POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

TRACTEBEL
ENGIE

Monte Carlo-Based Transient Analysis of the NuScale Small Modular Reactor: Neutronics and Fuel Behavior Coupling

Tesi di Laurea Magistrale in
Nuclear Engineering - Ingegneria Nucleare

Author: **Giada Mazzoni**

Student ID: 222332
Advisor: Prof. Antonio Cammi
Co-advisor: Bart Vermeeren
Academic Year: 2024-25

Abstract

For nuclear safety, it is of the highest importance to be able to predict the exact power production in a nuclear reactor during a transient. Such a transient might occur during regular operation, but also in an accident scenario. It is of vital importance to calculate the temperature during such a transient accurately, in order to predict fuel damage and other consequences.

This thesis, developed in collaboration with Tractebel, aims to explore and test the transient simulation capabilities of the Serpent Monte Carlo code. The primary objective was to simulate accidental transients in the core of the NuScale reactor.

The current methods for transient analysis, both deterministic and hybrid, although generally fast, still rely on numerous approximations such as homogenization, diffusion theory, few-group cross sections, and factorization. These approximations make it difficult to predict the error in the final result, and their validity for each specific case must be determined separately, which is especially challenging for new and unique reactor designs. Given the critical importance of transient analysis for safety calculations, it is desirable to employ a higher-fidelity method. Therefore, the Monte Carlo method emerges as an attractive option for performing reliable and accurate transient analyses.

In this work, a multi-physics coupling was implemented between Serpent and an external thermodynamic code to simulate various transient scenarios. This coupling enabled the study of both the neutronic behavior and the thermal feedback, specifically focusing on Doppler effects within the reactor core. The results of these simulations demonstrated the potential of using Monte Carlo for transient analysis, highlighting both its advantages in accuracy and the challenges posed by computational requirements.

The computational cost of these simulations was notably high, limiting the feasibility of performing large-scale or frequent transient analyses. To address these challenges and improve the efficiency of such simulations, future developments could focus on integrating machine learning algorithms and reduction order methods. These methods could significantly accelerate the simulation process and optimize computational resources, making it more practical to perform accurate and reliable transient analyses in real-world applica-

tions.

Keywords: Monte Carlo, Neutronics, Fuel Feedback, Doppler Effect, Serpent, Point Kinetics, Transient analysis, Reactivity insertion.

Abstract in lingua italiana

Per la sicurezza nucleare, è di fondamentale importanza essere in grado di prevedere con precisione la produzione di potenza in un reattore nucleare durante un transitorio. Un tale transitorio potrebbe verificarsi durante il normale funzionamento, ma anche in uno scenario incidentale. È di vitale importanza calcolare accuratamente la temperatura durante un transitorio, al fine di prevedere i danni al combustibile e le altre conseguenze.

Questa tesi, sviluppata in collaborazione con Tractebel, ha l'obiettivo di esplorare e testare le capacità di simulazione in modalità transitoria del codice Monte Carlo Serpent. L'obiettivo principale consisteva nel simulare i transitori accidentali nel core del reattore NuScale.

I metodi attuali per l'analisi dei transitori, sia deterministici che ibridi, sebbene generalmente veloci, si basano comunque su numerose approssimazioni come la omogeneizzazione, la teoria della Diffusione, la teoria del Multigruppo e la fattorizzazione. Queste approssimazioni rendono difficile prevedere l'errore nel risultato finale, e la loro validità per ciascun caso specifico deve essere determinata separatamente, il che è particolarmente difficile per nuovi e unici progetti di reattori. Data l'importanza cruciale dell'analisi dei transitori per i calcoli di sicurezza, è desiderabile impiegare un metodo a maggiore affidabilità. Pertanto, il metodo Monte Carlo emerge come un'opzione interessante per eseguire analisi transitorie affidabili e accurate.

In questo lavoro, è stato implementato un accoppiamento multifisico tra Serpent e un codice termodinamico esterno per simulare vari scenari transitori. Questo accoppiamento ha permesso lo studio sia del comportamento neutronico che del feedback termico, con particolare attenzione agli effetti Doppler all'interno del core del reattore. I risultati di queste simulazioni hanno dimostrato il potenziale dell'uso del Monte Carlo per l'analisi dei transitori, evidenziando sia i suoi vantaggi in termini di accuratezza che le sfide legate ai requisiti computazionali.

Il costo computazionale di queste simulazioni è stato notevolmente elevato, limitando la possibilità di eseguire analisi transitorie su larga scala o frequenti. Per affrontare queste sfide e migliorare l'efficienza di tali simulazioni, gli sviluppi futuri potrebbero concentrarsi

sull'integrazione di algoritmi di machine learning e tecniche di riduzione d'ordine. Questi metodi potrebbero accelerare significativamente il processo di simulazione e ottimizzare le risorse computazionali, rendendo più pratico eseguire analisi transitorie accurate e affidabili in applicazioni reali.

Parole chiave: Monte Carlo, Neutronica, Feedback del Combustibile, Effetto Doppler, Serpent, Cinetica Puntuale, Analisi Transitoria, Inserzione di Reattività.

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**

1.1 Aim and methodology of the Thesis 1

1.2 Thesis structure 2

1.3 NuScale Small Modular Reactor design 4

1.4 Monte Carlo approach 8

1.4.1 Basics of the Monte Carlo method 9

1.4.2 Precision and accuracy of a sample average 11

1.5 The SERPENT code 12

2 Theoretical background **19**

2.1 Neutronics problem to solve 19

2.2 Considerations on the time-dependent behavior of the neutronic problem . 23

2.2.1 Steady-state problems 23

2.2.2 Transient problems 26

2.3 From the Boltzmann Transport Equation to the Diffusion Equation: Approximations and Simplifications 27

2.3.1 Fick's Law and the Diffusion Approximation 28

2.3.2 Energy Discretization and the Multi-Group Diffusion Equations . . 29

2.3.3 Spatial Approximations and the Helmholtz Equation 29

2.4 Point Reactor Kinetics Model 30

2.5 Using MC to solve the neutron transport 32

2.6 Using MC for transient neutronics calculations 34

2.7 Feedbacks to the neutronic problem: thermodynamic effects 37

3 Description of SERPENT model and Steady-state analysus 41
3.1 Input model and geometric configuration 41
3.2 Statistical error and convergence of criticality calculations 44
3.3 Preliminary Analysis 46
3.4 Extraction of Kinetic Parameters 50
3.5 Preparation for Transient Simulation 53

4 Transient Analysis 59
4.1 Cinecolo: Point Kinetics Equations 59
4.1.1 Verification of Cinecolo Using Numerical and Analytical Solution . 61
4.2 Transient Analysis with Serpent 67
4.2.1 Development of a Dynamic Simulation Mode in Serpent 2 Monte Carlo Code 67
4.2.2 Basic approach in the time-dependent simulations 69
4.2.3 Input for Transient Simulations 71
4.2.4 Population Control in Dynamic Method for Transient Analysis in Monte Carlo Simulations 74
4.3 Results 78

5 Fuel Thermal Feedback 85
5.1 Introduction to Neutronic-Thermal Coupling 85
5.2 The Serpent Code and the Multiphysics Interface 86
5.3 Implementation of Coupled Simulations: Steady-State and Transient Analysis 98
5.3.1 Steady-State Analysis in Coupled Mode 100
5.3.2 Coupled Transient Simulation: Methodology and Assumptions . . . 103
5.4 Development of a Numerical Code for Comparison 106
5.4.1 The Fuel Feedback Coefficient α_f 109
5.5 Results: Boron Diluition Transient 113
5.5.1 Sensitivity Analysis on Neutron Population 113
5.5.2 Sensitivity Analysis on Number of Time Steps and Transient Duration 117
5.5.3 Analysis of a 1ms Transient with High Temporal Resolution 123
5.5.4 Limitations of Serpent in the Transient Calculation of β_{eff} 125
5.6 Results: Rod Extraction Transient 129
5.6.1 Transient Comparison for Two Different Control Rod Configurations 133

6 Conclusions and Future Developments 137
6.1 Main Conclusions 137
6.2 Future Developments 138

Bibliography **141**

A SERPENT input file **145**

B SERPENT input file **149**

C SERPENT input file - PYTHON code **153**

List of Figures **165**

List of Tables **169**

List of Symbols **171**

Acknowledgements **173**

1 Introduction

Nuclear reactor technology has advanced a lot over the last 80 years. Although traditional reactor designs are based on well-validated conservative approaches, the industry is continually looking for improvements in safety and efficiency. This has driven the development of advanced multi-physics methodologies that reduce modeling approximations and improve accuracy, to allow a more detailed analysis of modern reactor cores and safety-related parameters.
The progress of reactor physics relies on high-performance computing (HPC). HPC can do complex calculations that were impossible before. In this context, Monte Carlo (MC) particle transport codes have attracted a lot of attention because of their accuracy and suitability for HPC implementation. Recent developments in state-of-the-art neutronic codes ([28]) have introduced multi-physics and transient simulation capabilities, expanding the potential of MC methods. One key application is the study of coupled transient scenarios in Light Water Reactors (LWRs), which can improve the evaluation of safety parameters and operational limits. [7]

1.1. Aim and methodology of the Thesis

The main goal of this study is to explore an alternative modeling strategy for the coupled neutronic and thermal problem, moving beyond the standard industry approach that relies on deterministic solvers and multi-step approximations. In conventional reactor simulations, the neutronic and thermal-hydraulic domains are treated through a sequence of homogenization and condensation processes. Although this method decreases the computational costs and is very well validated for operational analysis, it inherently introduces a loss of resolution and fidelity. Crucial spectral effects, fuel heterogeneity, and local power distributions at the pin level are often not accurately captured, in particular during transient conditions.

The aim of this work is to study the capability of Serpent 2 to analyse the behaviour of a nuclear system, in this case a fuel assembly in an infinite core, and its behavior during a transient scenario. The focus is on investigating the feasibility of an MC-based

approach to modelling the coupled behaviour of neutronics and fuel thermodynamics under reactivity insertion events. Specifically, this study examines two critical transient scenarios: boron dilution and rod ejection, in order to evaluate how the system responds to these perturbations.

The MC method allows for the direct use of pointwise cross sections and realistic 3D geometries eliminating the need for simplifications in the transport physics. Coupled with a detailed thermal feedback model at the fuel pin level, this approach aims to provide more accurate, physically consistent predictions of reactor behaviour, particularly under transient conditions.

The methodology adopted to achieve these goals is based on an initial steady-state simulation (criticality calculation) of the system. This enabled key physical parameters to be extracted, which are essential for the implementation of subsequent transient simulations. Having analysed the system under stable conditions, transient simulations were performed to evaluate the ability of Serpent 2 to handle time-dependent scenarios. The parameters obtained during the criticality calculations were then used to develop analytical and numerical models for comparison with the results generated by Serpent 2. Finally, coupled simulations integrating neutronics and fuel thermodynamics were implemented to assess the thermal feedback of the fuel under adiabatic conditions. Serpent's multi-physics coupling capabilities enabled this simulation mode, utilising a dedicated multi-physics interface provided within Serpent 2. Furthermore, key quantities such as the time evolution of assembly and rod-level power, system reactivity, and fuel temperature solutions were evaluated to provide a comprehensive assessment of the system's transient behaviour.

1.2. Thesis structure

Concerning the structure of this thesis, a flow chart has been included to illustrate the main phases of the project. It highlights the progression from the initial problem definition, through the tools and methods used, to the final results and conclusions. This flow chart provides a clear overview of the logical sequence of steps involved in the study, emphasizing key stages such as steady-state analysis, transient simulations, code verification, multi-physics coupling, and the integration of thermal feedback effects.

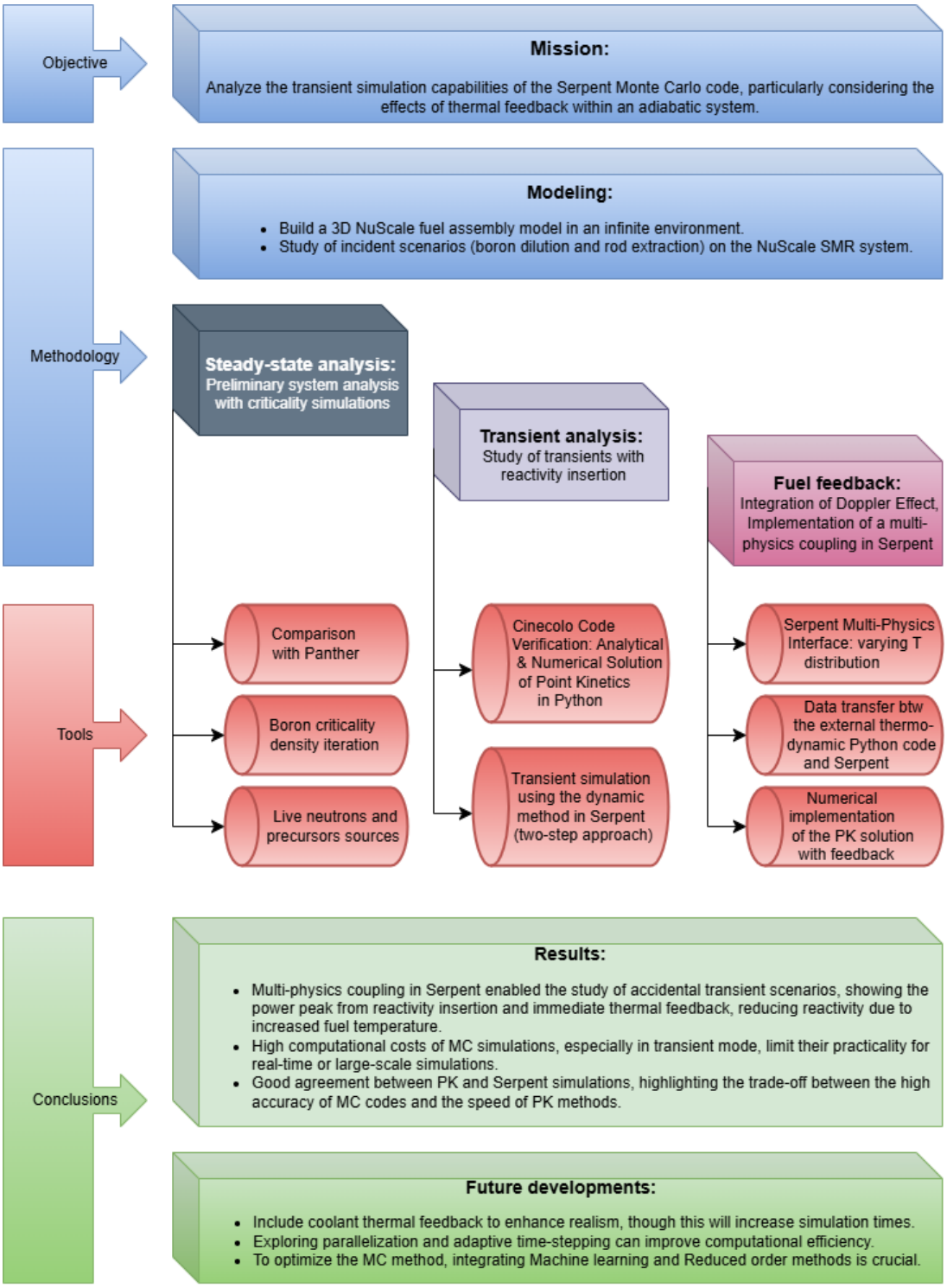


Figure 1.1: Thesis flow chart.

1.3. NuScale Small Modular Reactor design

The calculations were conducted using a Serpent model for the NuScale reactor core created at Tractebel by Agostin [1]. NuScale is an integrated PWR concept developed by NuScale Power LLC based in USA. A single module is rated at 160 MWt.

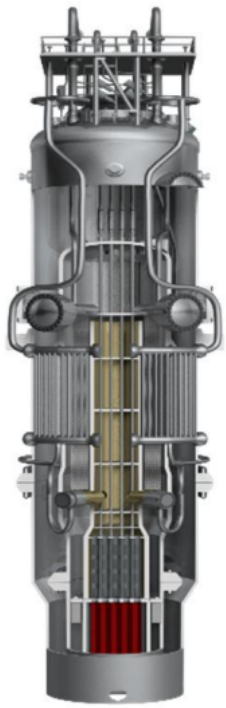


Figure 1.2: Schematic view of a NuScale reactor module [30].

The NuScale core consists of 37 fuel assemblies surrounded radially by a heavy steel reflector. The active height of the assemblies is 200 cm. The main thermal boundary conditions are listed in Table 1.1.

Parameter	Nominal Operation Value
Core power	160 MWth
System pressure (core outlet)	12.76 MPa
Core flow rate	587.15 kg/s (7.3% in bypass)
Core inlet temperature	531.45 K
Heavy metal loading	9367 kg
Power density	17.081 kW/kgU

Table 1.1: Thermal boundary conditions for the NuScale reactor core.

Fuel rods can be described as a combination of three different pin-cell geometries, the fuelled part, the plenum and the end cap. In addition to fuel rods, the fuel assemblies contain guide tubes (guide thimbles) into which control rods can be inserted. Control rods have several axial parts, the absorber AIC and B4C parts as well as a plenum part and end cap part. In this specification the fuel assemblies either have zero or 16 gadolinia containing rods. All fuel assemblies have 24 guide tubes into which control rods can be inserted. In addition, the assemblies contain a central instrumentation tube, which is modelled as an empty guide tube in this benchmark. The placement of burnable absorbers and guide tubes in fuel assemblies is shown in Figure 1.3. The fuel rod pitch inside the assembly is 1.2598 cm.

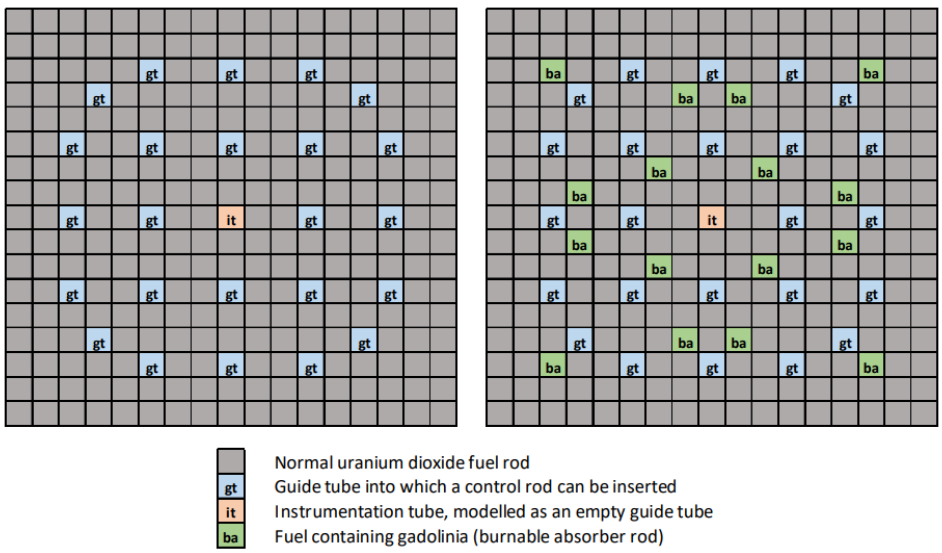


Figure 1.3: Placement of burnable absorbers and guide tubes in fuel assemblies [30].

The fuel assemblies have two different spacer grid types, the more numerous HTP spacer made of Zircaloy 4 and the bottommost HMP spacer grid made of Alloy 718 (Inconel). The bottom and top nozzle parts of fuel assemblies are modelled as homogeneous mixtures of steel and water. The loading pattern of the initial core is shown in the left part of Figure 1.4, while the right part of the figure illustrates the positioning of the control rod banks. The fuel assembly pitch inside the core is 21.5036 cm.

