos(\theta) - ysin(\theta) \\ xsin(\theta) + ycos(\theta) \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{4.1}$$

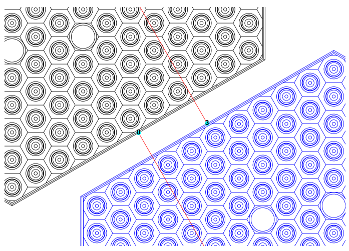
This rotates the point about the origin, which is not useful in SALAMANDER's use cases so relative rotation was introduced. By default geometries are rotated about their centers in a clockwise direction, this is done with the *rotate(angle)* method. Other reference points can be manually inputted with *rotateRel(Point, angle)*. Relative rotation involves correcting the rotation with a translation to the original reference point's coordinates. So, for a reference point $\vec{r}$, which would be rotated to $\vec{r'}$, the correction translation vector can be written as : $\vec{c} = \vec{r} - \vec{r'}$. These two steps are illustrated in the figure below, the original geometry in black is first rotated, in blue, then returned to its original center, in red. The red point represents the origin and the green points are the centers of the geometries.



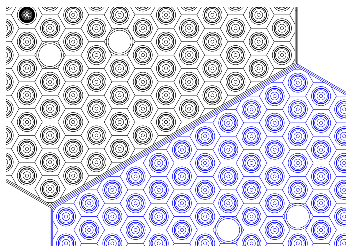
(a) Effect of the rotation matrix on the geometry (b) Translation to the geometry's original center

Figure 4.6: Illustration of a relative rotation combining a rotation and a translation

To simplify the construction of the minicore the snap point concept was introduced. Snap points are actually line **Element** objects perpendicular to outer boundary elements in a geometry, they start at the boundary **Element**'s center and extend outwards. When snapping two **Geometry** objects onto each other, one is referred to as a mother **Geometry** with its snap point serving as references, and the other is the child **Geometry** being translated. The child's snap vectors are compared to find the exact opposite of the mother's and then the child is translated so that its snap point overlaps with the the mother's. This is done by calling *snapOn(extGeo, pointIndex)* on the mother **Geometry**.



(a) Definition of snap points



(b) The blue geometry has been glued to the black one

Figure 4.7: The snapping mechanism in action

4.3.2 Connecting geometries

Connections are operations involving the detection of overlapping interfaces or boundaries in a geometry and reassigning some **Element** objects to appropriate **Region** objects. As these operations take place on several objects of different types they also require a higher level object to piece everything together, these operations are once again methods of the **Geometry** class.

Linking interfaces

The linking operation is meant to be used when connecting two fully defined geometries. It involves what are called interfaces which are **Element** objects with a node set to a negative integer value instead of an adequate **Region** object. This interface is meant to be paired with another overlapping one. Once paired only one of the two **Element** is kept and its interface node is assigned the **Region** object of the other **Element**. This last element is deleted and removed from its none interface region.

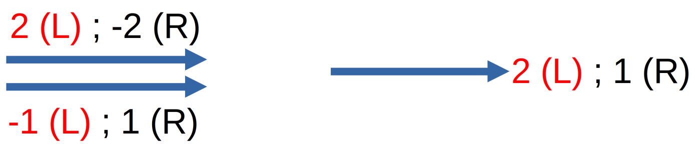


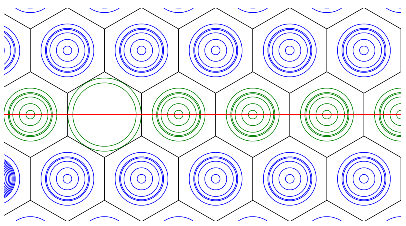
Figure 4.8: Connecting two paired interfaces

Connections are automatically made when using the *connect()* method from the **Geometry** class if interfaces were previously defined. The user can user manually create them or use *bound2inter('all')* and *inter2bound('all')* methods respectively before and after the use of the *connect()* method to open and close interfaces on all boundary elements after snapping two **Geometry** objects onto each other.

Fusing boundaries

Boundary region fusion is called with the *fuse()* method of the **Geometry** class and is only applied on boundary **Element** objects. It is used when unfolding symmetry geometries, removing their inner split and reconstructing their exact shape. It is not to be used when connecting two different geometries as it will remove some of their outer shell elements and degrade the finesse of the resulting geometry. Its inner workings are as follows:

1. Group overlapping boundary **Element** objects by fusion groups
2. Pick a master R_m **Region** per group to receive **Element** objects from other R_i **Region** objects in the group
3. Replace **Element** nodes bearing R_i with R_m and add these **Element** objects from R_i to R_m
4. Extend connecting arc-circles or transform them into circles if their opening angle ≥ 360
5. Delete R_i **Region** objects and overlapping boundary **Element** objects


(a) Before fusion, overlapping boundary elements in red

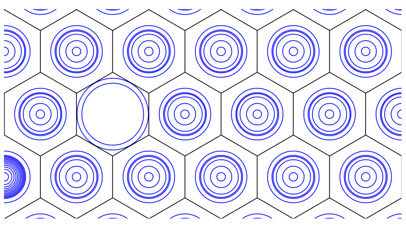

(b) After the fusion operation

Figure 4.9: Illustration of the boundary fusion mechanism, arcs in green, circles in blue

4.4 Indexing operations & code generation

As seen during 3.2, the geometry itself is just an empty shell describing sets of regions and their limits, it is only when coupled with a library that it can be sent to flux or self-shielding modules. SALAMANDER comes with its own CLE-2000 code generators to speed up geometry export and avoid human error when working on large domains such as a minicore. These operations are handled by the **Controller** class and require the **Geometry** class to have defined **Cell** objects and **Network** objects.

4.4.1 Medium indexing

When passed a **Geometry** object, the **Controller** class will automatically renumber its region indexes and region medium indexes. It looks the *MediumIndexing.conf* file in the *Settings* folder of SALAMANDER. In this file the user defines correspondence between material names as defined in the input .ap3 geometry and medium indexes as defined in the output .sal geometry and outlined in 3.2.1. They are defined with the following syntax:

$$\text{material\_name} = I_{medium}$$

Then a renumbering and sorting algorithm is applied to ensure a replicable and coherent region indexing.

1. Assign structural mediums to **Region** objects with structural materials
2. Go through each **Cell**'s **Region** by smallest radius and assign a unique medium index if the **Region** bears a non structural material
3. Sort **Region** objects by medium index
4. Renumber **Region** indexes by their position in the previously ordered list

4.4.2 Code generation

When generating code for DRAGON, SALAMANDER identifies what are called generator cells. A generator cell is the lowest indexed **Cell** of a given type, the medium indexes of its **Region** objects are used to define the native DRAGON geometry. With those generator cells identified, CLE-2000 code generation is made possible with the following methods:

- *generators()*: writes an easily readable correspondence table with all the structural mediums, generator cells and their composition so that users may quickly inspect SALAMANDER's output and use it to build their native DRAGON geometries
- *autop()*: produces the **mixA1_MCORE.c2m** file which generates the library and self-shielding library code
- *edit()*: generates the **EDIT_MCORE.c2m** file for data manipulation at the end of flux calculations and for later outputs
- *mix()*: generates the copy lines for the **MIX_MCORE.c2m** file with the generator mediums

4.5 Minicore design in SALAMANDER

Building the minicore geometry in SALAMANDER is ensured by the **MCORE()** method. This section outlines its inner workings and illustrates SALAMANDER's full capabilities and module chaining.

4.5.1 Assembly construction

Individual FA geometries are loaded from the *InputGeometries/[Geometry name]* folder in SALAMANDER's directory and opened with the **parse([Geometry name])** method which returns a **Geometry** class. The FA geometries are then unfolded with the*triangleUnfold(Geometry)* and *diamondUnfold(Geometry)* methods which work as follows:

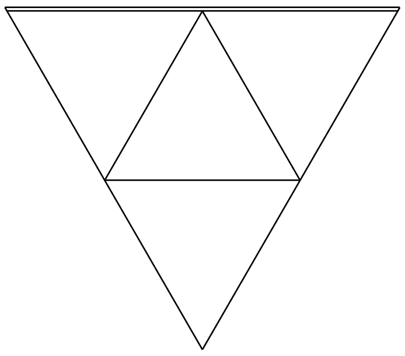
- Copy of the mother symmetry geometry
- Rotation about the identified center of the assembly
- Addition of all the children rotated geometries to the mother geometry
- Fusion and connection of internal border and interface elements

4.5.2 Assembly analysis and connection

Each unfolded FA is then analyzed by building its **Cell** objects and **Network** objects. FAs are automatically identified by the **Network** class and cell type assignment procedures are called. The 390GO FAs are rotated to ensure that corresponding cell families always face the same neighbors on their boundaries. They are then snapped onto the central 30AV5 assembly and added to to it. Their borders are turned into interfaces to connect neighbor assemblies and the remaining interfaces after connection are turned back into boundaries.