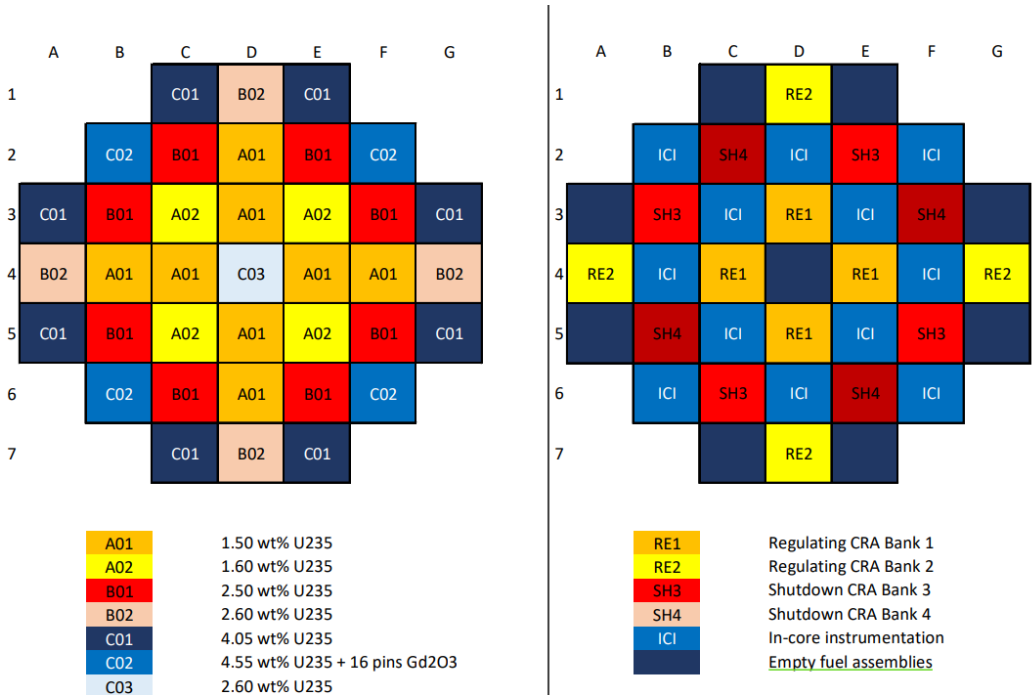


Figure 1.4: Loading pattern of the initial core (left) and positioning of the control rod banks (right) [30].

The control rod assemblies are divided into four banks, two of which are reserved for shutdown operation. The in-core instrumentation indicated in the figure is not modelled in this benchmark and the central instrumentation tubes in all assemblies are simply modelled as water-filled guide tubes.

The materials to be used for the different parts of the simulation model are collected in Table 1.2.

Part	Material
Guide tube	Zircaloy-4
Fuel pellets	Uranium dioxide (UO_2) or Gadolinium oxide (Gd_2O_3)
Fuel rod cladding	Zircaloy-4
Fuel rod plenum springs	Alloy 718
Fuel rod lower end cap	Zircaloy-4
Fuel rod upper end cap	Zircaloy-4
Control rod cladding	304 L SS
Control rod upper end plug	304 L SS
Control rod bottom plug	304 L SS
Core barrel	304 L SS
HMP™ grid	Alloy 718 (Inconel)
HTP™ grid	Zircaloy-4

Table 1.2: Materials used for different components of the NuScale core model.

In this analysis, a single fuel assembly of the entire reactor was considered, specifically assembly B01 at position F5 within the reactor core, along with its corresponding control rod bank, Shutdown CRA Bank 3, which was included in the simulation. This fuel assembly does not contain gadolinium, and its fuel characteristics are summarized in Table 1.3. [30]

Parameter	Value
Fuel Type	UO_2
Enrichment	2.5 wt% U-235
Gadolinium Content	None
Initial Burnup	0 MWd/tHM (Fresh Fuel)
Xenon Concentration	0 ppm (Clean Fuel)
Fuel Density	10.5216 g/cm^3
U-235	5.94084×10^{20} atoms/cm^3
U-238	2.28766×10^{22} atoms/cm^3
O-16	4.69414×10^{22} atoms/cm^3

Table 1.3: Fuel characteristics of assembly B01.

These modeling choices were made to simplify the system while maintaining key physical characteristics relevant to transient analysis. Simulating a single assembly instead of the entire core reduces computational costs and provides a more controlled environment for evaluating the response of the system to transient conditions. Additionally, the fuel was assumed to be fresh (i.e., without burnup) and clean (without xenon) to focus solely on the transient behavior, avoiding additional complexities such as reactivity effects due to fuel depletion or xenon oscillations. The exclusion of gadolinium was intended to prevent any additional spectral effects from burnable absorbers, ensuring that the transient behavior could be analyzed independently. While these simplifications help establish a baseline understanding of the transient response, future studies may incorporate additional effects such as burnup, xenon transients, and the presence of burnable absorbers to further refine the model.

1.4. Monte Carlo approach

The MC method is a statistical approach developed in the late 18th century that can be used to simulate mathematical and physical problems. The basic idea behind the MC technique is to estimate the expectation value of a given variable. The process is repeated, with each one being governed by a known or assumed probability density functions (PDFs), which make it a stochastic method. This is done using pseudo-random

numbers, which are now generated computationally, to model these processes. It is important to repeat simulations so that we can reduce statistical uncertainty and ensure a reliable estimation of the expected values.

1.4.1. Basics of the Monte Carlo method

In a MC simulation, the objective is to replicate a physical process where the governing physical laws are well-defined. Consequently, it is possible to define the corresponding PDFs, denoted here as $p(x)$, to describe these processes. Given the availability of random numbers, the main idea is to sample the outcome of a random process x based on a specified PDF $p(x)$, utilizing the cumulative probability density function (CDF), referred to here as $P(x)$.

To illustrate this concept, let's consider the case where we need to sample x from a known PDF $p(x)$, which is assumed to be properly normalized. The cumulative probability is then defined as:

$$P(a < x < b) = \int_a^b p(x)\, dx \tag{1.1}$$

As the cumulative probability $P(a < x < b)$ is in the range [0,1], the Monte Carlo method links this CDF $P(x)$ to a random variable that is distributed uniformly and then samples it accordingly. We can see this idea illustrated in Figure 1.5.

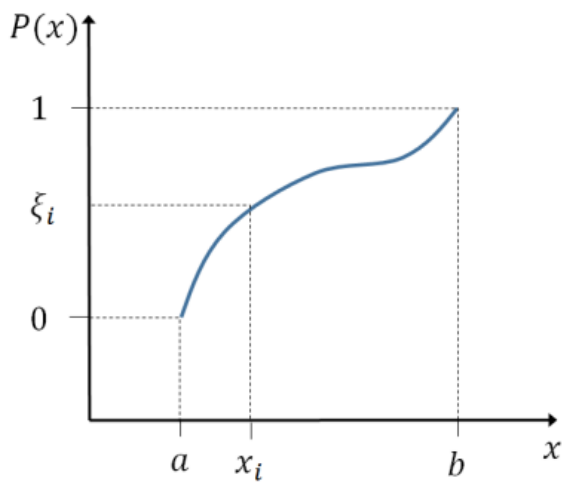


Figure 1.5: Sketch of the sampling of a random variable x.

This association can be demonstrated using the variable transformation rule [9], that leads to the equality:

$$P(a < x < b) = \xi \quad \Rightarrow \quad x = \xi^{-1}. \tag{1.2}$$

The process of selecting these random values ξ from a probability distribution is called sampling, and it depends on being able to generate random and uniformly distributed values for ξ while making sure they're properly represented in the PDF $p(x)$.

As a second step, the definitions of the expected value $E(x)$ and variance σ^2 can be introduced using the following expressions:

$$E(x) = \int x f(x) dx, \tag{1.3}$$

$$\sigma^2 = \int (x - E(x))^2 f(x) dx = E(x^2) - (E(x))^2. \tag{1.4}$$

These definitions work without need for specific constraints on the distribution of x, other than the conditions that $E(x)$ and σ^2 exist and are finite.

The last step involves using something called the law of large numbers [9], which, as the name suggests, states that the accuracy of an estimate of a quantity improves as the number of samples increases. So, if we let x_i be the sampled values of the random variable x, we can define the sample mean. Basically, the law of large numbers says that, as N approaches infinity, the sample mean approximates the true mean $E(x)$:

$$E(x) \approx \bar{X}_N = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{1.5}$$

Furthermore, the square root of the variance σ represents the standard deviation of the population. In practical calculations, it's important to remember that the true mean is unknown and that the variance σ must be estimated to apply the central limit theorem. Given N samples of the random variable x, the sample variance $\bar{\sigma}^2(x)$ is estimated like this:

$$\sigma^2(x) \approx \bar{\sigma}^2(\bar{X}_N) = \frac{1}{N-1} \left(\frac{1}{N} \sum_{i=1}^{N} x_i^2 - \bar{X}_N^2 \right). \tag{1.6}$$

When doing MC simulations, it is a common practice to report the standard deviation $\bar{\sigma}(\bar{X}_N)$ in terms of the relative statistical error $R(\bar{X}_N)$. This measure is useful because it

expresses the statistical uncertainty as a fraction of the estimated mean, providing a good indicator of precision. This parameter, which is also used in this study, is defined as:

$$R(\bar{X}_N) = \frac{\bar{\sigma}(\bar{X}_N)}{\bar{X}_N}. \tag{1.7}$$

Both $R(\bar{X}_N)$ and $\bar{\sigma}(\bar{X}_N)$ are inversely proportional to $\sqrt{N}$, which is a fundamental limitation of the Monte Carlo method. Infact, reducing the standard deviation $\bar{\sigma}(\bar{X}_N)$ by half requires increasing the number of sampled histories by a factor of four. This puts in evidence that to achieve a well-converged MC simulation can become a computationally expensive task quickly. [7]

1.4.2. Precision and accuracy of a sample average

To determine the confidence level on the outcome of a statistical process, we need to determine the precision and accuracy associated with the sample average. In this sense, it is crucial to understand that there is an important difference between the two concepts within MC calculations. We can see these two effects in Figure 1.6.

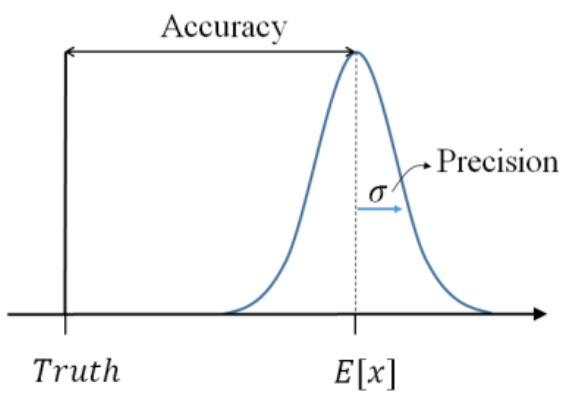


Figure 1.6: Sketch of the difference between accuracy vs. precision in MC neutronics.

Precision is measured by the relative statistical uncertainty, while accuracy refers to the deviation from the true value, which means how close the precise average is to the actual mean. Estimating accuracy typically requires experimental validation or a comparison with a well-established method.

If the physical description of the system is incorrect (e.g., improper modeling of materials or geometry), accuracy remains compromised regardless of the number of histories N used. MC simulations can yield highly precise but inaccurate results if the model does not accurately represent reality. Conversely, a low N leads to a lack of precision. [9]

In this study, precision can be improved by reducing statistical uncertainty, while accuracy is inherently limited due to the absence of experimental data, as the NuScale SMR is still a conceptual design. However, comparing MC results with numerical and analytical models can help refine accuracy.

To improve convergence, two main strategies can be adopted: increasing the number of histories N, or enhancing the contribution of each history to the tally of interest, even though this may increase the computational cost. The latter approach is often essential in MC codes when dealing with complex or neutronically large problems, and is commonly referred to as Variance Reduction.

In this context, an important efficiency metric, known as the Figure of Merit (FOM), is defined as: [7]

$$FOM = \frac{1}{R^2 T} \tag{1.8}$$

where T represents the computational time, and R is the relative error. The FOM serves as an indicator of the efficiency of an MC simulation, with a higher FOM value signifying that less computational time is required to achieve a specific level of statistical uncertainty R.

1.5. The SERPENT code

Serpent is a multi-purpose three-dimensional continuous-energy neutron and photon transport code, developed at VTT Technical Research Centre of Finland since 2004. The code is distributed with different license options for non-commercial research and educational use, and for commercial work [14]. Its main uses concern reactor neutronics, burnup calculations, Group constant generation, sensitivity analysis and multi-physics simulations. The version used for this project is Serpent 2.1.32, and some of its key features are described below:

- Like other MC codes, such as MCNP, the creation of the geometrical model is based on the definition and intersection of simple surfaces, including cylinders, planes, and squares. These elements form cells, where different materials are assigned.
- Serpent provides various boundary conditions at the domain limits, including reflective, black (absorbing), and periodic boundaries, offering flexibility in reactor simulations.
- The code utilizes continuous-energy cross-section data in ACE format, supporting

libraries such as JEFF and ENDF for nuclear data.

- Massive parallelization of the code is allowed, where the possibility to manage a hybrid compilation considering OMP (Open Multi-Processing, a shared memory multiprocessing programming interface) plus MPI (Message Passing Interface, a communications protocol for parallel architectures without shared memory) makes it specially suited for its use in HPC architectures, commonly composed by several calculation nodes (each of them with multiple processors).

- Serpent handles particle transport using a hybrid approach that combines conventional surface-tracking with the Woodcock delta-tracking method. The latter is based on the use of virtual collisions, where the neutron continues its path without undergoing a physical interaction. To enable this, a so-called majorant cross section is defined system-wide as the upper bound of the total cross sections across all materials. This allows the sampling of collision sites independent of material boundaries, significantly simplifying geometry handling and improving computational efficiency, particularly in complex geometries. However, delta-tracking may lead to undersampling in small regions with high absorption cross sections, resulting in poor statistical accuracy. To address this, Serpent employs an adaptive combination of both tracking methods, optimizing the trade-off between performance and precision.

- A notable feature of Serpent is its built-in Doppler broadening routine, which enables transport simulations at arbitrary material temperatures. This routine dynamically corrects cross-section data by computing the resonance broadening effect, improving the accuracy of temperature-dependent neutron transport calculations.

- Another feature in Serpent is Critical Density Iteration, designed primarily for determining critical boron concentration in LWRs. This option adjusts the atomic density of selected nuclides using a scaling factor to achieve the target k_{eff}.

  During each neutron cycle, Serpent solves the neutron transport equation while tracking neutron absorption by the flagged nuclides. The neutron balance equation is then updated, separating losses due to these nuclides from other absorption and production terms. The goal is to adjust the nuclide concentration so that the system reaches the desired k_{eff}, typically 1.

- An additional capability of Serpent is its ability to efficiently handle multi-physics coupling, enabling the integration of neutronics, thermal hydraulics, and fuel behavior within reactor simulations. One of the key challenges in coupling MC neutronics

with multi-physics solvers is managing the two-way data exchange between them. Serpent addresses this by completely decoupling the thermal-hydraulic state-point distributions from the user-defined geometry. This is achieved through a universal multi-physics interface, which overlays a three-dimensional structure onto the existing geometry, allowing the seamless integration of material temperatures and densities into the transport simulation.

- Standard detectors (tallies) in Serpent allow the calculation of neutron flux, power, and reaction rate distributions within geometry cells and materials, as well as in structured elements such as lattices and superimposed meshes.

  When analyzing nuclear reactors, reaction rates (RR) are key quantities of interest. These can be expressed as integral parameters using the neutron scalar flux, defined as:

$$\phi(\mathbf{r}, E, t) = \int_{4\pi} \varphi(\mathbf{r}, \mathbf{\Omega}, E, t) d\mathbf{\Omega}, \tag{1.9}$$

  which leads to the general reaction rate expression:

$$RR\,(1/s) = \int_V \int_E \phi(\mathbf{r}, E)\Sigma(\mathbf{r}, E)\, dE\, d^3r. \tag{1.10}$$

  These reaction rates represent key scalar fields, such as power density or material activation rates, when multiplied by appropriate conversion factors. For example, using the energy released per fission allows one to compute the thermal power corresponding to a given neutron flux.

  Serpent employs the collision estimate of neutron flux to compute user-defined reaction rates, integrating over spatial and energy domains:

$$R = \frac{1}{V} \int_V \int_{E_i}^{E_{i+1}} f(\mathbf{r}, E)\phi(\mathbf{r}, E)\, d^3r\, dE. \tag{1.11}$$

  Here, the response function $f(\mathbf{r}, E)$ and the integration domains are defined by detector parameters. By default, the integral is normalized by the detector volume, which is set to unity, as users typically seek the total reaction rate rather than its density.

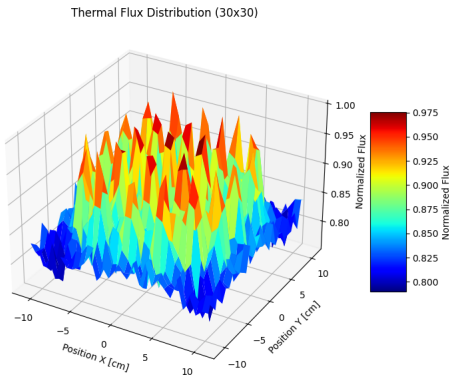
  Serpent provides various detector parameters to define the reaction rates, such as the reaction multiplier (`dr`), the detector volume used for normalization (`dv`), the

cell where reactions are recorded (`dc`), and the energy grid for binning the responses (`de`). Additionally, mesh-based scoring can be defined along Cartesian coordinates (`dx`, `dy`, `dz`).

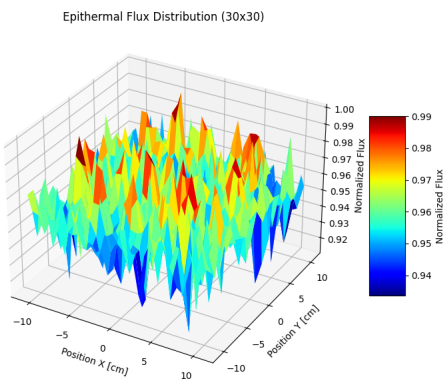
The detector response function determines the type of calculation. In the simplest case, setting $f = 1$ reduces Eq. 1.11 to the neutron flux integrated over space and energy. If a reaction cross-section is used, the result corresponds to the respective reaction rate.

Serpent allows users to specify the type of reaction rate to be computed. For example, -1 corresponds to the total reaction rate, -2 to the total capture rate, -6 to the total fission rate, and -8 to the total fission energy deposition. Similarly, specific ENDF reaction modes can be defined, such as 18 for total fission, 102 for radiative capture (n, γ), and 107 for neutron-induced alpha production (n, α). For instance, setting $mt = -8$ yields the total energy deposited in the system (in joules). The absolute value of the computed integral depends on source normalization. [17]

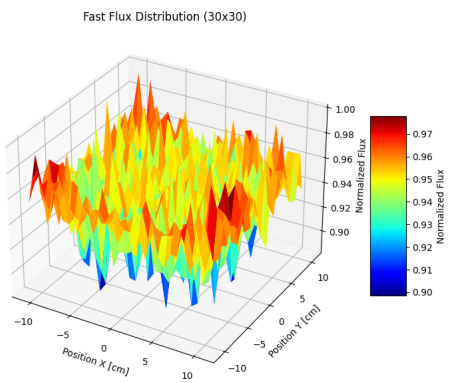
As example Figure 1.7 shows neutron flux in a 2D mesh and divided in 3 energy bins. The thermal flux shows higher values in the fuel rods at the center of the assembly, as the guide tubes located there are filled with water, which moderates the neutrons. The fast flux is higher at the periphery of the assembly due to the higher concentration of fuel rods in that region, considering an infinite environment. The epithermal flux presents an intermediate distribution between the two.



(a) Thermal flux



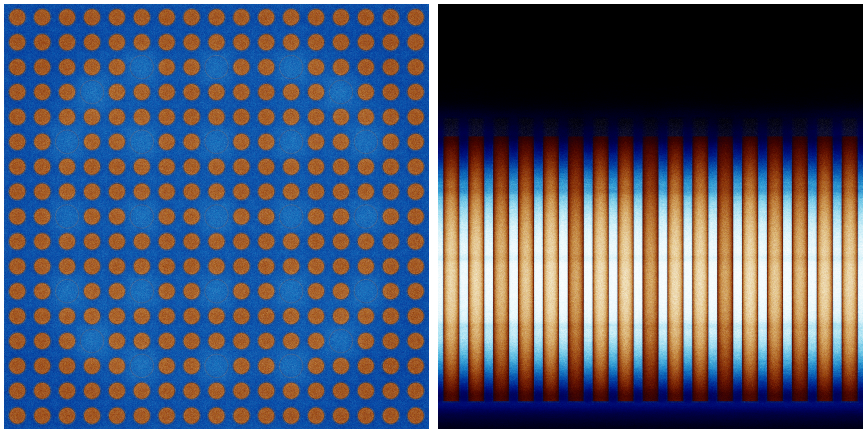
(b) Epithermal flux



(c) Fast flux

Figure 1.7: Thermal, Epithermal and Fast neutron fluxes inside the fuel assembly.

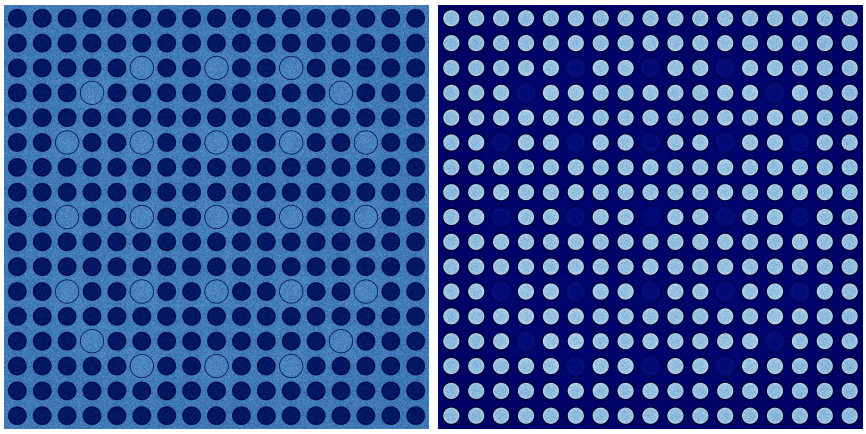
Figure 1.8 shows some 2D meshes representing the colormap of the assembly. Warm colors correspond to the fission rate (brighter areas indicate a higher fission rate), while cool colors represent the thermal flux (brighter areas indicate a higher concentration of thermalized neutrons). In the mesh 1.8a, the fuel rods appear in warm colors because they are the regions where power is generated, while the surrounding water appears in cool colors, as it contains thermalized neutrons. In the mesh 1.8b, the central region filled with water appears white, indicating a high concentration of thermal neutrons. The black areas correspond to regions without fuel and containing control rods, which absorb neutrons, preventing Serpent from plotting any flux values in those regions.



(a) XY mesh (b) XZ mesh

Figure 1.8: Fission rate and thermal flux colormaps.

Finally, Figure 1.9 shows some 2D meshes representing the colormap of the assembly. The meshes represent cold colormaps of the elastic scattering rate 1.9a and the capture rate 1.9b. In the mesh 1.9a, the fuel rods appear darker due to a lower scattering rate, while the water appears brighter as it has a higher scattering rate. In the mesh 1.9b, the water appears darker due to a lower capture rate, whereas the fuel rods appear brighter because of an higher capture rate. The left mesh is less defined because it was generated using a simulation with 10000 neutrons, whereas the right mesh is more detailed, as it was obtained from a simulation with 100000 neutrons.



(a) Elastic scattering rate (b) Capture rate

Figure 1.9: Thermal, Epithermal and Fast neutron fluxes inside the fuel assembly.

2 | Theoretical background

Abstract

A concise overview of the fundamental physical principles used to describe neutron behavior in nuclear reactors is presented. The first section introduces the theoretical foundations of the problem, focusing on reactor physics concepts essential for understanding neutron transport and interactions. The second section is dedicated to the specific aspects related to the use of Monte Carlo codes, detailing their application in neutronic analysis and simulations.

2.1. Neutronics problem to solve

To start, we introduce the most commonly used quantities in the description of the neutronic problem. A neutron is characterized not only by its position $\mathbf{r}$, but also by its velocity vector $\mathbf{v}$. The direction of motion can be described by the unit vector $\mathbf{\Omega} = \mathbf{v}/|\mathbf{v}|$ which is schematically illustrated in Figure 2.1. The fundamental quantities of interest in

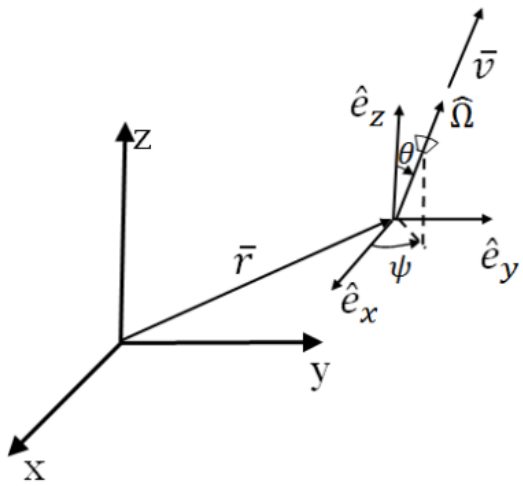


Figure 2.1: Position and direction variables to characterize the neutrons distribution [6].

neutronic analysis are now defined, following the convention from [6]:

Main Neutronic Quantities

Quantity	Formula	Unit / Description
Angular neutron density	$n(\mathbf{r}, \mathbf{\Omega}, E, t)$	Neutron count per unit volume, energy, and solid angle at time t
Neutron speed	$v = \lvert\mathbf{v}\rvert$	cm/s
Neutron angular flux	$\varphi(\mathbf{r}, \mathbf{\Omega}, E, t) = vn(\mathbf{r}, \mathbf{\Omega}, E, t)$	Neutron flow rate per unit area, energy, and direction
Neutron scalar flux	$\phi(\mathbf{r}, E, t) = \int_{4\pi} \varphi(\mathbf{r}, \mathbf{\Omega}, E, t) d\mathbf{\Omega}$	Total neutron flux over all directions
Neutron current density	$\mathbf{J}(\mathbf{r}, E, t) = \int_{4\pi} \mathbf{\Omega}\varphi(\mathbf{r}, \mathbf{\Omega}, E, t) d\mathbf{\Omega}$	Net neutron flow per unit area and energy
Microscopic cross section	$\sigma_j(\mathbf{r}, E)$	barns (10^{-24} cm^2), probability of neutron interaction
Macroscopic cross section	$\Sigma_j(\mathbf{r}, E) = N\sigma_j(\mathbf{r}, E)$	(1/cm), interaction rate per unit path length, where N is the atomic density (1/cm^3)

Table 2.1: Main neutronic quantities and their corresponding formulas and units, including neutron current density.

The prediction of neutron distribution within a nuclear reactor is a fundamental aspect of reactor physics, as it determines the ability to sustain a controlled and stable chain reaction while defining the power generation profile. The governing equations describing neutron behavior are derived by considering a balance between neutron production and losses, accounting for the intrinsic properties of neutron transport. A detailed derivation can be found in literature [6], leading to the formulation of the neutron transport equation.

This equation expresses the angular neutron flux $\varphi(\mathbf{r}, \mathbf{\Omega}, E, t)$ and is subject to specific boundary and initial conditions:

$$\frac{1}{v}\frac{\partial\varphi}{\partial t} + \mathbf{\Omega}\cdot\nabla\varphi + \Sigma_t(\mathbf{r},E)\varphi(\mathbf{r},\mathbf{\Omega},E,t) = \int_{4\pi} d\mathbf{\Omega}' \int_0^\infty dE'\,\Sigma_s(E'\to E)\varphi(\mathbf{r},\mathbf{\Omega}',E',t) + S(\mathbf{r},\mathbf{\Omega},E,t). \tag{2.1}$$

with the initial condition:

$$\varphi(\mathbf{r},\mathbf{\Omega},E,0) = \varphi_0(\mathbf{r},\mathbf{\Omega},E), \tag{2.2}$$

and the boundary condition for non-re-entrant surfaces:

$$\varphi(\mathbf{r}_s,\mathbf{\Omega},E,t) = 0, \quad \text{if } \mathbf{\Omega}\cdot\mathbf{e}_s < 0. \tag{2.3}$$

In these equations, $\varphi(\mathbf{r},\mathbf{\Omega},E,t)$ represents the neutron angular flux, which depends on position $\mathbf{r}$, direction $\mathbf{\Omega}$, energy E, and time t. The term $S(\mathbf{r},\mathbf{\Omega},E,t)$ accounts for neutron sources appearing within a differential volume and solid angle at a given energy. The left-hand terms in Eq. (2.1) represent temporal changes and neutron losses due to leakage and collisions, while the right-hand side accounts for scattering contributions and external sources. This equation embodies a fundamental balance between neutrons that escape and those that appear within the system, ensuring conservation. Moreover, the presence of the time derivative highlights how the neutron population evolves dynamically, capturing transient effects in the system. Although the macroscopic cross sections Σ are written as functions of position and energy, they may also vary with time due to changes in fuel composition or geometrical configurations during reactor operation.

For a nuclear reactor, the source term $S(\mathbf{r},\mathbf{\Omega},E,t)$ can be expressed as the sum of an external fixed source and a fission source. However, in the majority of practical reactor core calculations, only the fission source is considered. The fission process is then described in terms of the fission spectrum $\chi(E)$, the fission cross-section $\Sigma_f(E,\mathbf{r})$, and the average number of neutrons emitted per fission event, $\nu(E)$ [6]. In Eq. (2.1), the neutrons produced by fission should be categorized based on their emission characteristics. The majority are released almost instantaneously, within approximately 10^{-14} seconds, and are referred to as prompt neutrons. However, a small fraction, denoted by β, is emitted with a significant time delay and is classified as delayed neutrons. These delayed neutrons constitute less than 1% of the total, with specific values of approximately 0.65% for ^{235}U and 0.22% for

^{239}Pu when fissioned by thermal neutrons. The delayed neutron component originates from the decay of unstable fission fragments, with emission times ranging from roughly 0.2 to 55 seconds. Despite their small fraction, delayed neutrons play a crucial role in reactor physics and control. This difference in emission time for delayed neutrons enables the practical control of the overall reaction in an engineered system. Specifically, it allows for regulation through mechanical adjustments, such as the movement of control rods, or through modifications in reactor composition, such as varying the boron concentration in the coolant [7].

To simplify the modeling of delayed neutrons, they are classified into j decay groups, each characterized by a concentration of precursor nuclei, C_i, an associated decay constant, λ_i, and a corresponding delayed neutron fraction, β_i. This classification introduces a new set of coupled differential equations to be solved.

By reconsidering the neutron transport equation (Eq. (2.1)) and assuming that fission is an isotropic process [4, 19], we obtain Eq. (2.4):

$$\frac{1}{v}\frac{\partial\varphi}{\partial t} + \mathbf{\Omega}\cdot\nabla\varphi + \Sigma_t(\mathbf{r},E)\varphi(\mathbf{r},E,\mathbf{\Omega},t) = \int_{4\pi} d\mathbf{\Omega}' \int_0^\infty dE'\,\Sigma_s(E'\to E)\varphi(\mathbf{r},\mathbf{\Omega}',E',t) \tag{2.4}$$

$$+\frac{\chi(\mathbf{r},E,t)}{4\pi}\int_0^\infty\int_{4\pi} d\mathbf{\Omega}'dE'(1-\beta(\mathbf{r},E,t))\nu(E')\Sigma_f(E',\mathbf{r})\varphi(\mathbf{r},E',\mathbf{\Omega}',t)$$

$$+\frac{1}{4\pi}\sum_{i=1}^{j}\chi_i(E)\lambda_i C_i(\mathbf{r},t),$$

with the initial conditions:

$$\varphi(\mathbf{r},\mathbf{\Omega}',E',0) = \varphi_0(\mathbf{r},\mathbf{\Omega}',E'), \tag{2.5}$$

and the boundary conditions:

$$\varphi(\mathbf{r}_s,\mathbf{\Omega}',E',t) = 0, \quad \text{if } \mathbf{\Omega}\cdot\hat{\mathbf{e}}_s < 0. \tag{2.6}$$

The evolution equation for the precursor concentration is given by:

$$\frac{\partial C_i(\mathbf{r},t)}{\partial t} = \int_{4\pi}\int_0^\infty dE' d\mathbf{\Omega}\, \beta_i(\mathbf{r},E',t)\nu(E')\Sigma_f(E',\mathbf{r})\varphi(\mathbf{r},E',\mathbf{\Omega},t) - \lambda_i C_i(\mathbf{r},t), \tag{2.7}$$

for $i = 1, 2, \ldots, j$, with the initial conditions:

$$C_i(\mathbf{r},0) = C_{0i}(\mathbf{r}). \tag{2.8}$$

2.2. Considerations on the time-dependent behavior of the neutronic problem

From a neutronic perspective, time dependence is addressed using different approaches based on the timescales in which the reactor undergoes significant changes. In this context, reactor physics core calculations typically involve two types of problems, each with its own specific terminology:

- Cases where Eq. 2.1 is solved without considering time dependence, referred to as *steady-state* calculations.
- Cases where Eq. 2.1 is solved while accounting for time dependence, including the modeling of delayed neutrons (as shown in Eqs. 2.4 and 2.7), referred to as *transient* or *kinetic* calculations.

Each of these approaches presents different levels of modeling complexity, and their key aspects are discussed here in relation to the objectives of this study. It is important to highlight that transient calculations in a nuclear reactor must be preceded by an initial steady-state calculation.

2.2.1. Steady-state problems

The spatial distribution of neutrons within a reactor core, along with its safety characteristics related to startup, operation, and shutdown, is determined by solving a static problem, where the temporal dependence is neglected. This requires modifying the balance between neutron production and losses. One common approach [6, 19] is to introduce a multiplication factor k, transforming the equation into an eigenvalue problem.

By taking Eq. 2.1 and modifying the production term by dividing it by k, while neglecting external sources, we obtain:

$$\mathbf{\Omega} \cdot \nabla\varphi + \Sigma_t(\mathbf{r}, E)\varphi(\mathbf{\Omega}, \mathbf{r}, E) = \int_{4\pi}\int_0^\infty d\mathbf{\Omega}' dE'\, \Sigma_s(E' \to E, \mathbf{\Omega}' \cdot \mathbf{\Omega})\varphi(\mathbf{\Omega}', \mathbf{r}, E') \tag{2.9}$$

$$+\frac{1}{k}\chi(E)\int_0^\infty dE'\, \nu(E')\Sigma_f(E', \mathbf{r})\int_{4\pi} d\mathbf{\Omega}'\varphi(\mathbf{\Omega}', \mathbf{r}, E'),$$

with the boundary condition:

$$\varphi(\mathbf{r}_s) = \varphi_{r_s}. \tag{2.10}$$

The multiplication factor k is often expressed using the *six-factor formula*:

$$k = \eta f \rho p P_{\mathrm{NL}}^F P_{\mathrm{NL}}^T \tag{2.11}$$

where:

$$\begin{cases} \eta = \dfrac{\text{neutrons produced from fission}}{\text{absorption in fuel}}, \\ f = \dfrac{\text{absorption in fuel}}{\text{total absorption}}, \\ \rho = \dfrac{\text{total number of fission neutrons}}{\text{neutrons from thermal fission}}, \\ p = \dfrac{\text{neutrons slowed to thermal energies without absorption}}{\text{total fission neutrons}}, \\ P_{\mathrm{NL}}^F = \dfrac{\text{fast neutrons that do not leak from the reactor}}{\text{fast neutrons produced by fission}}, \\ P_{\mathrm{NL}}^T = \dfrac{\text{thermal neutrons that do not leak from the reactor}}{\text{neutrons slowed to thermal energies}}. \end{cases} \tag{2.12}$$

Each of these factors depends on reactor conditions, particularly temperature, which affects neutron interactions through density and cross-section variations. The cross-section is influenced by the relative velocity between neutrons and nuclei; while high-energy neutrons can be approximated as colliding with stationary nuclei, thermal neutrons require consideration of nuclear motion due to thermal agitation. Since material temperature impacts nuclear motion, it also affects the cross-section, ultimately influencing k and reactor dynamics. In Table 2.2 are reported, as reference, the values for the six-factor formula parameters obtained with Serpent for the entire NuScale SMR reactor model simulated by Agostin ([1]). These values refer to hot conditions, meaning the reactor is at its

nominal operating temperature and pressure. Additionally, they correspond to an ARO (All Rods Out) configuration, where all control rods are fully withdrawn, representing an unperturbed core state.

Six-Factor Formula Parameters for NuScale SMR (Hot Conditions)

Parameter	Value
η	1.73
f	0.72
ε	1.26
p	0.65
P_{NL}^{F}	0.99
P_{NL}^{T}	0.99

Table 2.2: Indicative values (taken by Agostin's work [1]) for the six-factor formula parameters obtained with Serpent for NuScale SMR under hot conditions. The associated uncertainty ($\pm 2\sigma$) is on the order of 10^{-4}.

As can be seen, the value of ϵ in the six-factor formula is quite high, even though the reactor is thermal and not fast. This result is unexpected, because high epsilon values are usually found in fast reactors. One possible explanation is that the reactor I studied is a Small Modular Reactor (SMR), which means it is much smaller than a standard PWR. The mean free path of neutrons in a reactor core ranges tipically between 1 and 3 cm in thermal reactor. In smaller reactor, the reduced dimensions lead to a higher surface-to-volume ratio. Consequently, neutrons are more likely to reach the reactor boundaries before undergoing interactions, resulting in increased neutron leakage. As a result, some neutrons leave the core before being fully moderated. This reduces the thermalization process and increases the number of fast neutrons in the flux. Therefore, the thermal flux is lower, and the fast flux becomes relatively more important. When calculating epsilon, the fissions caused by fast neutrons contribute more to the total, which could explain the higher value.

Reactivity ρ is a key parameter in defining the behavior of a reactor and is expressed as:

$$\rho = \frac{k-1}{k}. \tag{2.13}$$

Due to its typically low values, ρ is commonly measured in percent-mille (pcm) or in dollars ($), the latter obtained by normalizing with the effective delayed neutron fraction β:

$$\rho[\$] = \frac{k-1}{k\beta}. \tag{2.14}$$

In reactor analysis, the total thermal power is derived from the energy released by fission, primarily as kinetic energy of fission fragments, with additional contributions from other particles [6]. Disregarding the detailed components of energy release, the thermal power acts as a normalization factor:

$$P[W] = \int_V \int_0^{E_{\max}} dE' E_{\text{fiss}}(E')\Sigma_f(E', \mathbf{r})\phi(\mathbf{r}, E)d^3r, \tag{2.15}$$

where $E_{\text{fiss}}(E)$ represents the energy released per fission.

2.2.2. Transient problems

The analysis of transient problems involves several key assumptions and challenges:

- **Constant composition**: During transient analysis, the material composition (e.g., fuel and moderator) is assumed to remain unchanged over time. This means that slow phenomena such as fuel burnup or long-lived fission product accumulation are not considered, as a consequence in our equation all of the material properties, such as cross sections, are independent of time.
- **Delayed neutron modeling**: The effect of delayed neutrons, which play a crucial role in reactor control, is accounted for by grouping precursor nuclides into decay families. This approach simplifies the complex decay dynamics, as represented in Eq. 2.4.
- **Increased computational complexity**: Since the solution of transient problems provides the neutron distribution evolution in both time and space, the system becomes significantly more complex than steady-state problems, which assume a static neutron balance without temporal variations.

In summary, transient analysis introduces a time-dependent dimension to neutron transport calculations, requiring more sophisticated numerical methods and computational resources.

2.3. From the Boltzmann Transport Equation to the Diffusion Equation: Approximations and Simplifications

The neutron transport equation (Eq. 2.1) is a high-dimensional integro-differential equation that depends on position, direction, energy, and time, making its direct resolution impractical for complex reactor geometries. To obtain a computationally feasible model while preserving essential physical behavior, several approximations are introduced, leading to the **diffusion equation**. This section summarizes the key steps and simplifications involved in this process.

A widely used method in the nuclear industry is the **cell-core approach** [6], which divides the problem into two stages:

- **First step – Lattice calculations:** The neutron physics at the pin or fuel assembly level is modeled in detail, with approximations in angular dependence and energy discretization. The result is a set of homogenized parameters that will be used for core-level calculations.
- **Second step – Core calculations:** The entire reactor core is modeled using the homogenized parameters from the first step. The transport equation is further simplified by integrating over angles and applying Fick's law, leading to the diffusion equation.