4.5.3 Adding reflectors

The water reflectors are the only actual geometry entirely produced by SALAMANDER. They are built from equilateral triangles subdivided into a variable number of other equilateral triangles. They are then then bundled into trapezoids or full hexagons in an operation very much analogous to the unfolding of FAs. A reflector of matching VVER FA dimensions is returned when calling the *equiHex(N, nFold)* method. They are rotated, snapped onto the mother minicore **Geometry**, added to it and connected in the same way as individual assemblies before. This project retained $N = 20$ as a default value for reflector mesh finesse.


(a) A reflector triangle split with $N = 1$

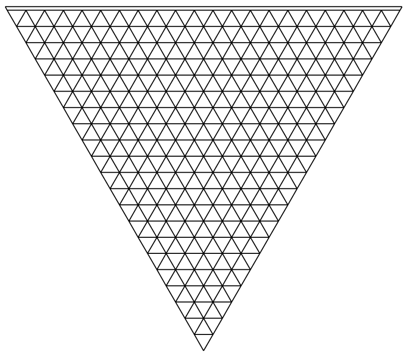
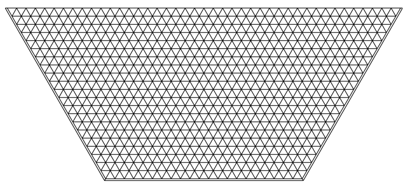

(b) A reflector triangle split with $N = 20$

Figure 4.10: Two different equilateral triangle splits for the water reflector


(a) Reflector trapezoid with $nFold = 3$

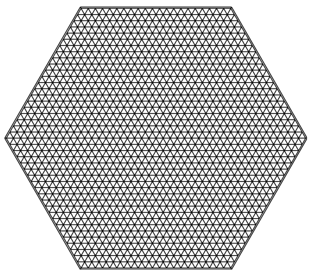

(b) Reflector hexagon with $nFold = 6$

Figure 4.11: Output trapezoid and hexagonal reflectors for the minicore with $N = 20$

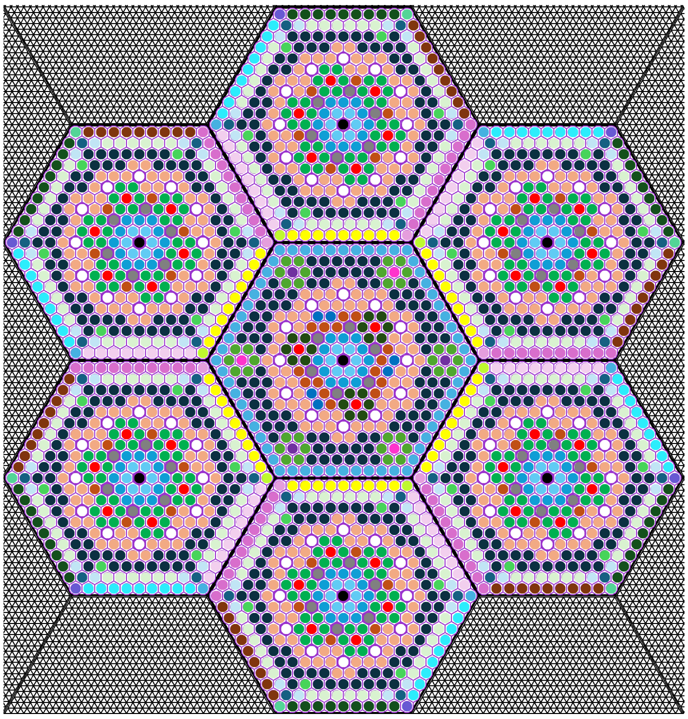
4.5.4 Final analysis and outputting files

When the geometry is ready for export the **Controller** class and its code generation methods are called and then the *Writr.write(Geometry)* method is called. They both output their files to the *OutputGeometries/[Geometry name]* folder.

Chapter 5: Results & analysis

5.1 Final SALOMON minicore geometry

The minicore geometry and the companion code to couple it with DRAGON5 are results in and of themselves. The geometry is able to be tracked with the parameters supplied in 2.2.2 and the code is fully read by DRAGON with no errors.