The first step aims to accurately model neutron interactions at the fuel assembly level. The following approximations are introduced:

- **Angular approximation:** The complex angular dependence is simplified using methods such as Discrete Ordinates (S_n), Spherical Harmonics (P_n), Collision Probabilities (CP), or Heterogeneous Response Methods (HRM) [6].
- **Energy discretization:** Instead of considering a continuous energy spectrum, cross-sections are computed in **energy groups** (typically tens to hundreds). A pre-tabulated Nuclear Data Library (NDL) is used for each energy group.
- **Homogenization:** The heterogeneous structure of fuel assemblies, which consists of different materials such as fuel, moderator, and coolant, is simplified by replacing it with a **homogenized macroscopic cross-section** and **diffusion coefficients** that represent an equivalent medium. This step is crucial as it averages the material properties over the entire fuel assembly, making the problem more manageable.

The process involves **flux-weighted averaging**, where the neutron flux distribution is used to calculate effective cross-sections, such as absorption and scattering rates. While homogenization simplifies reactor modeling, it introduces approximations by losing fine spatial details of the fuel assembly. Despite this, it is essential for computational feasibility, enabling efficient large-scale simulations. However, it's important to choose appropriate methods to minimize errors and perform sensitivity studies, particularly for new reactor designs with unique geometries.

- **Condensation:** The multi-group energy structure is collapsed into fewer energy groups (typically 2–5), preserving reaction rates.

2.3.1. Fick's Law and the Diffusion Approximation

A key step in simplifying the transport equation is the application of **Fick's Law**, which provides a phenomenological relation between the neutron current and the flux gradient. This approximation assumes that the neutron current density is proportional to the negative gradient of the neutron flux:

$$J(\mathbf{r}, E, t) = -D(\mathbf{r}, E)\nabla\phi(\mathbf{r}, E, t), \tag{2.16}$$

where $D(\mathbf{r}, E)$ is the **diffusion coefficient**, given by:

$$D(\mathbf{r}, E) = \frac{1}{3\Sigma_{\text{tr}}(\mathbf{r}, E)}, \tag{2.17}$$

with the **transport cross section** defined as:

$$\Sigma_{\text{tr}}(E) = \Sigma_t(E) - \bar{\mu}\Sigma_s(E), \tag{2.18}$$

where $\bar{\mu}$ is the anisotropy factor. Σ_{tr} accounts for the effect of directional scattering on neutron trajectories. Substituting Fick's law into the transport equation and integrating over all angles, we obtain the **diffusion equation**:

$$\frac{1}{v}\frac{\partial\phi}{\partial t} - \nabla \cdot D(\mathbf{r}, E)\nabla\phi + \Sigma_t(\mathbf{r}, E)\phi(\mathbf{r}, E, t) = \int_0^\infty dE'\Sigma_s(E' \to E)\phi(\mathbf{r}, E', t) + S_{\text{ext}}(\mathbf{r}, E, t). \tag{2.19}$$

2.3.2. Energy Discretization and the Multi-Group Diffusion Equations

To make the diffusion equation computationally tractable, the neutron flux and cross-sections are discretized into a finite number of energy groups. The group flux is defined as:

$$\phi_g(\mathbf{r}, t) = \frac{1}{\Delta E_g} \int_{E_g}^{E_{g+1}} \phi(\mathbf{r}, E, t) dE. \tag{2.20}$$

Similarly, macroscopic cross-sections are averaged over each energy group:

$$\Sigma_{x,g}(\mathbf{r}) = \frac{\int_{E_g}^{E_{g+1}} \Sigma_x(\mathbf{r}, E)\phi(\mathbf{r}, E, t) dE}{\int_{E_g}^{E_{g+1}} \phi(\mathbf{r}, E, t) dE}. \tag{2.21}$$

Applying this discretization to the neutron balance equation leads to the **multi-group diffusion equations**:

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} - \nabla \cdot D_g \nabla \phi_g + \Sigma_{t,g}\phi_g = \sum_{g'=1}^{G} \Sigma_{s,g'\to g}\phi_{g'} + \chi_g \sum_{g'=1}^{G} \nu \Sigma_{f,g'}\phi_{g'} + S_{\text{ext},g}. \tag{2.22}$$

2.3.3. Spatial Approximations and the Helmholtz Equation

With both energy and space discretizations applied, solving the multi-group diffusion equations requires further mathematical treatment. Assuming a steady-state reactor, the neutron flux within each homogeneous material region is assumed separable in space and energy:

$$\phi_g(\mathbf{r}) = \Phi_g R(\mathbf{r}). \tag{2.23}$$

Substituting into Eq. 2.22 and assuming $\frac{\partial \phi_g}{\partial t} = 0$, we obtain:

$$\nabla^2 \Phi_g + B^2 \Phi_g = 0. \tag{2.24}$$

This is the **Helmholtz equation**, which describes the spatial distribution of the neutron flux in steady-state conditions. The term B^2 represents the **geometric buckling**, which quantifies how the neutron flux spatially varies within the reactor. It arises from the

Laplacian term $\nabla^2\Phi_g$ in the equation, indicating the curvature of the neutron flux distribution. A higher B^2 corresponds to a more confined neutron flux shape, while a lower B^2 suggests a more spread-out distribution. The Helmholtz equation does not account for time dependence, which is crucial for transient analysis. This aspect will be addressed in the next section through the **Point Kinetics equations**.

The diffusion equation provides a sufficiently accurate representation of reactor physics under well-homogenized conditions. However, its validity is limited in:

- **Boundaries and interfaces**, where neutron streaming effects are significant.
- **Highly absorbing materials**, where leakage is not well represented.
- **Localized sources**, where the assumption of an isotropic flux is inaccurate.

Despite these limitations, proper homogenization and condensation allow the diffusion approach to effectively describe the main characteristics of a reactor core. It will be seen in further sections that several of these approximations can be evaded through a **MC-based full-core oriented approach**.

2.4. Point Reactor Kinetics Model

In the previous section, the Helmholtz equation (Eq. 2.24) was derived under the assumption that the neutron flux can be separated into spatial and energy-dependent components in a steady-state system. The general solution of this equation is a linear combination of spatial modes, each associated with an eigenvalue B_n^2, determining the geometric buckling of the flux distribution.

To extend this formulation to time-dependent problems, it is assumed that the time dependence can also be factorized, leading to a flux representation of the form:

$$\phi_g(\mathbf{r}, t) = \sum_{n=1}^{\infty} \Phi_{n,g}(\mathbf{r}) T_n(t), \tag{2.25}$$

where each mode $\Phi_{n,g}(\mathbf{r})$ has an associated time-dependent amplitude function $T_n(t)$. Substituting Eq. 2.25 into the multi-group diffusion equation and using the Helmholtz equation to simplify the spatial dependence, the following equation is obtained for each mode n:

$$\frac{1}{v}\frac{dT_n(t)}{dt} + \left(DB_n^2 + \Sigma_a - \nu\Sigma_f\right) T_n(t) = 0. \tag{2.26}$$

Solving this first-order differential equation gives:

$$T_n(t) = T_n(0)e^{-\lambda_n t}, \tag{2.27}$$

where the time decay constant is defined as:

$$\lambda_n = v\left(DB_n^2 + \Sigma_a - \nu\Sigma_f\right). \tag{2.28}$$

Since higher-order modes (B_n^2) decay faster than the fundamental mode ($B_1^2 = B_g^2$), only the lowest eigenmode dominates for long timescales. This leads to a simplification where the flux shape remains constant, and only its amplitude varies over time, giving rise to the **Point Kinetics Model (PK)**.

Under the point kinetics approximation, the neutron flux shape is assumed to be time-independent, reducing the problem to solving for the total neutron population $n(t)$, where:

$$n(t) \propto \int_V \int_0^\infty \phi(\mathbf{r}, E, t) dE d^3r. \tag{2.29}$$

The time-dependent neutron balance equations, including delayed neutrons and the evolution of precursor concentrations, are expressed as:

$$\begin{cases} \dfrac{dn(t)}{dt} = \dfrac{\rho - \beta}{\Lambda} n(t) + \displaystyle\sum_{j=1}^{J_d} \lambda_j C_j(t), \\ \dfrac{dC_j(t)}{dt} = \dfrac{\beta_j}{\Lambda} n(t) - \lambda_j C_j(t), \quad j = 1, \ldots, J_d. \end{cases} \tag{2.30}$$

These equations describe the reactor power evolution over time, considering the contributions from prompt and delayed neutrons.

The standard point kinetics model does not account for energy-dependent effects, particularly the different spectral characteristics of prompt and delayed neutrons. Delayed neutrons are emitted at lower energies, making them more effective in sustaining fission in thermal reactors than in fast reactors. To correct for this, the **effective delayed neutron fraction** β_{eff} is introduced, defined as:

$$\beta_{\text{eff},j} = \frac{\int_V \int_0^\infty \phi^\dagger(\mathbf{r}, E)\beta_j(E)\Sigma_f(\mathbf{r}, E)\phi(\mathbf{r}, E) dE d^3r}{\int_V \int_0^\infty \phi^\dagger(\mathbf{r}, E)\Sigma_f(\mathbf{r}, E)\phi(\mathbf{r}, E) dE d^3r}, \tag{2.31}$$

where $\phi^\dagger(\mathbf{r}, E)$ is the adjoint neutron flux. This weighting accounts for the contribution of delayed neutrons to the overall chain reaction, making β_{eff} a more accurate parameter than β. A similar flux-weighting is applied to define the **effective neutron generation time** Λ_{eff}.

The point kinetics equations provide a powerful tool for analyzing reactor power evolution following a reactivity insertion. The presence of delayed neutrons is critical in determining the response characteristics. The system behavior changes significantly depending on the magnitude of the inserted reactivity ρ relative to β_{eff}:

- **For $\rho < \beta_{\text{eff}}$ (Delayed Critical Condition):** The power change is initially rapid (governed by prompt neutrons) but is then moderated by delayed neutrons, leading to a slower, controlled power evolution. This is known as the **prompt jump** behavior.
- **For $\rho > \beta_{\text{eff}}$ (Prompt Supercritical Condition):** The reactor becomes supercritical based only on prompt neutrons, leading to a very rapid power increase. This scenario is highly dangerous as it occurs on the order of the **prompt neutron lifetime**, making external control mechanisms ineffective.

When power increases due to positive reactivity insertion, the concentration of delayed neutron precursors also increases. As a result, even if reactivity is brought back to zero, the system does not return to its initial power level but stabilizes at a higher steady-state power level.

In practice, reactor safety is designed to ensure that normal operation remains in the **delayed critical** regime, where power changes occur on a timescale manageable by control rods and other control mechanisms.

2.5. Using MC to solve the neutron transport

In general, a particle is emanated from a source (fixed or fission) with a random spatial position, random direction, and random energy. Each particle has a chance of traveling freely in a medium before undergoing an interaction with nuclei. Different types of interactions may occur, depending on the particle type and energy, and the composition of the medium. These interactions, which can be described by pdfs that are established using nuclear data and physics models, may lead to production of one or more particles, termination of the particle, and/or a change in particle energy or direction. A particle also may be terminated if it escapes from the medium. In a Monte Carlo simulation, the "history" of each particle from birth to death is followed, expected tallies of interest

are estimated by simulation of numerous histories, and the associated variance and/or relative uncertainties for these tallies are evaluated [9].

To illustrate this, the estimation of a neutron track length can be discussed. The pdf describing the probability of a neutron undergoing a collision at a distance x is given by:

$$p(x)dx = \Sigma_t e^{-\Sigma_t x} dx, \tag{2.32}$$

where Σ_t is the total macroscopic cross-section [6]. Integrating this expression to obtain the cdf and equating it to a uniformly distributed random number ξ in the range $[0, 1]$, we obtain:

$$\xi = \int_0^x p(x')dx' = \int_0^x \Sigma_t e^{-\Sigma_t x'} dx'. \tag{2.33}$$

Applying the inversion method, the neutron track length x is given by:

$$x = -\frac{\ln \xi}{\Sigma_t}. \tag{2.34}$$

By sampling N random numbers ξ_i, the mean track length can be estimated as:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} \left(-\frac{\ln \xi_i}{\Sigma_t} \right). \tag{2.35}$$

The interaction of a neutron with matter (e.g., fission, scattering, absorption) can then be determined by comparing another sampled random number to the ratio of the absorption to total cross-sections. In an analog absorption scheme, if absorption occurs, the neutron history is terminated, and its energy is deposited at that location. However, this method is computationally inefficient, leading to the adoption of the implicit absorption method. In this approach, each neutron is assigned a statistical weight W_n that represents the number of neutrons the sampled history corresponds to. Instead of terminating the neutron at the capture site, its weight is reduced according to the capture probability:

$$W_n^{\text{new}} = \left(1 - \frac{\sigma_a}{\sigma_t}\right) W_n^{\text{old}}, \tag{2.36}$$

where σ_a and σ_t are the absorption and total cross-sections at the interaction site. This allows neutrons to continue propagating while statistically accounting for absorption. To

prevent excessively large or small weights from skewing results, splitting and Russian roulette techniques are employed. In splitting, a particle with weight W_0 is divided into n particles, each with a weight W_0/n. In Russian roulette, a neutron of weight W_0 is either eliminated or increased to $W_1 > W_0$ with probability W_0/W_1, while being terminated with probability $1 - W_0/W_1$ [15].

Regarding neutron transport simulations, MC methods distinguish between two main types of calculations:

- Fixed-source problems: Neutrons originate from a known external source with a predefined spatial, angular, and energy distribution. This method is typically used for shielding and transient calculations.
- Criticality problems: These correspond to an eigenvalue problem where the chain reaction is artificially balanced. They are used for steady-state and burnup calculations.

In criticality simulations, neutrons are simulated in M cycles with N neutrons per cycle. The source distribution for each cycle is determined by sampling N fission sources from P fission sites in the previous cycle. Since the number of new fission neutrons typically differs from N, the neutron multiplication factor k emerges as an automatic scaling factor. However, convergence issues arise, as the spatial and energy distributions of the fission source are more challenging to converge than the multiplication factor itself. One widely used technique to analyze convergence is the evaluation of the Shannon entropy of the fission source distribution [15]. Proper initialization of fission source sites is also crucial, as a good initial guess accelerates convergence.

Regardless of the type of simulation, results are collected using estimators, which record sampled histories. Various techniques exist in MC codes to define accumulators (e.g., tallies or detectors), ensuring that results are reported both as expected values and with associated uncertainties [6, 15].

2.6. Using MC for transient neutronics calculations

MC simulations of transient problems are analogous to fixed-source calculations, with an additional time-binning technique used to track particle histories over discrete time intervals. However, when modeling multiplying media, specific challenges arise in maintaining a reasonable number of particles per time bin, particularly in subcritical and supercritical conditions [7]. Additionally, the treatment of delayed neutrons is crucial, as their characteristic emission time differs significantly from prompt neutrons.

Instead of directly sampling delayed neutrons, most MC codes track **delayed neutron precursors**, which are produced at fission sites and later decay into delayed neutrons. This method avoids storing neutrons that may be emitted several seconds later and improves computational efficiency [27]. The precursors are grouped into j families, each with a characteristic decay constant λ_j, as defined in Eq. 2.4.

When a precursor is produced at time t^* with an initial weight w_0, only a fraction of its weight survives until the **end of the interval (EOI)** due to radioactive decay, while the remaining fraction contributes to delayed neutron emission. This is schematically shown in Fig. 2.2, where t_0 represents the beginning of the interval (BOI) and t_1 the EOI.

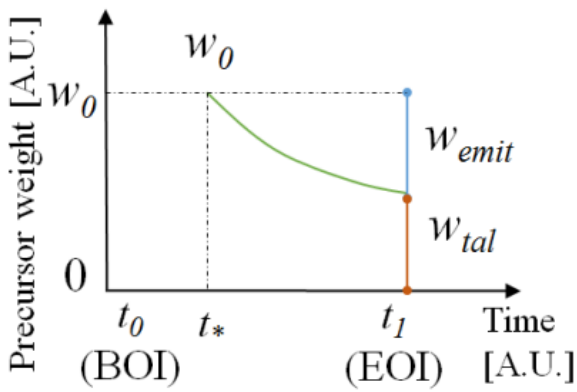


Figure 2.2: Scheme of precursors decay approach within Serpent [7].

The surviving precursor weight at the end of the interval is given by the decay law:

$$w_{\text{tal}} = w_0 e^{-\lambda_j (t^* - t_0)}, \tag{2.37}$$

while the fraction that contributes to neutron emission is:

$$w_{\text{emit}} = w_0 - w_{\text{tal}}. \tag{2.38}$$

Since w_{emit} is typically small compared to w_0, Russian roulette techniques are applied to either increase the neutron weight to a reasonable level or eliminate it.

The transient simulation follows a **two-step approach**:

1. **Initial Criticality Calculation** A preliminary **criticality simulation** is performed to obtain an initial neutron and precursor distribution. This ensures that the starting point of the transient correctly represents the reactor's steady-state configuration. Live

neutrons are typically stored at random times during their lifetimes to avoid tracking full histories. Precursors, on the other hand, are recorded using implicit estimators during active cycles [27]. These source distributions are saved in dedicated files for later use in the transient phase.

2. **Time-Dependent Fixed-Source Calculation** The transient simulation is then performed as a fixed-source problem, where two distinct external sources are considered simultaneously:

- **Live Neutrons:** Free neutrons propagating with a defined position, direction, and energy.
- **Delayed Neutron Precursors:** Precursor nuclei stored with their corresponding decay constants, governing the emission of delayed neutrons over time.

The time domain is discretized into n time bins, where the simulation progresses step-by-step from the BOI to the EOI. During each step, the behavior of precursors is tracked, and if changes in the reactor configuration occur (e.g., temperature or density variations, control rod movement), these modifications are also incorporated. A schematic representation of this process is shown in Fig. 2.3.

Ensuring an accurate initial distribution of **live neutrons and precursors** is critical for transient MC calculations. The ability to correctly reconstruct the reactor state at the start of the transient significantly affects the accuracy and efficiency of the simulation.

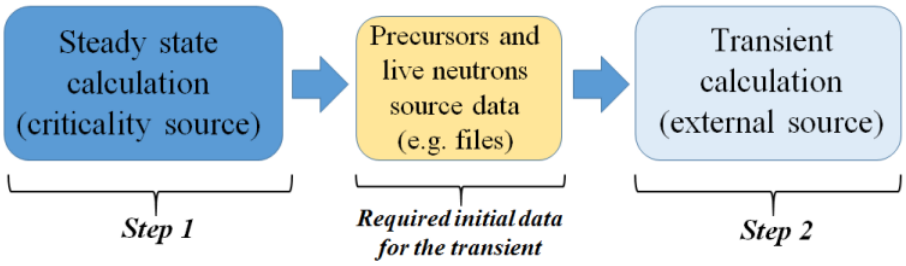


Figure 2.3: Two-step approach for transient calculations

In the case of a coupled transient calculation that accounts for both neutronics and thermodynamics from the initial state, the simulation framework requires some modifications. However, the fundamental two-step approach remains the basis for all MC transient calculations in this work. In the next section, the coupling between neutronics and thermodynamics will be introduced and discussed in detail.

2.7. Feedbacks to the neutronic problem: thermodynamic effects

The neutron distribution within a reactor is influenced by feedback mechanisms arising from changes in material properties due to temperature variations. In LWRs, the most significant feedbacks are associated with the moderator (coolant) temperature and the fuel temperature. While the coolant density directly affects neutron moderation, this study primarily focuses on the thermal feedback of the fuel, as the system is modeled under adiabatic conditions, meaning that coolant effects are not explicitly simulated.

The neutron flux spectrum in the modeled system, depicted in Fig. 2.4, shows two main peaks: the thermal peak, corresponding to moderated neutrons that sustain the fission chain reaction, and the fast peak, representing high-energy neutrons produced directly from fission events.

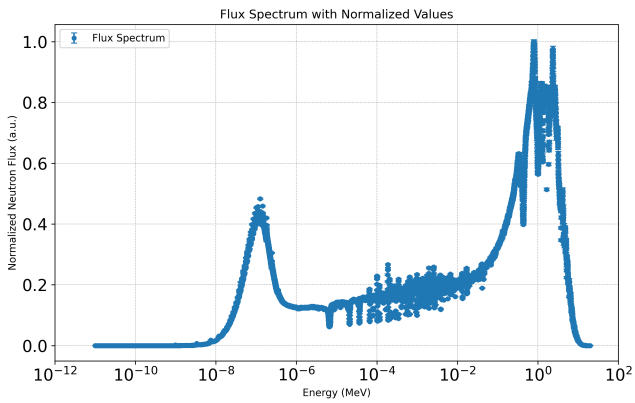


Figure 2.4: Energy dependence of the neutron flux in the assembly

Neutrons are initially emitted from fission with an energy distribution $\chi(E)$ that follows a Maxwellian spectrum with an average energy of about 2 MeV [6]. In a moderated system, these fast neutrons undergo successive elastic scattering interactions, primarily with hydrogen nuclei in water, losing energy until they reach the thermal range (eV), where they have the highest probability of inducing fission in ^{235}U.

However, during this moderation process, neutrons pass through the epithermal energy range, where ^{238}U exhibits strong absorption resonances. As illustrated in Fig. 2.5, the absorption cross-section of ^{238}U has pronounced peaks in this energy region, making it an effective neutron absorber. The presence of these resonances impacts the overall neutron balance, as some neutrons are lost before reaching the thermal region, reducing the

system's reactivity.

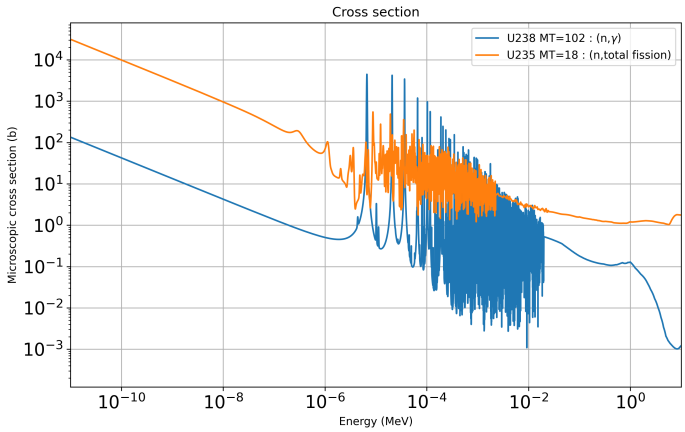


Figure 2.5: Most relevant nuclear cross sections plotted with Serpent 2.

The fuel temperature plays a crucial role in modifying these absorption characteristics through the Doppler broadening effect. As temperature increases, the thermal motion of ^{238}U nuclei causes a broadening of the absorption resonances, effectively spreading them over a wider energy range. This phenomenon, illustrated in the left part of Fig. 2.6, results in an increased probability of neutron absorption in the epithermal range, reducing the number of neutrons available for fission and thereby inducing a negative reactivity feedback.

There is a further phenomenon connected with spatial and energy self-shielding. **Spatial self-shielding** occurs in nuclear reactors due to the heterogeneous structure of the reactor core, particularly in the fuel rods. When neutrons are born during fission, they are initially moderated in the surrounding material (like water in light water reactors) and then return to the fuel to be absorbed and cause further fission. However, the outer layers of the fuel rod absorb more neutrons, shielding the inner layers from the neutron flux. This results in a lower neutron flux inside the rod. The effect is more significant near resonance energies, where the energy-dependent flux is drastically reduced, leading to a reduced probability of further fission in the inner parts of the fuel rod. **Energy self-shielding** refers to the reduction in neutron flux near the resonance energy of materials (like U-238) due to the increased neutron absorption probability in the resonance region, which lowers the effective absorption per nucleus. This effect occurs because the energy-dependent flux decreases near the resonance, leading to fewer neutrons being absorbed in this region. The energy self-shielding is significantly affected by the material temperature, as illustrated in the right part of Fig. 2.6, due to the fact that the energy-dependent neutron flux closely follows the behavior of the changed resonances in a reverse manner, leading to a reduction

in the self-shielding and an increase in the effective microscopic absorption cross-section. This, in turn, provides negative reactivity feedback against fuel temperature increase. This effect is fundamental for the stability of thermal reactors, as it provides an inherent safety mechanism by limiting power excursions.

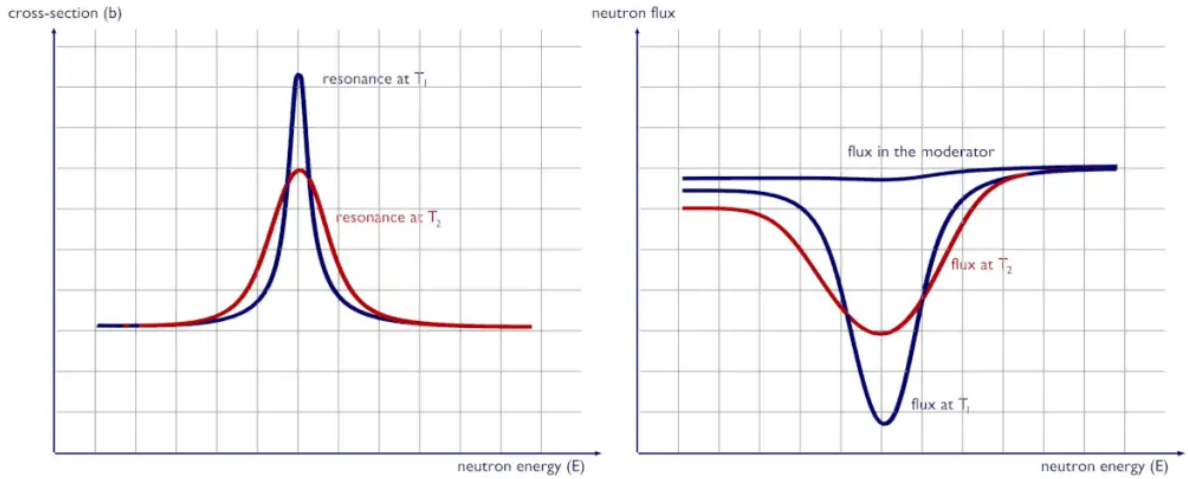


Figure 2.6: An increase in temperature from T1 to T2 causes the broadening of spectral lines of resonances. Although the area under the resonance remains the same, the broadening of spectral lines causes an increase in neutron flux in the fuel $\phi_f(E)$, increasing the absorption as the temperature increases [24].

Accurately modeling these temperature-dependent feedback mechanisms is essential for capturing the dynamic behavior of the reactor. In Chapter 5, the approach for representing this thermal effect and its integration into Monte Carlo neutronic calculations will be presented.

Conclusion

In recent years, there has been a growing international interest in developing more accurate simulation tools for reactor physics. Unlike traditional industrial approaches, which mostly rely on deterministic diffusion codes such as PANTHER and are based on simplified models and decoupled physics, new methods aim to reduce approximations and improve precision. In particular, Monte Carlo codes have gained attention for their ability to handle complex reactor configurations without relying on many of the standard assumptions. Thanks to recent advancements [28], they now offer multi-physics coupling

and transient analysis features. Combined with access to high-performance computing, these tools represent a promising alternative for next-generation reactor simulations.

3 | Description of SERPENT model and Steady-state analysus

Abstract

This chapter describes the fuel assembly model in an infinite core of the NuScale reactor adopted in this work. Two main aspects are discussed: the input model, emphasizing the geometric configuration, and the convergence procedure. Moreover, this chapter presents the steady-state analysis conducted as a preliminary step in preparing the system for transient simulations. This part is organized into three main sections. The first section focuses on the preliminary analysis, where the system is simulated in criticality mode under different conditions to explore various possible configurations of the model, and where some results obtained with Serpent are also compared with Panther to provide a basic verification reference. The second section discusses the extraction of kinetic parameters essential for transient analysis. Finally, the third section details the steps required to set up the transient simulation.

3.1. Input model and geometric configuration

The Serpent input model used in this work is provided in Appendix A. The geometric configuration is based on the model developed by Agostin ([1]) from which the fuel assembly of interest has been extracted for this study. The input file is structured with linked files to include different components of the geometry: materials, fuel rods, control rods, guide tubes, empty positions, lattice (having a 1/8 symmetry), and pool. Particular attention has been given to the pool definition, which contains the entire fuel assembly and is defined by a cell belonging to universe 0, as this universe must always be present in the Serpent input to ensure a properly defined geometry.

Before running Serpent, the environment must be properly configured by setting the

necessary environment variables and defining an alias in the `.bashrc` file. This ensures Serpent has access to the required libraries, specifically the ENDF/B-VII cross sections library, which are defined on specific temperatures, but, thanks to Serpent it is possible to interpolate between them obtaining the cross sections at arbitrary temperatures.

Model setup and simulation parameters are the following:

- Control rods movement: the `trans` card is used to simulate their motion, allowing for fully inserted (ARI), fully withdrawn (ARO), or intermediate positions.
- Boundary conditions (`set bc` card): - Radial BCs: set to reflective to simulate an infinite environment, where neutrons reaching the boundary are reintroduced into the system, preventing leakage. - Axial BCs: represented by water above and below the assembly, ensuring an infinite environment in the axial direction as well.
- Power normalization (`set powdens` card): The power density method is used for source normalization, with a value of 17.081 kW/kgU (17.081×10^{-3} kW/gU, as required by Serpent). A normalization issue arose due to Serpent's incorrect fuel volume calculation (likely due to the 1/8 symmetry causing volume overlaps). This was resolved by manually specifying the fuel volume using the `set mvol` "fuel250" card option, with a value computed as follows:

$$V = \pi \times r^2 \times H \times N_{\text{pins}} = \pi \times (0.4058\,\text{cm})^2 \times 200\,\text{cm} \times 264 = 27315.42\,\text{cm}^3 \quad (3.1)$$

where: $r = 0.4058$ cm is the radius of a single fuel pin, $H = 200$ cm is the fuel height, $N_{\text{pins}} = 264$, as the fuel assembly is a 17×17 pin array with 25 guide tubes removed.

This volume allows for the calculation of the mass of uranium in the assembly. The molar mass of UO_2 is:

$$M_{UO_2} \approx 238 + (2 \times 16) = 238 + 32 = 270\,\text{g/mol} \quad (3.2)$$

The percentage of uranium in UO_2 is given by:

$$\frac{238}{270} \times 100 = 88.15\% \quad (3.3)$$

The mass of uranium in the assembly is computed as:

$$M_{\mathrm{U}} = V \times \rho_{\mathrm{fuel}} \times \left(\frac{238}{270}\right) = 27315.42\,\mathrm{cm}^3 \times 10.5216\,\mathrm{g/cm}^3 \times 0.8815 = 253.34\,\mathrm{kgU} \tag{3.4}$$

where: $\rho_{\mathrm{fuel}} = 10.5216$ g/cm^3 is the fuel density.

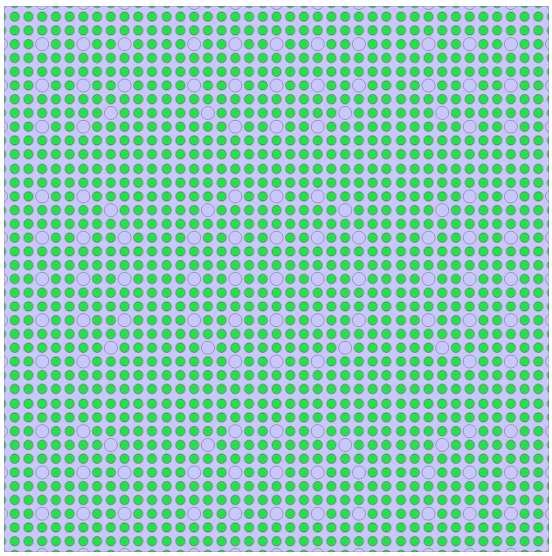
The power of the fuel assembly is determined using the power density:

$$P = \text{Power density} \times M_{\mathrm{U}} = 17.081\,\mathrm{kW/kgU} \times 253.34\,\mathrm{kgU} = 4327.32\,\mathrm{kW} \tag{3.5}$$

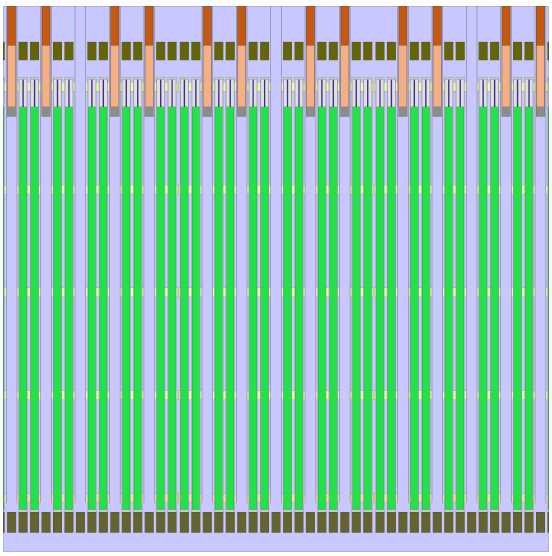
This final result represents the total power output of the fuel assembly.

- Neutron population settings (`set pop` card): the number of simulated neutrons per cycle is specified, along with the number of active and inactive cycles in the criticality simulation.
- Fission source initialization (`src` card): to ensure proper fission source sampling, an axial source distribution (`sz 0 200`) was set. This was necessary as Serpent was otherwise unable to initiate the simulation due to insufficient fission events in the initial source population.

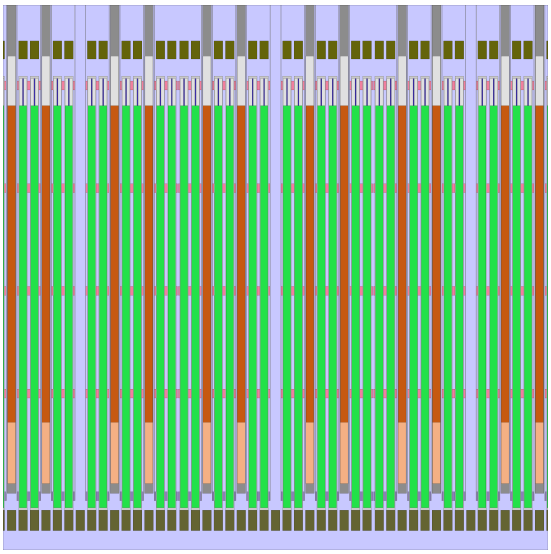
Geometry plots of the Serpent model are shown below:



(a) Horizontal cut of the system, reflective BCs.



(b) Vertical cut through assembly centerline (YZ-plot), ARO configuration.



(c) Vertical cut through assembly centerline (YZ-plot), ARI configuration.

Figure 3.1: Geometry plots of the Serpent model.

3.2. Statistical error and convergence of criticality calculations

The size of the neutron population and the number of iterations in calculations on criticality mode were chosen in order to guarantee the convergence of k with an acceptable statistical error. To obtain correct results, free of contamination from the initial guess

for the fission distribution, it is imperative to determine when the iteration procedure has converged, so that a sufficient number of the initial batches are discarded. Batches are thus divided into two types: inactive, where the distribution is not yet converged, and active, where stationarity has been reached and Monte Carlo tallies are accumulated. Determining convergence is complicated by the statistical noise inherent in the random walks of the neutrons in each generation. Studies [3] have shown that the effective multiplication factor (k_{eff}) converges faster than the fission source distribution. Therefore, it is essential to monitor the fission source convergence using a dedicated parameter: the Shannon entropy of the fission source distribution (H_{src}). This metric provides a single numerical value for each batch, helping to characterize the convergence of the fission source distribution.

The Shannon entropy for a given batch is defined as:

$$H_{\text{src}} = -\sum_{J=1}^{N} P_J \cdot \ln_2(P_J) \tag{3.6}$$

H_{src} varies between 0 for a point distribution to $\ln_2(N_s)$ for a uniform distribution. To compute H_{src}, a 3D grid must be superimposed over the entire system, encompassing all fissionable regions. The number of fission sites per batch in each grid cell is recorded, forming a discretized estimate of the source distribution, $\{P_J, J = 1, N_s\}$, where:

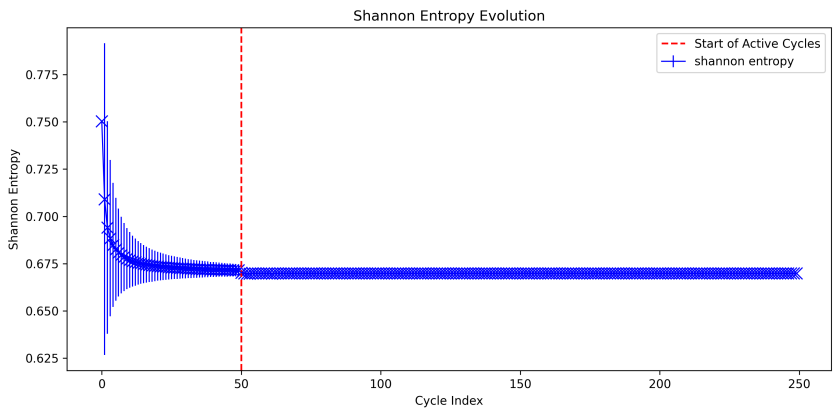
- N_s is the total number of grid cells.
- $P_J = \frac{\text{number of source sites in the } J\text{-th grid cell}}{\text{total number of source sites}}$.

To monitor the Shannon entropy evolution in Serpent, the following input command is used:

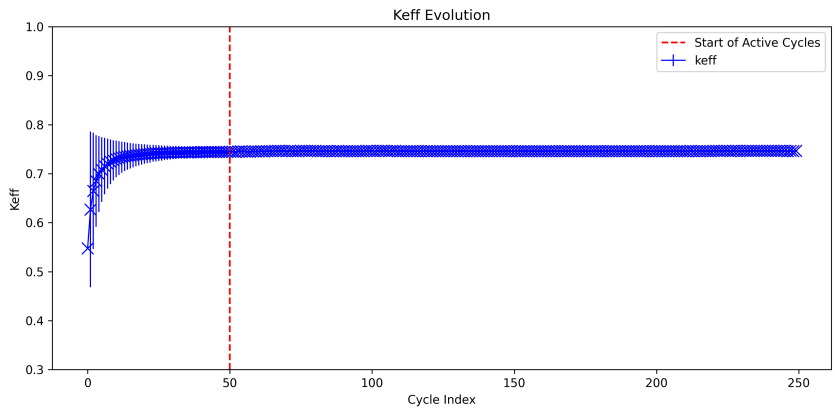
```
set his 1
set entr 100 100 20
```

where "100 100 20" specifies the number of mesh cells in the x, y, and z directions.

Below, we present the Shannon entropy and k_{eff} evolution over the first 50 inactive cycles and the subsequent 200 active cycles.



(a) Shannon entropy evolution



(b) k_{eff} evolution

Figure 3.2: Evolution of Shannon entropy and k_{eff} over the simulation cycles.

From the images, it can be concluded that 50 inactive cycles are sufficient, as both k_{eff} and the fission source have converged. Additionally, the images illustrate how the error decreases as the cycles progress, stabilizing at low values after the 50 inactive cycles and remaining low throughout the subsequent 200 active cycles.

3.3. Preliminary Analysis

The system was first simulated in criticality mode in both ARO (All Rods Out) and ARI (All Rods In) configurations. These simulations provided the first results, including the multiplication factor k_{eff}[1], as shown in the following table, along with the main parameters set for the simulation.