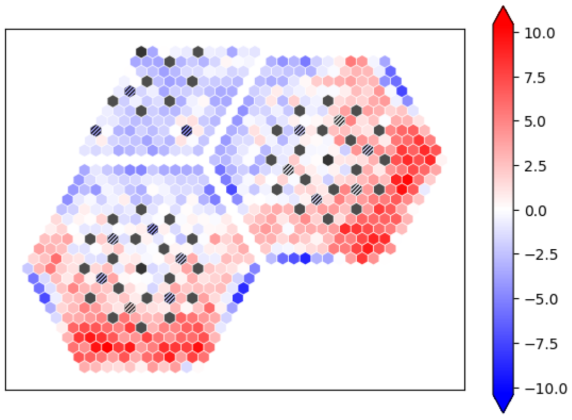


Total element count	59413
Total region count	40490
Total cell count	2247
Unique cell types	40

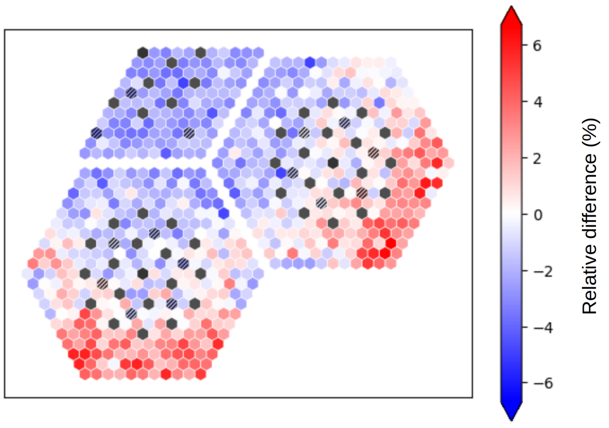
Figure 5.1: The output minicore geometry zoomed in on assemblies with their cells colored by type

5.2 DRAGON5 simulation & comparison with SERPENT2

For easier comparison and visualization, the simulation results are observed on a third of the minicore geometry but symmetry ensures that results observed here are the same in the other two thirds of the geometry. This report only considers static results at burnup 0 serving as a demonstrator and already proving the geometry's working state. SERPENT2's results are taken as a reference for relative difference comparisons. Gadolinium cells are hatched while the empty instrument and control rod guide tubes are filled with black.