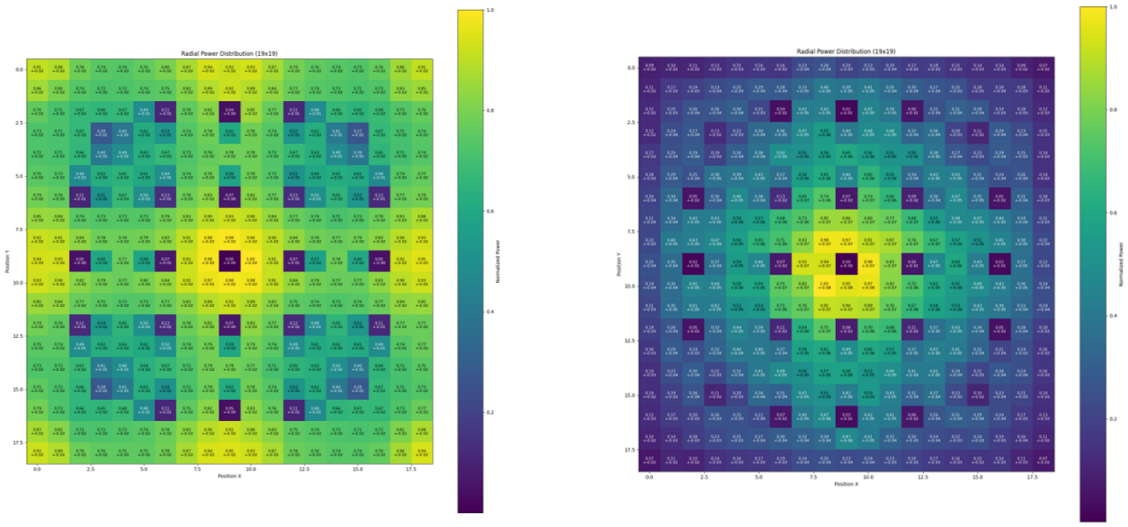
Steady-State Simulation Parameters

Parameter	**Value**
Simulated Neutrons	100000
Inactive Cycles	50
Active Cycles	200
CPU Time (min)	48.6
Configuration	$k_{\mathbf{eff}}$
ARO	**1.13256 ± 0.00018**
ARI	**0.74642 ± 0.00018**

As observed in the table, in the **ARI** configuration, the system is **subcritical** since the control rods are fully inserted, absorbing a significant number of neutrons and preventing the chain reaction from being sustained. On the other hand, in the **ARO** configuration, the k_{eff} value indicates that the system is **supercritical**, meaning that the number of neutrons increases progressively, leading to an acceleration of the reaction.

Subsequently, considering the ARO configuration, additional simulations were conducted by varying the BC. The system was analyzed using both `reflective` and `black` BCs. Using dedicated detectors in the Serpent input, the axial and radial power distributions were extracted, as illustrated in the following figures.

[1]In this thesis, I consider the k_{eff} value obtained using the implicit Monte Carlo method rather than the analog method, as it provides greater accuracy. This is due to its lower statistical variance and relative error, since it accounts for neutron weights instead of discarding absorbed neutrons, ensuring more efficient use of computational resources and faster convergence.



(a) Reflective boundary condition

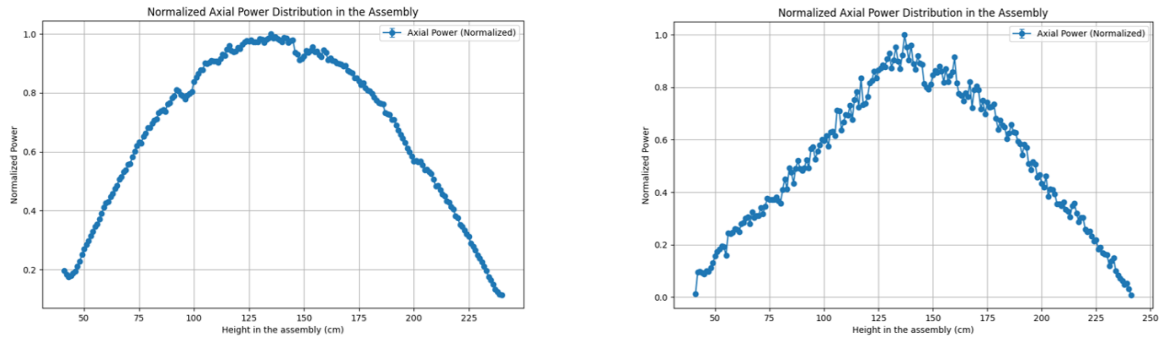
(b) Black boundary condition

Figure 3.3: Radial power distributions for reflective (a) and black (b) boundary conditions.

The radial power distribution shown in Figure 3.3a exhibits peaks at the center, edges, and corners of the assembly. This occurs because the concentration of fuel rods is higher in these regions, leading to increased fission rates. The presence of a reflector enhances neutron moderation and reflection, further contributing to the power peaks at the periphery.

In contrast, the radial power distribution in Figure 3.3b shows a different behavior. While a peak remains at the center of the assembly, the power at the edges and corners is significantly lower. This is due to the absence of a reflector, which allows neutrons to escape the core without returning, reducing neutron availability in these regions.

In both cases, the guide tubes, which are composed of water, exhibit zero power since they do not contain fissile material. Moreover, the radial power distribution maintains a symmetric pattern, reflecting the symmetric nature of the assembly itself. However, as noted in Monte Carlo simulations, statistical variations can introduce minor asymmetries despite the underlying geometric symmetry.



(a) Reflective boundary condition

(b) Black boundary condition

Figure 3.4: Axial power distributions along the assembly for reflective (a) and black (b) boundary conditions.

The axial power distribution shown in Figure 3.4a exhibits a peak at the center of the assembly. This trend follows the expected cosinusoidal shape of the axial neutron flux distribution, where the flux is highest in the middle and decreases toward the axial boundaries due to leakage and absorption effects.

In Figure 3.4b the axial power distribution has a steeper drop near the axial boundaries compared to the reflective case. This is because, without reflective boundary conditions, neutron leakage is higher at the axial edges, leading to a more rapid decrease in neutron flux.

Comparison between Panther and Serpent

An interesting result from the criticality simulations concerns the comparison between the effective multiplication factor k_{eff} obtained with the deterministic diffusion code Panther and Serpent. Both simulations were performed on the same infinite lattice model of the fuel assembly, using the same nuclear data library (ENDF/B-VII.1). Specifically, Panther used the `/w10e71v1.dat` library, while Serpent used the `s2v0_endfb71.xsdata` file. The configuration considered corresponds to all control rods out (ARO), with a uniform fuel temperature of 900 K.

The table below presents the boron concentration used and the resulting k_{eff} values from both codes:

	Panther	Serpent	Delta
Boron concentration	2146.5 ppm	2146.5 ppm	0 ppm
k_{eff}	1.00000	1.00177 ± 0.00018	177 pcm

Table 3.1: Comparison of criticality results between Panther and Serpent (ARO configuration, T = 900 K, ENDF/B-VII.1).

The results show a difference of 177 pcm between the two simulations. This deviation is considered acceptable given the stochastic nature of Monte Carlo simulations and the fact that Serpent directly tracks particles in continuous energy, while Panther relies on multigroup diffusion approximations.

The following sections will discuss the extraction of kinetic parameters and the preparation for transient simulations.

3.4. Extraction of Kinetic Parameters

As part of the steady-state analysis, the kinetic parameters of the system were extracted. These parameters play a fundamental role in transient simulations as they define the neutron population behavior over time. The extracted quantities include: the effective delayed neutron fraction (β_{eff}), the prompt neutron lifetime (Λ), the delayed neutron precursor fractions (β_i) for six precursor groups, the decay constants (λ_i) associated with the precursor groups.

To obtain these parameters, different methodologies implemented in Serpent were used [11]:

- **Meulekamp Method:** The Meulekamp method estimates the effective delayed neutron fraction (β_{eff}) by computing the fraction of fissions caused by delayed neutrons in the next generation. It provides a simple and computationally efficient approach, requiring minimal modifications to Monte Carlo codes. However, it introduces approximations that may lead to significant biases, especially in fast reactors.
- **Nauchi Method:** The Nauchi method improves the estimation by considering the expected number of fission neutrons emitted in the next generation. It also estimates the neutron generation time (Λ) by introducing a perturbation in the system using a fictitious 1/v absorber. This method provides better accuracy than Meulekamp's approach but requires two separate simulations, increasing computational cost. [12]

- **Iterated Fission Probability (IFP) Method:** The IFP method extends the concept of next fission probability by tracking neutron populations over multiple generations. It calculates the total number of neutrons produced from a single initiating fission event, making it proportional to the adjoint flux. This method yields the most accurate estimates of β_{eff} and Λ, but it is computationally expensive due to the need for extensive Monte Carlo sampling over many neutron generations.

For consistency, the values reported in the following tables were obtained using the IFP method, as it provides the highest accuracy. The kinetic parameters were extracted for three different system configurations:

1. **ARI**
2. **ARO**
3. **Critical (At 100 Steps):** A configuration in which the system is made critical using the *Critical Density Iteration* option, which allowes to adjust the boron concentration in the water through an iterative process to achieve the desired k value (set to 1 in this case). In this configuration, the control rods are raised to 100 steps, and criticality is achieved considering this specific configuration.

ARI and ARO were simulated with 100000 neutrons, 200 active cycles, and 50 inactive cycles, while the Critical (100 steps) configuration was simulated with 1200000 neutrons and the same number of cycles as the other configurations.

Extracted Kinetic Parameters (IFP Method)

Parameter	ARI	ARO	Critical (100 Steps)
β_{eff} [pcm]	703.5 ± 22.3	678.7 ± 17.9	693.5 ± 5.5
Λ [s]	2.92E-05 ± 1.1E-07	2.45E-05 ± 5.3E-08	2.39E-05 ± 1.8E-08

Table 3.2: Kinetic Parameters extracted using the IFP method, reported with a statistical uncertainty of $\pm 2\sigma$.

Additionally, the delayed neutron fraction (β_i) and decay constant (λ_i) were determined for each precursor family in all three configurations:

Precursor Group Parameters (IFP Method)

Group	ARI		ARO		Critical (100 Steps)	
	β_i [pcm]	λ_i [s^{-1}]	β_i [pcm]	λ_i [s^{-1}]	β_i [pcm]	λ_i [s^{-1}]
1	20.6 ± 3.7	$0.0133 \pm 1.2E-05$	24.3 ± 3.3	$0.0133 \pm 1.1E-05$	23.1 ± 1.1	$0.0133 \pm 3.5E-06$
2	121.6 ± 9.8	$0.0325 \pm 4.0E-05$	113.5 ± 6.6	$0.0326 \pm 2.6E-05$	117.6 ± 2.2	$0.0325 \pm 8.5E-06$
3	107.8 ± 8.6	$0.1212 \pm 8.5E-05$	103.4 ± 6.3	$0.1211 \pm 5.3E-05$	114.5 ± 2.4	$0.1211 \pm 1.7E-05$
4	273.8 ± 15.1	$0.3070 \pm 4.6E-04$	266.9 ± 11.9	$0.3061 \pm 2.6E-04$	268.9 ± 2.9	$0.3063 \pm 9.8E-05$
5	131.2 ± 9.4	$0.8661 \pm 2.0E-03$	121.1 ± 7.2	$0.8619 \pm 1.4E-03$	120.0 ± 2.3	$0.8635 \pm 4.7E-04$
6	48.5 ± 6.6	$2.9107 \pm 1.04E-02$	49.3 ± 4.6	$2.899 \pm 8.2E-03$	49.4 ± 1.6	$2.899 \pm 2.5E-03$

Table 3.3: Precursor Group Parameters using the IFP method, reported with a statistical uncertainty of $\pm 2\sigma$.

The results presented in the table show variations across the three cases. Typically, these parameters are constant, but the results from the three different setups show how they actually vary, reflecting the fact that these kinetic values are integrals, and thus depend on the specific configuration considered. It is interesting to note how the three different configurations also affect the factors of the six-factor formula, as shown in the table below.

Six-Factor Formula Parameters (ARI, ARO, Critical 100 Steps)

Factor	ARI	ARO	Critical (100 Steps)
η	1.77 ± 8.5E-04	1.77 ± 7.1E-04	1.77 ± 2.1E-04
f	0.54 ± 2.9E-04	0.78 ± 2.3E-04	0.67 ± 7.8E-05
ε	1.35 ± 6.5E-04	1.23 ± 3.9E-04	1.28 ± 1.4E-04
p	0.57 ± 2.5E-04	0.66 ± 1.8E-04	0.66 ± 7.1E-05
P_{NL}^{F}	1.00 ± 0.0E+00	1.00 ± 0.0E+00	1.00 ± 0.0E+00
P_{NL}^{T}	1.00 ± 0.0E+00	1.00 ± 0.0E+00	1.00 ± 0.0E+00

Table 3.4: Six-Factor Formula parameters extracted for the three different cases (ARI, ARO, and Critical 100 Steps), reported with a statistical uncertainty of $\pm 2\sigma$.

The kinetic parameters are crucial in determining the system's time-dependent behavior during transients. The extracted values will be used as fundamental input for the transient simulations in the following sections.

3.5. Preparation for Transient Simulation

Three fundamental steps were necessary to prepare for transient simulations, all of which had to be performed in criticality mode.

Criticality State with Criticality Density Iteration

First, the system needed to be brought to a critical state using the *Criticality Density Iteration* method, which was discussed earlier. This process required multiple simulations until the iteration factor output by Serpent stabilized, with only small percentage changes. Achieving this critical state required an accurate concentration of boron-10 and boron-11 (components of the water) that were iteratively adjusted to achieve a $k = 1$. Boron is used in reactors to absorb neutrons, and by calibrating its concentration, the desired k value can be obtained, ensuring the system reaches criticality. The following code was used for the critical density iteration:

```
% ---- Critical density iteration
set iter nuc 50 1.00000 2 50100 50110 1 water
```

Where: - `50100` and `50110` are the ZAI of the nuclides to be included in the iteration (e.g., `50100` for boron-10 ground state). - `1.00000` is the desired k value. - `2` is the number of nuclides for which the concentration will be varied. - `1` refers to the number of materials, in this case, water, where the nuclide concentrations will be adjusted. - `50` indicates the number of additional inactive cycles to run for the convergence of the iteration.

Criticality Calculations for Source File Generation

Once a stable, critical reactor configuration was achieved, the next step was to perform criticality calculations to produce the neutron source files containing both prompt and delayed neutrons. This was done using the input option `set savesrc "./source"`, where "source" refers to the names of the files containing the steady-state neutron and delayed neutron precursor sources. These files will later be linked to the transient simulation input, as explained in the following chapter. It is crucial that these source files accurately represent the system in its critical configuration, as they will serve as the initial condition for the transient analysis. These files essentially provide a snapshot of the system before any reactivity insertion occurs.

Generating the initial source for the dynamic simulation is done with a criticality source simulation. The required modification to the input is the addition of the following line

[17]:

```
set savesrc FILEPATH [PN PP NX NY NZ]
```

Here, `[FILEPATH]` indicates where Serpent will save the generated source distributions. Serpent generates four source files:

- `[FILEPATH].main` - containing general information of the generated source.
- `[FILEPATH].prec` - containing a precursor source tallied on a Cartesian mesh.
- `[FILEPATH].live` - containing the live neutron source (binary).
- `[FILEPATH].precpoints` - containing a point-wise precursor source (binary).

To obtain such a source from a Monte Carlo simulation, we need to store neutrons at random times during their life. If we don't want to slow down our criticality source calculation by stopping the neutrons at various times during the simulation, we can save neutrons at their tentative interaction sites. After a path length has been sampled for a neutron and it has been transported to a new (possible) interaction site, information such as position, velocity, and energy is stored in a source file.

However, these tentative interaction sites are not uniformly distributed in time or space. If neutrons were stored at each interaction site with a fixed probability, we would encounter significant biases: too many fast neutrons would be stored (since fast neutrons interact more frequently due to their higher velocity), and too many neutrons from materials with large macroscopic cross-sections would be captured (as materials with larger cross-sections have more interactions per unit time). This non-uniform sampling would lead to an incorrect representation of the neutron distribution and would not provide an accurate snapshot of the system.

To correct this, the probability of storing neutrons must be adjusted to account for the fact that fast neutrons and neutrons in high cross-section materials are more likely to interact. The probability of saving a neutron at a given interaction site should thus be proportional to the mean free path and interaction frequency, which depends on the neutron's energy and the material's cross-section.

The mean free path for a neutron with energy E is given by:

$$\lambda_{\text{mean}}(E) = \frac{1}{\Sigma_{\text{tot}}(E)},$$

where $\Sigma_{\text{tot}}(E)$ is the total macroscopic cross-section over the path length. The time it takes

for the neutron to travel this path length gives the mean time between two interactions:

$$t_{\text{mean}}(E) = \frac{\lambda_{\text{mean}}(E)}{v(E)} = \frac{1}{\Sigma_{\text{tot}}(E)v(E)},$$

where $v(E)$ is the neutron's velocity.

The mean interaction frequency is then:

$$f_{\text{mean}}(E) = \Sigma_{\text{tot}}(E)v(E).$$

To sample neutrons uniformly in time, the probability of saving a given neutron at an interaction site should be proportional to:

$$P \propto \frac{1}{f_{\text{mean}}(E)} = \frac{1}{\Sigma_{\text{tot}}(E)v(E)}.$$

A normalization constant A is then introduced to ensure the probability is less than one:

$$P = A\frac{1}{\Sigma_{\text{tot}}(E)v(E)}.$$

In non-analog simulations, some neutrons may have a non-unity weight w, meaning they represent a larger or smaller group of neutrons than the average neutron. This is taken into account by including the weighting term in the probability:

$$P = A\frac{w}{\Sigma_{\text{tot}}(E)v(E)}.$$

When delta tracking is used, the cross-section used to sample each path length may be larger than the local macroscopic total cross-section. This is handled by using the path length sampling cross-section $\Sigma_{\text{path}}(E)$ instead of $\Sigma_{\text{tot}}(E)$:

$$P = A\frac{w}{\Sigma_{\text{path}}(E)v(E)}.$$

The final step is to choose the normalization constant A, which is selected as:

$$A = \frac{1}{\max\left(\frac{w}{\Sigma_{\text{path}}(E)v(E)}\right)},$$

where the maximum is taken over all neutron energies and the entire geometry. This

ensures that the probability is always less than unity, with Serpent using an internal minimum cross-section for path length sampling in void regions.

The current implementation uses:

$$A = P_N v_{\min},$$

where $v_{\min}$ is the minimum velocity for neutrons, and P_N is a user-defined parameter that covers the variation in the minimum path length sampling cross-section and neutron weight (the default value is 1).

Finally, it is important to note that the system must be as close to criticality as possible, as any deviations result in errors in both the live neutron source and the precursor source, since the concentrations are assumed to be steady-state.

Setting the Iteration Factor for Transients

Finally, additional simulations in criticality mode were needed, setting the `set iter` value to the desired k in order to trigger the transient. Once the desired reactivity insertion for the transient is determined, the corresponding k value is calculated using the formula:

$$k = \frac{1}{1-\rho}$$

For instance, if we want to insert a reactivity of 1000 pcm, the corresponding k value is calculated as:

$$k = \frac{1}{1 - 1000 \times 10^{-5}} = 1.01010$$

Once the k value is determined, it is set in the input file using the `set iter` card. For this example, the corresponding input line is:

```
% ---- Critical density iteration
set iter nuc 50 1.01010 2 50100 50110 1 water
```

Serpent will then output an iteration factor, which is used to multiply the previous concentrations of boron-10 and boron-11 to obtain the new concentrations. These new concentrations are then reintegrated into the input file. This process must be repeated several times until the iteration factor stabilizes and does not change by more than a few percentage points, ensuring that the boron concentrations are as precise as possible to insert the exact desired reactivity into the system.

Conclusion

After introducing the geometry of the system and evaluating the convergence of the simulations by considering the Shannon entropy, this chapter presented the steady-state analysis performed as the first step in preparing the system for transient simulations.

The system was simulated in criticality mode under different conditions to explore various possible configurations of the model. Key results, such as the radial and axial power distributions, were obtained for both reflective and black boundary conditions. These distributions provided insights into the neutron flux behavior and its impact on the power output of the system, highlighting the differences in neutron moderation and absorption depending on the boundary conditions.

Kinetic parameters, which are crucial for the subsequent transient analysis, were then extracted. These parameters, including the effective delayed neutron fraction (β_{eff}) and prompt neutron lifetime (Λ), were calculated using the IFP method. The delayed neutron precursor parameters (β_i and λ_i) were also determined for each precursor group, providing the necessary input for the time-dependent simulations.

Finally, the steps required to prepare the system for transient simulations were outlined. These included bringing the system to criticality using the Critical Density Iteration method, generating the neutron source files using the criticality results, and setting the boron concentration based on the desired reactivity insertion. These preparatory actions are fundamental to ensure that the system is accurately set up for the transient analysis presented in the next chapter.

4 | Transient Analysis

Abstract

This chapter presents the transient analysis of the assembly system. The first section focuses on Cinecolo, a module developed by Tractebel, which was used to solve the point kinetics equations in order to study the system's transient behavior following a reactivity insertion. The primary objective in this section is to verify the accuracy of the Cinecolo code by comparing its results with the analytical solution of the point kinetics equations and with the results obtained from Serpent. The second section covers the transient analysis in Serpent, focusing on the initialization of the transient simulation, followed by a detailed presentation and analysis of the simulation results.