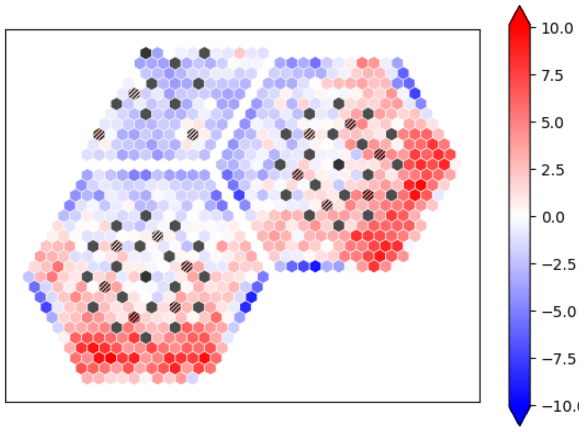


(a) Fission rates in the slow group

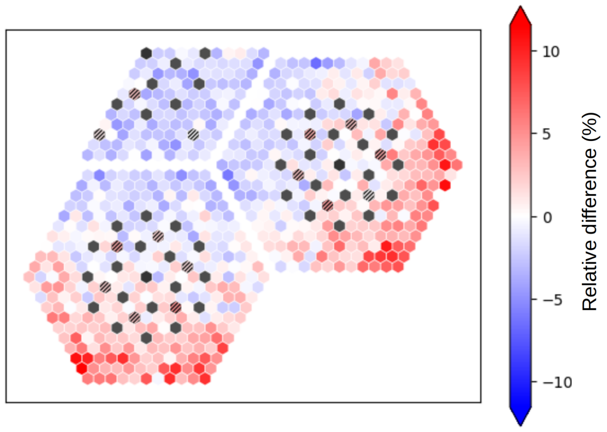


(b) Fission rates in the fast group

Figure 5.2: Comparison of minicore fission rates



(a) Radiative captures in the slow group



(b) Radiative captures in the fast group

Figure 5.3: Comparison of minicore radiative capture rates

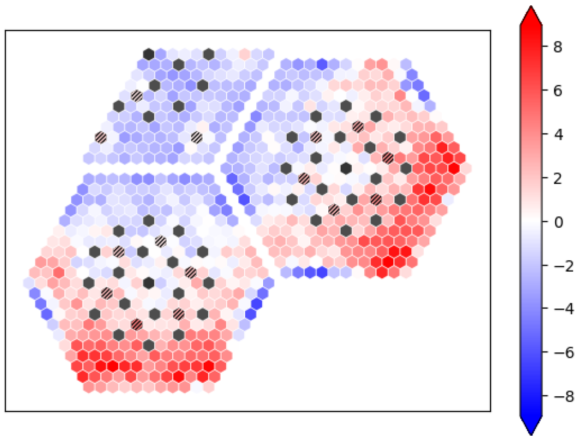


Figure 5.4: Comparison of total absorption rates in the minicore

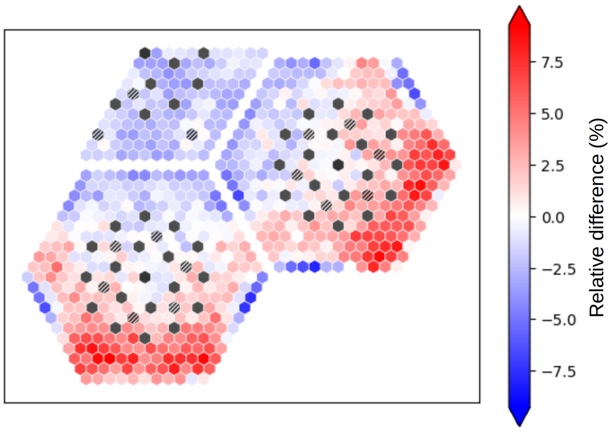
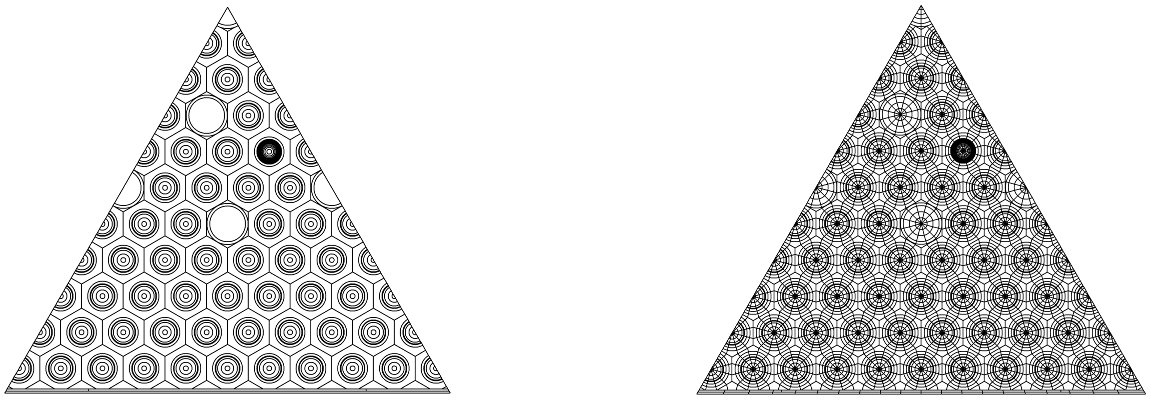


Figure 5.5: Comparison of total power factors in the minicore

A global pattern emerges from the comparison of minicore data: DRAGON overestimates all data in the periphery of the minicore while underestimating them around the center. Cells with the lowest relative differences are located around the centers of 390GO FAs and in the central 30AV5 FA. Overall, DRAGON5's results are decent for a proof of concept phase, falling in the same orders of magnitude as SERPENT2's, however they are not satisfying for any consequential minicore studies and they remain above the 1% error margin the previous POLYVVER had set for result validation [11]. Sources of errors must come from the calculation scheme and / or the geometry.

The currently observed discrepancies could be in part explained by the geometry's finesse or lack thereof as the previous POLYVVER project used an ultra fine flux calculation geometry and here only the coarser geometry is used to be build the minicore. The differences can be appreciated in 5.6 on the 390GO triangle. The ultra fine geometry would have been very costly for minicore simulations but it can still be processed by SALAMANDER.



(a) Currently used coarse geometry

(b) Ultra fine geometry

Figure 5.6: Comparison of the finesse of available geometries

A more local observation reveals 390GO FA boundaries behave differently depending on their neighbors:

- 390GO-30AV5: these borders' data is systematically underestimated and does not show any particular space dependence
- 390GO-390GO: data is underestimated closer to the 30AV5 and then slightly overestimated before dipping again
- 390GO-Reflector-390GO: data is underestimated closer to the neighbor 390GO and is then overestimated farther
- 390GO-Reflector: data is systematically and the most notably overestimated

These discrepancies could be explained in part by how this calculation scheme treats self-shielding. The actual self-shielding flux calculation is applied over individual assembly geometries set in infinitely repeated domains of themselves. As such, no concern is given to the actual neighborhood and boundaries of a given assembly. This proves somewhat reasonable for boundaries involving the 30AV5 FA as these boundary regions are made up of fuel rods on both sides even though they do not share the same enrichment values. On the other hand, it is particularly problematic when dealing with fuel-water boundaries, as self-shielding only considered a fuel-fuel environment. These observations are further proven by the gradient observed on the 390GO-390GO borders which go from under to overestimating data, with middle points showing really small relative differences with SERPENT2. A costly but effective way of reducing relative differences would be to perform self-shielding calculations on a whole minicore geometry. Individual assembly geometries could also be configured with proper boundary conditions although this method will not completely erase the errors on the boundaries and would require more in depth studies of the flux received and sent between FAs.

While there could be other factors skewing the results such as the finesse of the geometry, self-shielding seems to be the major explanation for current differences. This is however a positive note as this proves that actual minicore flux calculations are feasible in DRAGON5 and that the limit is more calculation scheme side than software side. Moreover it shows that even with the current simplified calculation scheme DRAGON5's results are within the order of magnitude of reference results provided by SERPENT2. It must be noted that DRAGON5 was updated to accommodate some of this project's needs such as OpenMP parallelization or library concatenation. The improvements and additions made to DRAGON's code which have been tested during this project are also a major steppingstone for more refined minicore studies, whether for VVER cores or not.

Chapter 6: Conclusion

6.1 Current and future state of the POLYVVER project

With an initial minicore calculation scheme and geometry being proven to work in DRAGON, this step of the project proves conclusive and provides a base for future studies. New features implemented to DRAGON5 from this project's needs have lifted most of the technological barriers preventing minicore studies, namely library size and calculation time. The first results have shown that the challenges now lie mostly on the calculation scheme side than in the geometry itself or in DRAGON. From these elements, it can be understood that much of the future work lies in developing a more refined calculation scheme to better model the effects of self-shielding. If results prove conclusive with much reduced relative differences, minicore studies in complex scenarios mirroring that of KIT would become possible. With these intermediary steps providing reliable, precis results, full core simulations in DONJON5 could be envisioned. Additionally, some performance tuning may be of interest or even required to find optimal OpenMP parameters to speed up calculations with improved parallelization..

6.2 Outlooks for SALAMANDER and geometry tools

This current minicore project was in part made possible by the SALAMANDER tool to both build and process a uniquely large study domain. It has proven its relevance by providing a coherent and trackable geometry for flux calculation. The companion CLE-2000 scripting code it generated has accurately translated its material composition and medium location. In its current state can be also be considered satisfying for the project and some future uses. It remains a rather limited tool with no graphical editing abilities, its code generators are only project specific and the geometry files it supports are meant to be defined with external CAD tools. As such, it is meant to be superseded a longer term solution that would involve the developmen of an an open-source data format and a module for the SALOME. This would allow the user to define geometries with a complex structure and have access to more advanced operations, such as shape coherence checks and shape healing procedures. Preliminary parts of this work, mainly on the data format and module architecture are already being undertaken byD. Manzione, a newcleo engineer detached to the IGN to explore the possible routes to develop such a software. This project would be named GLOW (*Geometry Layout for OpenCascade Workflow*) and would offer an alternative to current proprietary SALOME modules like ALAMOS. GLOW would make reactor geometry design a simpler and more streamlined experience, eliminating the need for project specific glue tools.

Bibliography

[1] *A VVER-1000 LEU and MOX Assembly Computational Benchmark*. Tech. rep. Nuclear Energy Agency, 2002. URL: `https://www.oecd-nea.org/upload/docs/application/pdf/2020-01/nsc-doc2002-10.pdfl`.