4.1. Cinecolo: Point Kinetics Equations

In this section, the approach for studying the transient behavior of the system using Cinecolo is discussed. Cinecolo is a module that solves the point kinetics equations, using the kinetic parameters extracted from Serpent in criticality simulations. The module is capable of simulating the reactor's response to changes in reactivity by solving the point kinetics equations, with six precursor groups. The main focus of this section is on the initial attempts to model the transient behavior of the system following a reactivity insertion, without incorporating thermodynamic feedback. This step represents the first phase toward developing more detailed simulations that will include such feedback in future stages.

Introduction to Cinecolo module

Cinecolo is a code developed by Tractebel, originally written in Pascal and later translated into Python. The code uses the Kaganove's method to solve the point kinetics equations by discretizing the time of the transient into steps. In this method, the reactor reactivity function is approximated with a parabolic evolution over each time step. The reactivity function $p(t)$ is fitted to this parabola by matching the values of the function at three

distinct points within the time step: the beginning, the middle, and the end. Using this approach, the reactor's kinetic quantities, such as the neutron flux and precursor concentrations, are calculated at each step. These values are then used as the initial conditions for the next time step, iterating until the transient behavior is fully simulated. The method ensures that the reactor's reactivity is accurately tracked over time and allows for detailed simulations of the system's response to changes in reactivity. This method is explained in more detail in the literature [5].

For running the Cinecolo code, several input parameters are required, including the initial step, maximum step, final time, minimum accuracy, maximum accuracy, the minimum and maximum variance, and the maximum number of steps. An example of these inputs is shown below:

```
Enter Initial step: 0.001
Enter Maximum step: 1
Enter Final time: 10800
Enter Accuracy (epsmin): 1e-6
Enter Accuracy (epsmax): 1e-5
Enter Accuracy (varmin): 0.05
Enter Accuracy (varmax): 0.25
Enter maximum number of steps: 9999
```

These parameters define how the time of the transient simulation is discretized. The Kaganove's method, as described in the literature, refines the time step iteratively to satisfy the input accuracy requirements. This process starts with the initial step specified by the user, and the time step is further adjusted as the simulation progresses to meet the desired accuracy, up to the specified number of iterations.

Moreover, for Cinecolo to run correctly, a .DAT file must be provided containing the system's kinetic parameters: the neutron generation time (Λ) in the first line, the delayed neutron fractions (β_i) for each group in the second line, the decay constants (λ_i) in the third one, and the reactivity insertion, which can be a step or ramp, specifiying in the first column the time instant and in the second column the corresponding reactivity. An example of the .DAT file format for a step reactivity inserion of 535.12 pcm, and of a ramp reactivity insertion are shown below:

```
Cinecolo - steps de reactivite - 6 groupes
--------
28.5869e-6
0.0133567 0.0325950 0.1210990 0.3061150 0.8630210 2.899980
```

```
0.000216789 0.00120478 0.00116502 0.00272011 0.00124575 0.000518655
0          0e-5
1e-5     +535.12e-5
10       +535.12e-5

Cinecolo - rampe de reactivite - 6 groupes
--------
28.5869e-6
0.0133567 0.0325950 0.1210990 0.3061150 0.8630210 2.899980
0.000216789 0.00120478 0.00116502 0.00272011 0.00124575 0.000518655
    0            0e-5
   10      +6.6408e-4
   20            0e-5
   30            0e-5
```

This file contains the necessary data to describe the kinetics of the system and the reactivity step or ramp insertion, which will be processed by Cinecolo to compute the transient behavior.

The Kaganove's method is particularly well suited for this type of simulation because of its ability to handle reactivity transients accurately, especially for systems without significant feedback effects.

4.1.1. Verification of Cinecolo Using Numerical and Analytical Solution

In this part of the thesis, the Cinecolo code was verified using both numerical and analytical solutions. The primary reason for this verification was largely driven by company objectives, as Cinecolo is intended to be developed further in the future. Ensuring the accuracy and reliability of the code was essential to provide a solid foundation for its future development, especially considering the inclusion of thermal feedbacks. These feedbacks are crucial for simulating the dynamic behavior of the reactor under varying operational conditions, which is why a thorough verification was a necessary step for both the company's goals and the code's successful application. To verify the accuracy of the Cinecolo code, a numerical solution was implemented using the Runge-Kutta method. Additionally, the analytical solution was also implemented, which is mathematically correct and serves to validate the results obtained from Cinecolo. Initially, a low reactivity insertion transient (equal to 0.2β, approximately 128 pcm) was considered. This allows for the study of the system using the prompt jump (PJ) approximation.

Point Kinetics Equations and Simplification to One Effective Delayed Group

The point kinetics equations are traditionally solved using multiple precursor groups, each associated with a specific decay constant λ_i and delayed neutron fraction β_i. However, these equations can be simplified by combining the six precursor groups into one effective delayed group. This simplification involves summing the delayed neutron fractions for the six groups to define an effective delayed neutron fraction β:

$$\beta = \sum_{i=1}^{6} \beta_i.$$

Similarly, the decay constant for the effective delayed group, λ, is calculated using a weighted average of the individual λ_i values for the groups:

$$\lambda = \frac{\sum_{i=1}^{6} \lambda_i \beta_i}{\beta}.$$

This single group approximation is useful in reducing the complexity of the equations, especially when studying transient behavior with limited computational resources.

Exact Solution of the Point Kinetics Equations

The exact solution of the point kinetics equations, using one effective delayed group, is based on solving the system of ordinary differential equations for the neutron population $n(t)$ and precursor concentration $C(t)$. In the case of a step reactivity insertion, where the reactivity is constant over time, the general form of the solution is derived by substituting the exponential form into the one effective delayed group point kinetics equations. These equations take the form of:

$$\frac{dn}{dt} = \frac{\rho - \beta}{\Lambda} n + \lambda C,$$

The solution for the neutron population can be expressed as:

$$n(t) = n_2 e^{s_2 t} + n_1 e^{s_1 t},$$

where s_2 represents the prompt neutron lifetime and is large, corresponding to the fast

response of the reactor, while s_1 is the time constant for the delayed neutrons, which is small, describing the slow buildup of the neutron flux over time due to delayed neutrons.

The prompt neutron lifetime is associated with the prompt response to reactivity changes and governs the rapid exponential rise of the neutron population at the beginning of the transient. The delayed neutron contribution becomes significant after some time, contributing to a slower increase in neutron population.

Prompt Jump Approximation

In the response of the reactor power to a step change in reactivity, there is initially a very rapid transient behaviour on a time-scale characteristic of the prompt neutron lifetime, followed by a more slowly varying response governed by delayed neutron behaviour. Since this transient is so rapid, a very useful approximation to make for systems below prompt critical is one in which the prompt neutron lifetime is essentially taken to be zero such that the power level jumps instantaneously to its asymptotic behaviour. This so-called prompt jump approximation is effected by merely **neglecting the time derivative $\frac{dn}{dt}$ in the point reactor kinetics equation**. Mathematically, the assumption is that:

$$0 = \frac{\rho - \beta}{\Lambda} n + \sum_i \lambda_i C_i \tag{4.1}$$

Since the delayed neutron production cannot respond immediately to a step change in reactivity, this model predicts that a reactivity jump from ρ_0 to ρ_1 causes an instantaneous change in reactor power from n_0 to n_1 as given by:

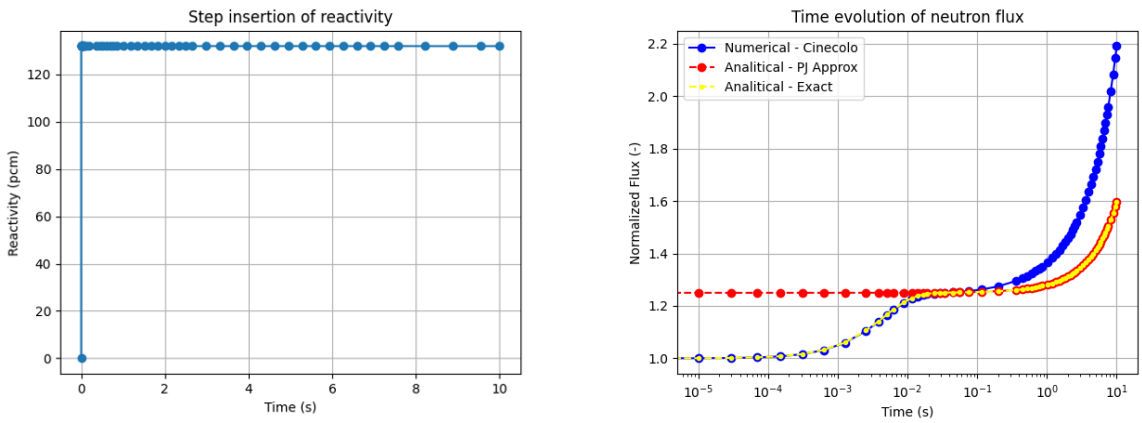
$$\frac{\rho_0 - \beta}{\Lambda} n_0 = \frac{\rho_1 - \beta}{\Lambda} n_1 \tag{4.2}$$

$$\frac{n_1}{n_0} = \frac{\beta - \rho_0}{\beta - \rho_1} \tag{4.3}$$

The PJ approximation simplifies the problem significantly and is accurate when the system is far from prompt critical. However, it fails to capture the initial behavior of the flux as accurately as the exact solution, which accounts for the initial time derivative of the flux.

Comparison of PJ Approximation and Exact Solution

Both the exact solution and the PJ approximation were initially modeled using a single precursor group, representing the average of the six precursor groups. The comparison of the Cinecolo results with these solutions is shown in the plot below.



(a) Step insertion of reactivity (128 pcm).

(b) Comparison of Cinecolo results with the analytical solutions: PJ approximation and exact solution using a single precursor group.

Figure 4.1: Step insertion of reactivity and Comparison of Cinecolo with Analitical solutions results.

As seen in the plot, the PJ approximation does not match the exact solution at the beginning of the transient, as it neglects the initial time derivative of the neutron flux. This leads to a discrepancy in the initial part of the transient. The exact solution, on the other hand, accounts for both prompt and delayed neutrons, accurately capturing the initial response and then transitioning to the slower delayed neutron response.

However, both analytical solutions (approximate and exact) fail to fully represent the flux behavior calculated by Cinecolo, as they model the system using a single precursor group. This simplification leads to an underestimation of the flux increase over time, particularly at later times, because the contribution of longer-lived precursors is not adequately accounted for.

Six Precursors Groups Analytical and Numerical Solution

The next step was to develop the analytical solution, where the PJ approximation was no longer considered, as it would not be suitable for analyzing higher reactivity insertions in subsequent transient simulations. Instead, the focus shifted to the exact solution, which was further improved by considering a six-group approximation. This approach allows a better representation of the transient behavior over longer timescales.

Developing an analytical solution with six precursor groups is not trivial, as it requires solving a system of seven equations with seven unknowns: the neutron population $n(t)$ and the six precursor concentrations $C_i(t)$. To solve this system, a method described in the literature [23] was used, based on the following solution process:

1. Laplace Transformations: To solve the system, Laplace transformations are applied to the point kinetics equations. The starting equations are:

$$s\hat{n}(s) - n(0) = \frac{\delta k_{\text{eff}}}{L_{\text{tot}}}\hat{n}(s) + \sum_{i=1}^{6} \lambda_i \hat{C}_i(s)$$

$$s\hat{C}_i(s) - C_i(0) = -\lambda_i \hat{C}_i(s) + \frac{k_{\text{eff}}\beta_i}{L_{\text{tot}}}\hat{n}(s), \quad i = 1, \ldots, 6$$

2. Elimination of $\hat{C}_i(s)$: After applying transformations and simplifying the equations, we can express the relationships for $\hat{n}(s)$ (the Laplace-transformed neutron population) and $\hat{C}_i(s)$ (the Laplace-transformed precursor concentrations):

$$\hat{n}(s) = \frac{n(0) + \sum_{i=1}^{6} \frac{\lambda_i C_i(0)}{s+\lambda_i}}{s - \frac{\delta k_{\text{eff}}}{L_{\text{tot}}} - \sum_{i=1}^{6} \frac{\lambda_i}{s+\lambda_i} \frac{k_{\text{eff}}\beta_i}{L_{\text{tot}}}}$$

3. Identification of Poles: The poles of the function $\hat{n}(s)$ correspond to the roots of the equation, which are real values representing the time constants for both prompt and delayed neutrons. These poles allow us to compute the exact time evolution of the neutron population.

4. Final Solution: The final solution for the neutron population is expressed as a rational function in the complex variable s, where the exact time dependence of the neutron flux is given by:

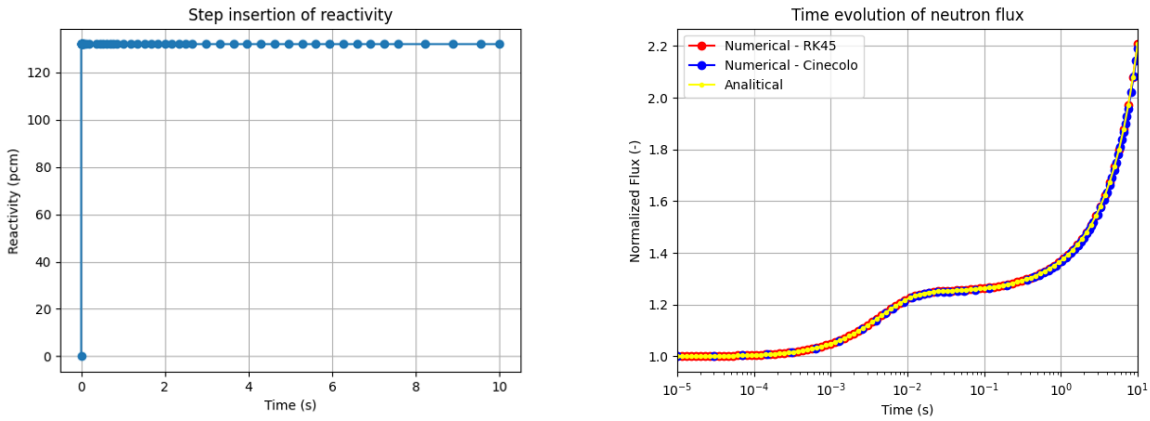
$$n(t) = \sum_{j=1}^{7} A_j \exp(s_j t), \quad t \geq 0$$

where the coefficients A_j (which describes the time evolution of the neutron population) are determined by the poles in both the numerator and the denominator of the rational function, and the time constants s_j describe the temporal evolution of the neutron population.

This solution allows a full representation of the neutron flux, considering both the prompt and delayed neutron contributions.

The next step was to implement a numerical solution to solve the same system of equations, but using the Runge-Kutta method. This numerical solution was compared to both the analytical solution and the Cinecolo results to verify its accuracy.

This numerical solution was also found to be in agreement with the Cinecolo results, as shown in the plot below. The same transient was simulated with a reactivity insertion of approximately 128 pcm, and both the analytical solution and the numerical solution were compared to the Cinecolo results.



(a) Step insertion of reactivity (128 pcm).

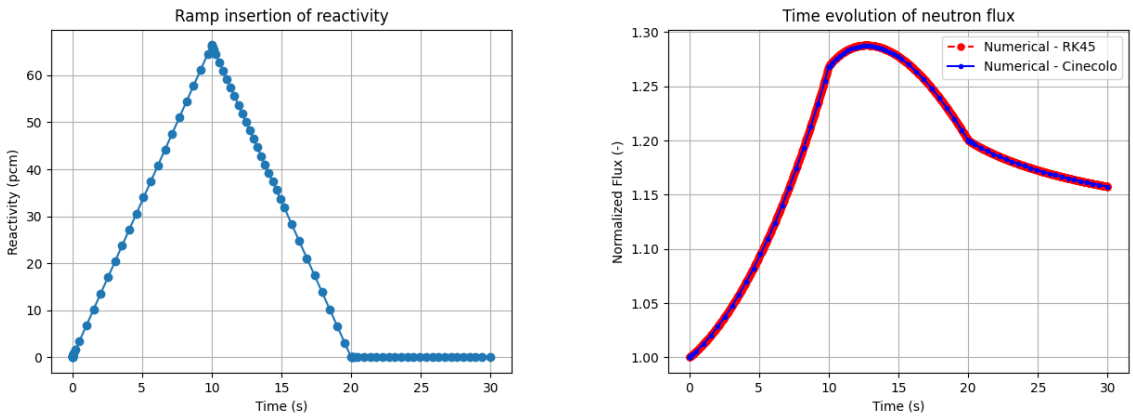
(b) Comparison of the Cinecolo results with the Analytical and Numerical solutions.

Figure 4.2: Step insertion of reactivity and Comparison of Cinecolo with Analitical and Numerical solutions results.

The numerical solution, using the Runge-Kutta method, effectively models the system and provides results that match the Cinecolo simulations, as well as the exact analytical solution. This numerical solution will serve as the basis for implementing the code to

include thermodynamic feedback and compare the results of Serpent coupled with fuel behavior.

Below are the results obtained with Cinecolo for the transient involving a reactivity ramp, compared with the numerical solution using the Runge-Kutta method. As can be observed, the results are in good agreement.



(a) Ramp insertion of reactivity.

(b) Comparison of the Cinecolo results with the Numerical solution.

Figure 4.3: Ramp insertion of reactivity and Comparison of Cinecolo with Numerical solution result.

In the next section, we will focus on the transient simulations performed using Serpent, where the kinetic parameters extracted from criticality simulations are utilized in a more advanced setup for transient behavior analysis.

4.2. Transient Analysis with Serpent

This section covers the implementation of transient simulations in Serpent, detailing the steps taken to initialize the simulation and the results obtained. The discussion focuses on the analysis of neutron flux, power distribution, and the changes observed throughout the transient period.

4.2.1. Development of a Dynamic Simulation Mode in Serpent 2 Monte Carlo Code

This subsection will provide an overview of the implementation of a time-dependent (dynamic) neutron transport mode in the development version of the code, Serpent 2. The

methods are based on a standard external source simulation, with sequential population control for the modeling of super-critical systems, for which there is only a time dependent solution for the flux: there is no time independent, steady state solution [25]. Some theoretical background on time-dependent simulations is also given, before introducing the implemented methodology in the next subsections.

Time Dependence in k-eigenvalue Criticality Source Simulation Neutron transport simulation in the criticality source mode is run in cycles, or generations, in which the source distribution is iteratively regenerated from the fission neutron distribution of the previous cycle. The iteration is started from an initial guess and the average population size is maintained either by scaling the source term with the criticality eigenvalue, or by applying population control on the banked source neutrons before the next cycle is run. Since the simulation proceeds from one fission event to the next, neutron histories are not bound in time, which in turn implicates that flux and reaction rates are assumed to remain constant over any given time interval. The simulation mode essentially corresponds to solving the k-eigenvalue form of the static transport equation:

$$\hat{\Phi} \cdot \nabla + \Sigma_{\text{tot}} \Phi = S + \frac{1}{k} F, \tag{4.4}$$

where the scattering and fission source terms are written in simplified form and all variables are omitted. The k-eigenvalue criticality source method always simulates the chain reaction without time dependence. If the system is not inherently in a steady state condition (i.e. $k \neq 1$), it is forced into one by adjusting the fission source [16].

Time Dependence in External Source Simulation All neutron histories in an external source simulation are started from the same user-defined source distribution, which fixes the initial position, direction, energy, weight, and time of each starting neutron. Since the histories are not batched into generations, the simulation can be divided into explicit time intervals. In deterministic transport theory, the simulation mode corresponds to solving the time-dependent form of the transport equation:

$$\frac{1}{v} \frac{\partial}{\partial t} \Phi + \hat{\Phi} \cdot \nabla \Phi + \Sigma_{\text{tot}} \Phi = S + F + G, \tag{4.5}$$

where G is the external source. There is no population control applied to neutron histories, and fission reactions are explicitly simulated by branching the history. All histories eventually die out in sub-critical systems, but in the case of super-criticality, the histories tend to develop an infinite number of branches. This, in turn, means that time cut-

off must be enforced to terminate the simulation. Since the growth rate is exponential, the simulation of prompt super-critical reactivity excursions is practically limited to very short periods of time [16].