[2] B. Vezzoni A. Willien. *D4.3 – Definitions of tests cases for the verification phases of the multi-parametric library generator*. Tech. rep. Electricité De Francce, Feb. 2021.

[3] Y. Bilodid, E. Fridman, and T. Lötsch. "X2 VVER-1000 benchmark revision: Fresh HZP core state and the reference Monte Carlo solution". In: *Annals of Nuclear Energy* 144 (2020), p. 107558. ISSN: 0306-4549. DOI: `https://doi.org/10.1016/j.anucene.2020.107558`. URL: `https://www.sciencedirect.com/science/article/pii/S0306454920302565`.

[4] R. Camarero. *MEC6212 - Generation de maillages : Une introduction par la pratique*. May 2024.

[5] Direction de l'énergie nucléaire Commissariat à l'énergie atomique. *La neutronique*. Editions le moniteur, 2015. ISBN: 978-2-281-11371-6.

[6] European Nuclear Assistance Constortium. *ANALYSIS OF VVER SAFETY DOCUMENTATION*. Sept. 1993. URL: `https://nuclear-safety-cooperation.ec.europa.eu/contracts/analysis-vver-safety-documentation_en`.

[7] *Contribution au développement d'un schéma de calcul basé sur le code DRAGON5 pour l'étude neutronique de crayons et d'assemblages de réacteurs de type VVER*. Master thesis. Aug. 2023.

[8] A. Bruneton D. Tomatis F. Bidault and Z. Stankovski. "Overview of SERMA's Graphical User Interfaces for Lattice Transport Calculations". In: *Energies* (Feb. 2022). DOI: `10.3390/en15041417`. URL: `https://doi.org/10.3390/en15041417`.

[9] *ENDF/B-VII.1 U-235 Principal cross sections.*

[10] Jordan A. Evans et al. "Burnable absorbers in nuclear reactors – A review". In: *Nuclear Engineering and Design* 391 (2022), p. 111726. ISSN: 0029-5493. DOI: `https://doi.org/10.1016/j.nucengdes.2022.111726`. URL: `https://www.sciencedirect.com/science/article/pii/S0029549322000802`.

[11] L. Fede and M. François. *Développement de schémas de calcul avec le code de réseau DRAGON5 pour la simulation d'assemblages de réacteurs de type VVER-1000 et validation avec le code stochastique Serpent2*. Master thesis. Sept. 2023.

[12] P. Fontaine and P. Panisi. *Contribution à la modélisation d'un mini cœur VVER dans l'environnement DRAGON5 et validation avec le code stochastique Serpent2*. Master thesis. Sept. 2024.

[13] A. Hébert G. Marleau and R. Roy. *A user guide for Version5*. Tech. rep. Institut de Génie Nucléaire, June 2024.

[14] L. Ghasabyan. *Validation of DRAGON5 lattice code for PWR (Pressurized Water Reactor) applications using depletion benchmarks by detailed comparison with SERPENT2 Monte Carlo code.* Master thesis. Dec. 2020.

[15] P. Groudev et al. *D3.3 – Definition report of SB LOCA + SG tubing break benchmark.* Tech. rep. Institute for Nuclear Research and Nuclear Energy, Aug. 2021. URL: `http://www.camivver-h2020.eu/src/assets/doc/D3-3.pdf`.

[16] O. Sevbo H. Artur. *D3.1 - A comprehensive review of the available VVER data for verification and validation of neutronics and thermal-hydraulics codes.* Tech. rep. Energorisk, Apr. 2021. URL: `http://www.camivver-h2020.eu/src/assets/doc/D3-1.pdf`.

[17] G. Marleau H. Prabha and A. Hébert. "Tracking algorithms for multi-hexagonal assemblies (2D and 3D)". In: *Annals of Nuclear Energy* 69 (2014), pp. 175–182. ISSN: 0306-4549. DOI: `https://doi.org/10.1016/j.anucene.2014.01.018`. URL: `https://www.sciencedirect.com/science/article/pii/S0306454914000267`.

[18] Moataz Harb, Dieter Leichtle, and Ulrich Fischer. "A Novel Algorithm for CAD to CSG Conversion in McCAD". In: *Journal of Nuclear Engineering* 4.2 (2023), pp. 436–447. ISSN: 2673-4362. DOI: `10.3390/jne4020031`. URL: `https://www.mdpi.com/2673-4362/4/2/31`.

[19] A. Hébert. *Applied Reactor Physics, second edition.* Presses Internationales Polytechniques, June 2016. ISBN: 978-2-553-01698-1.

[20] A. Hébert. *Les géométries du calcul de réseau.* May 2023.

[21] A. Hébert. "PyNjoy-2012: A system for producing cross-section libraries for the DRAGON lattice code". In: Sept. 2016. URL: `https://www.researchgate.net/publication/308566629`.

[22] Alain Hébert. "DRAGON5 and DONJON5, the contribution of École Polytechnique de Montréal to the SALOME platform". In: *Annals of Nuclear Energy* 87 (2016). Special Issue of The 3rd International Conference on Physics and Technology of Reactors and Application, pp. 12–20. ISSN: 0306-4549. DOI: `https://doi.org/10.1016/j.anucene.2015.02.033`. URL: `https://www.sciencedirect.com/science/article/pii/S0306454915001103`.

[23] R. N. Hwang. *Neutron Resonance Theory for Nuclear Reactor Applications: Modern Theory and Practices.* Tech. rep. Argonne National Laboratory, Sept. 2016. DOI: `10.2172/1351300`. URL: `https://www.osti.gov/biblio/1351300`.

[24] Jaakko Leppänen. *Serpent - User's manual.* Tech. rep. Valtion Teknillinen Tutkimuskeskus Technical Research Center of Finland, June 2015. DOI: `10.2172/1351300`. URL: `https://www.osti.gov/biblio/1351300`.

[25] T. Lötsch, V. Khalimonchuk, and A. Kuchin. "PROPOSAL OF A BENCHMARK FOR CORE BURNUP CALCULATIONS FOR A VVER-1000 REACTOR CORE". In: 2009. URL: `https://inis.iaea.org/collection/NCLCollectionStore/_Public/41/035/41035568.pdf`.

[26] L. Mercatali et al. "High-Fidelity Serpent2/SCF Solutions for Rod Ejection Scenarios in Support to the Verification of the CAMIVVER APOLLO3®/CATHARE3 Coupling Prototype". In: Apr. 2024. URL: `https://www.researchgate.net/publication/308566629`.

[27] V.L. Molchanov. "Nuclear fuel for VVER reactors. Current status and prospects". In: Sept. 2005.

[28] *OpenMOC : Theory and Methodology.* Tech. rep. Massachusetts Institute of Technology, 2019. URL: `https://mit-crpg.github.io/OpenMOC/methods/index.html`.

[29] M. Ouisloumen. "Résolution par la méthode des probabilités de collision de l'équation intégrale du transport à deux ou trois dimensions en géométrie hexagonale". PhD thesis. Ecole Polytechnique de Montréal, Aug. 1993. URL: `http://merlin.polymtl.ca/downloads/these_ouisloumen.pdf`.

[30] H. P. Raghav. "Computation of Neutron Fluxes in Fuel Pins Arranged in Hexagonal Lattices". PhD thesis. Ecole Polytechnique de Montréal, Aug. 2012. URL: `https://publications.polymtl.ca/785/1/2012_HemPrabhaRaghav.pdf`.

[31] P. Reuss. *Précis de neutronique*. EDP Sciences, 2003. ISBN: 2-86883-637-2.

[32] *Serpent Wiki: Validation and verification.*

[33] J.-F. Vidal, E. Garcia-Cervantes, and A. Willien. *D4.5 – Guideline for future VVER assembly calculation schemes*. Tech. rep. Comissariat à l'Energie Atomique and Electricité De France, Aug. 2023. URL: `http://www.camivver-h2020.eu/src/assets/doc/D3-1.pdf`.

[34] J.F. Vidal et al. "New Modelling of LWR Assemblies using the APOLLO2 Code Package". In: Apr. 2007.

[35] X. Warin. *NOTICE THEORIQUE DE LA METHODE DES CARACTERISTIQUES 2D ET DU GENERATEUR DE TRAJECTOIRES SALT*. Tech. rep. IGE–329. Institut de Génie Nucléaire, Mar. 2002.

[36] Zelong Zhao et al. "Validation and application of the Dragon5 lattice code for neutronics and burnup analysis of VVER-1000 pin cell and assembly model". In: *Nuclear Engineering and Design* 407 (2023), p. 112279. ISSN: 0029-5493. DOI: `https://doi.org/10.1016/j.nucengdes.2023.112279`. URL: `https://www.sciencedirect.com/science/article/pii/S0029549323001280`.