4.2.2. Basic approach in the time-dependent simulations

Serpent implements time-dependent simulations by tracking both neutrons and precursors continuously, without the need for any time-discretization. The simulation starts with the initial neutron and precursor sources, which are obtained from the source generation simulation, as explained in the previous chapter. The transient simulation, indeed, is conducted in two steps, as described in the figure below.

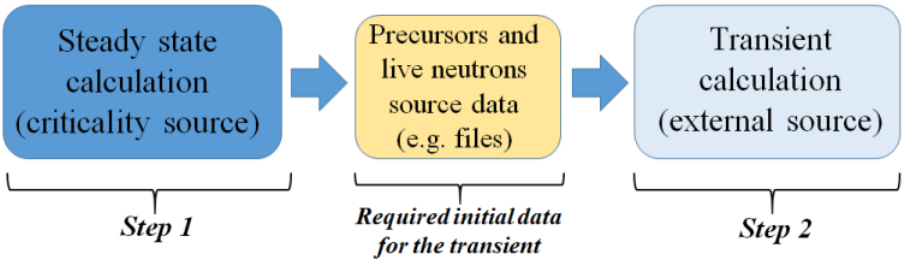


Figure 4.4: Two-step approach for transient calculations [7].

The first step involves a steady-state calculation (criticality source), while the second step handles the transient calculation using external sources, such as files containing the initial distributions of live neutrons and delayed neutron precursors. Moreover, in the transient analysis, the simulation can be divided into sub-intervals for population control purposes. In the simplest case, the transient can be simulated as a single time interval.

The basic process for each time interval is as follows:

1. **Create the primary source for the time interval:** Calculate the live neutron population at the beginning of the interval. This can either be based on the initial neutron source or from the current neutron banks.

2. **Calculate precursor decay:** The decay of precursor tallies is calculated based on the decay law, which determines the delayed neutron population emitted during the interval.

3. **Store initial values:** At the end of the interval, the initial value of each tally is stored.

4. **Divide the initial source:** The initial source points are divided between live

neutrons and delayed neutrons emitted from precursors, based on the proportion of the two populations.

5. **Population control:** The live neutron source is controlled to the required number.
6. **Sample delayed neutrons:** The required number of delayed neutrons is sampled from existing precursors.
7. **Simulate particle emission:** The primary particles, as well as any secondary particles emitted during the time interval, are simulated.
8. **Tally precursor production:** Precursor production is tallied during the neutron tracking.
9. **Store end-of-interval data:** Neutrons that reach the end of the time interval are stored, and the precursors that are produced and survive until the end of the interval are added to the end-of-interval precursor tallies.

The process then restarts for the new precursors produced during the time interval, with only a portion of their weight surviving until the end of the time interval due to decay that occurs during the interval.

A schematic plot of precursor production during a time interval of a transient simulation can be seen below.

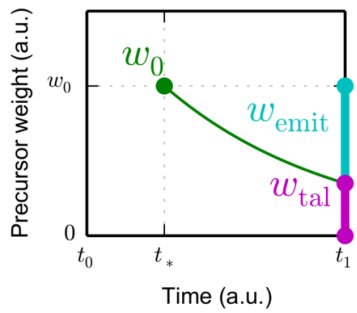


Figure 4.5: Schematic representation of precursor weight evolution during a time interval in a transient simulation [14].

The initial produced precursor weight in an interaction occurring at time t_* is denoted by w_0. Only a fraction of the weight survives until the end of the interval, and it is added to the end-of-interval tallies:

$$w_{\text{tal}} = w_0 e^{-\lambda_g (t_* - t_0)}$$

The remaining portion decays during the interval and is emitted as a delayed neutron:

$$w_{\text{emit}} = w_0 \left(1 - e^{-\lambda_g (t_* - t_0)}\right)$$

As w_{emit} is typically small compared to w_0, the delayed neutron to be emitted is subjected to Russian roulette, either to increase its weight to a more reasonable level or to not emit it at all. Finally, it is important to note that in non-coupled transient simulations the simulation proceeds by first simulating the first neutron batch through all of the time-intervals before simulating the next batch.

4.2.3. Input for Transient Simulations

To run a time-dependent simulation in Serpent, some small modifications must be made to the criticality source input. These changes enable the model to track both neutrons and precursor populations throughout the transient period. The key modifications are as follows:

- Time Binning: The time binning for the simulation is defined using the `tme` card. This allows the simulation to track the changes in neutron flux and precursor concentrations over specific time intervals. For example, in the case of the transient simulation in this work, the following input was used:

  ```
  % --- Set up time structures for simulation and detectors:
  tme simutime 2 100 0 0.1
  ```

  Here, the values correspond to the following:

  `simutime` : The identifier for the time binning structure.

  `2` : The binning type (2 = uniform binning).

  `100` : The number of bins.

  `0` : The minimum time boundary in seconds.

  `0.1` : The maximum time boundary in seconds.

  In this example, a uniform binning structure is defined with 100 time bins, ranging from 0 to 0.1 seconds. This means that the time steps will be uniformly distributed between these two values.

  The time binning card can be customized using several options [13]:

`tme NAME 1 LIM1 LIM2 ...` : This defines an arbitrary binning, where specific time boundaries can be set for each bin. The first limit in the arbitrary type is the lower bound of the first bin, and the second limit is the upper bound of the first bin, and so on.

`tme NAME 2 NB Tmin Tmax` : This defines a uniform binning, where the time boundaries are set between a minimum (`Tmin`) and maximum (`Tmax`) value.

`tme NAME 3 NB Tmin Tmax` : This defines a log-uniform binning, where the time boundaries are logarithmically spaced between `Tmin` and `Tmax`.

Finally, two additional notes regarding this card should be mentioned. First, if only a single time step is specified instead of multiple time steps, population control will not be performed at each time step, during the simulation. Second, a different time step structure can be defined for the detectors compared to that of the simulation. For example, one might use:

```
% --- Set up time structures for detectors (1000 time bins):
tme dettime 2 1000 0 0.1
```

- Population Settings: The `set nps` option should be used to set the simulated population for the transient simulation, specifying the number of particles to be simulated within each time bin. As an example, the following set-up was used for the transient simulations:

  ```
  % --- Neutron population:
  set nps 1200000 600 simutime
  ```

  where:

  - `1200000` refers to the total number of particles to be simulated.
  - `600` refers to the number of batches, with a default value of 200 if not specified.
  - `simutime` corresponds to the time binning definition, which was set earlier in the simulation.

  The total number of particles is divided by the number of batches, and particles are sampled for each time interval when an initial transient source is linked using the `set dynsrc` option. In the case of multiplying systems, the neutron and precursors buffers might need to be adjusted using the `set nbuf` and `set pbuf` option to ensure the simulation runs efficiently, as follows:

  ```
  % --- Increase size of precursor buffer
  ```

```
set pbuf 10000

% --- Increase size of neutron buffer
set nbuf 20000
```

- Linking the Steady-State Source: A previously generated steady-state source must be linked using the `set dynsrc` option. This links the source files, which will be used as the starting point for the transient simulation.

- Delayed Neutron Source: To set up the delayed neutrons, the previously generated source files are linked with the `set dynsrc PATH [ MODE ]` input option. Here, the parameters are:

  - PATH: The path of the previously generated source file, excluding the `.main` suffix. - MODE: Specifies the precursor tracking mode, where 0 indicates mesh-based tracking and 1 indicates point-wise tracking.

  The four source files required for the transient simulation are `[PATH].main`, `[PATH].prec`, `[PATH].live`, and `[PATH].precpoints`, all of which should be located in the specified path.

  For example, this input was used:

  ```
  % --- Link source (use point-wise precursor tracking)
  set dynsrc "./source" 1
  ```

- Set-up of detectors: To set up time-dependent detectors, it is necessary to specify, after the usual parameters (det NAME) and the type of reaction (`dr MT MAT`) the Detector time binning (`di TBIN`), where `TBIN` is the time binning structure name, which determines how the results are divided into multiple time bins.

  For example, the following input sets up detectors for neutron population and fission energy deposition as a function of time:

  ```
  % --- Neutron population as a function of time
  det 1 dr -15 void di dettime

  % --- Fission energy deposition as a function of time
  det 2 dr -8 void di dettime
  ```

  Here, `dettime` refers to the time binning structure defined earlier in the simulation.

These adjustments allow Serpent to track the time evolution of neutron populations and

precursor concentrations, simulating the effects of reactivity insertions and their impact on the system's behavior over time.

Before starting the time-dependent simulation, it is possible to modify the geometry and material compositions of the system to introduce perturbations. These modifications can help simulate different scenarios, such as variations in the system configuration or changes in material properties. However, it is important to note that the neutron and precursor source distributions will be overlaid onto the new geometry without any further modifications. In the case of the transient simulated in this work, the material composition was modified by adjusting the concentration of boron in the water in order to achieve the desired reactivity insertion, as explained in the previous chapter, or the geometry was modified by raising the control rods using the card `trans`.

Additionally, the results of the time-dependent simulation, including the neutron and precursor sources, can be saved using the `set savesrc` option. To avoid overwriting the original source distribution, a different file-path should be specified when saving the updated source. If the source is linked to a previous time-dependent simulation, the time binning should begin from the end time of the previous simulation.

It is also important to consider that the physical results obtained from time-dependent simulations, such as detector outputs, are integrated with respect to time. For example, detector response -8 (fission power deposition) will yield the fission energy deposition in joules rather than watts. Therefore, for post-processing the results, it is necessary to divide the values by the time of each time step to obtain the proper time-dependent quantities.

4.2.4. Population Control in Dynamic Method for Transient Analysis in Monte Carlo Simulations

Population control is a technique used in Monte Carlo simulations to manage the neutron population during criticality and transient calculations. In the case of a super-critical system, the neutron population can grow rapidly, potentially exceeding the capacity of the computer to handle the data. For example, if we start a time interval with 10,000 neutrons and, by the end of the time interval, 15,000 neutrons are present, we must control this excess population to prevent computational overload. This is where population control, specifically combing, comes into play. The process involves reducing the population back to the desired level by discarding the excess neutrons, ensuring that the simulation continues with the same number of neutrons (in this case, 10,000). This process is akin to performing a time-dependent Russian roulette, where neutrons are randomly discarded

to maintain a balanced population without introducing any bias.

In sub-critical systems, the issue is the opposite: the neutron population may decrease to zero over time. Population control in these systems ensures that there are always enough neutrons to keep the calculation running by either adding new neutrons (through neutron splitting) or adjusting the existing population (through Russian roulette). It is important to note that the disappearance or creation of neutrons during this process does not correspond to actual neutron absorption or production, but is a result of importance sampling used to control the population.

This report [25] provides a clear explanation of the differences between existing methods for Monte Carlo simulations in criticality analysis. An important objective of this paper is to determine whether or not criticality methods produce the same answers as a true analog calculation. In particular, the methods discussed include k-static, alpha-static, and the dynamic method. Each method handles the neutron population in a different way, which affects the accuracy of the simulation results.

The dynamic method adjusts the neutron population at the end of each time step. Neutrons are distributed uniformly in space, energy, and time by a factor of $\exp(\alpha t)$, which brings the neutron population back to its initial value at the start of the time step. This method ensures that the sample remains unbiased, allowing for accurate results in subsequent time intervals. As shown in [26], comparisons with true analog calculations without combing demonstrate that the dynamic method with combing provides exactly the same results, making it the correct approach for transient analysis.

The k-static method adjusts the neutron population at each time step to match the initial count, solving a different problem than that posed in transient analysis. Although computationally efficient, it introduces biases—particularly in super-critical systems—by softening the neutron spectrum and shifting the flux away from fissile materials. These inaccuracies grow as the system deviates further from criticality, making the method progressively less suitable for accurate transient simulations compared to dynamic approaches.

The alpha-static method combines neutron absorption over each time step, similar to the dynamic method. It adjusts the neutron population uniformly in space, energy, and time by the factor $\exp(\alpha t)$, ensuring that the number of neutrons at the end of the time step matches the number at the start. This method is mathematically equivalent to the dynamic method, which is why their results agree. However, it is important to note that the alpha-static method is suitable for systems that are close to criticality. As the system moves further from criticality, the method requires more adjustments, leading to larger

discrepancies between the alpha-static results and the true solution.

The dynamic method is the most appropriate for simulating transient behavior in Monte Carlo simulations, as it accurately models the evolution of the neutron population over time. The implementation of this method in Serpent 2, as detailed in [16], ensures that these issues are addressed, providing a more accurate representation of the reactor's behavior during transients. In the following sections, we will discuss the key concepts of the dynamic method, with further details available in the original reports ([26]).

In Monte Carlo simulations for nuclear reactor transient analysis, delayed neutrons and their precursors play a critical role. Unlike static calculations, where all fission neutrons are considered to be released instantly (prompt neutrons), the dynamic behavior of neutrons includes both prompt and delayed neutrons. Prompt neutrons are released almost instantaneously, within a time scale of approximately 10^{-14} seconds. However, precursors, which are responsible for delayed neutrons, have much longer lifetimes, ranging from 10^{-4} seconds for a light water reactor to as long as 10^{2} seconds for certain types of reactors.

Role of Precursors in Monte Carlo Simulations

Precursors are isotopes that decay into delayed neutrons. Their long lifetime, compared to the neutron lifetime, allows them to play a key role in controlling the reactor using mechanical control rods. For Monte Carlo simulations, the difference in the time scales of neutrons and precursors poses a significant challenge. While neutrons have lifetimes on the order of 10^{-4} seconds in a light water reactor, precursors can persist for much longer, up to hundreds of seconds.

To model this effectively in a Monte Carlo simulation, the time is divided into discrete intervals, with each precursor forced to produce a weighted delayed neutron in every time interval. This approach ensures that a prompt fission chain is initiated in each time interval, even if no prompt neutrons are generated within that period. The probability of a precursor decaying at any given time is uniformly distributed across each time interval. This method guarantees that there are always fission chains present in every time interval, preventing long periods without neutron generation.

Population Control for Precursors

Population control in the context of precursors differs from that applied to neutrons. When simulating precursors, there are three main statistical weights to consider:

- The main statistical weight of the precursor, denoted as w_C, represents the num-

ber of physical precursors that the Monte Carlo precursor corresponds to at the time of creation. This weight is subject to adjustment through variance reduction techniques but remains constant over time.

- The expected delayed neutron weight represents the interaction of the precursor with the system via the delayed neutrons it produces.
- The timed statistical weight is related to the timing of the precursor's decay within the time interval.

The main weight w_C is the weight that can be adjusted when applying population control. Variance reduction techniques, such as weight windows, are often used to ensure that all particles contribute similarly to the simulation.

Variance Reduction Techniques for Precursors

To control the population of precursors during the simulation, several techniques can be used:

- Weight windows: This technique keeps the weights of different particles within the same range, ensuring that all particles have a similar contribution. The weight of the precursor, either timed or delayed neutron, is monitored to maintain a consistent statistical weight.
- Russian roulette: This method is applied when the weight of the delayed neutron falls below a certain threshold. The probability of survival for the precursor is calculated, and if the precursor survives, its statistical weight is updated accordingly.
- Splitting: If the expected delayed neutron weight exceeds a predefined threshold, the precursor can be split into multiple parts, each with an equal share of the original weight. This ensures that the precursor population remains within manageable bounds.
- Combing: A more advanced technique, combing involves redistributing the total weight of all particles over a fixed number of particles with equal weights. This ensures that the total precursor concentration is conserved while adjusting the number of particles.

Implementation of the Dynamic Method

The dynamic Monte Carlo method for transient analysis involves dividing the time domain into intervals and simulating each interval consecutively, as schematically represented in

Figure 4.6.

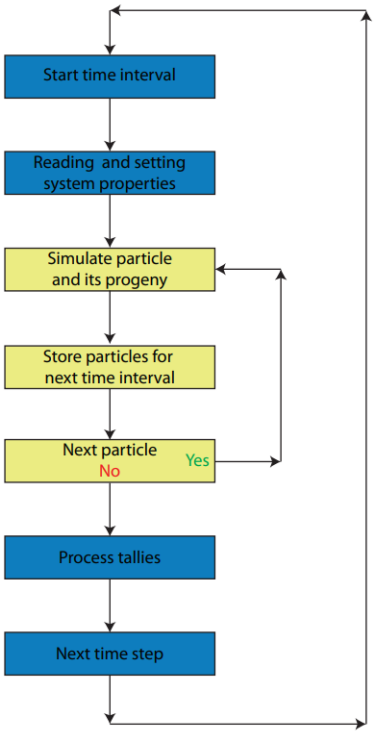


Figure 4.6: The scheme for performing a dynamic Monte Carlo calculation [26].

In each interval, both precursors and neutrons are simulated, and the particles are followed until they are either absorbed, leak out, or cross the time boundary. At each boundary, the particles are stored for the next interval, and any decay processes are forced to occur within the time remaining.

The statistical weight of the resulting neutrons from precursor decay is calculated based on the time interval in which the precursor is created. This ensures that even if the weight of the delayed neutron is low, the particles are still simulated to maintain unbiased results. Russian roulette is used to adjust the survival probability of the particles, and weight windows are applied to maintain a consistent statistical distribution across time intervals.

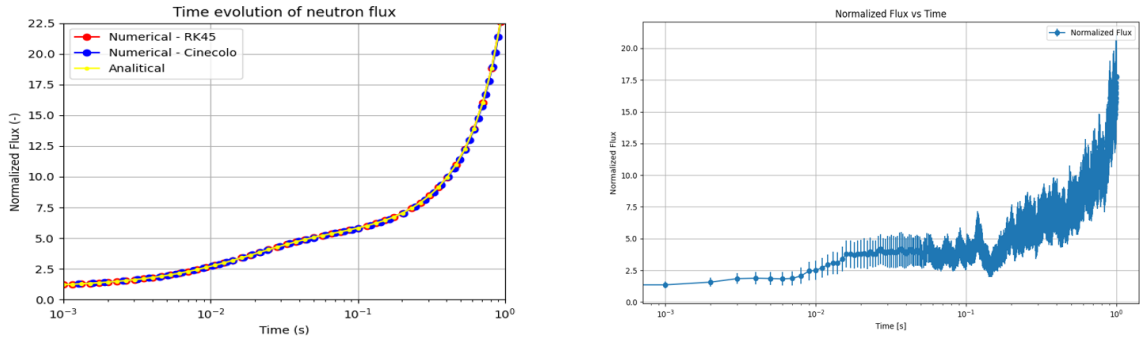
4.3. Results

A first test was carried out to simulate a 1-second transient initiated by a step reactivity insertion of 535 pcm, starting from a critical system in steady state conditions. The fuel temperature in these uncoupled simulations was kept constant at 900 K, since thermal feedback effects were not yet considered.

The boron concentration of the system in both critical and transient configurations, along with other relevant simulation parameters—such as the number of simulated neutrons, computational time, control rod position, and reactivity insertion—are summarized in Table 5.9. The evolution of the neutron flux following the 535 pcm reactivity insertion is presented in Figure 4.7. The left graph (4.7a) shows the time-dependent neutron flux computed using analytical and numerical methods (Cinecolo and Runge-Kutta), while the right graph (4.7b) displays the neutron flux behavior obtained from the Serpent simulation.

Simulation Parameter	**Value**
Simulated Neutrons	1M
Number of batches	100
Number of time bins	1000
Fuel Temperature	900 K (fixed)
Control Rod Position	Raised of 25 steps
Reactivity Insertion	535 pcm (step)
Simulation Type	Uncoupled (no thermal feedback)
Transient Duration	1 s
Configuration	**Boron Concentration (ppm)**
Critical (steady-state)	468,14
Transient (post-insertion)	426,04

Table 4.1: Transient simulation parameters – 535 pcm step reactivity insertion



(a) Analytical and numerical solutions (Cinecolo, Runge-Kutta).

(b) Serpent Monte Carlo simulation result.

Figure 4.7: Comparison of neutron flux evolution following the 535 pcm reactivity insertion.

As observed in the results, the neutron flux behavior predicted by Serpent aligns well with both the analytical and numerical solutions, following a similar overall trend during the transient. However, the Serpent simulation still shows a significant amount of statistical noise, as evidenced by the large uncertainty bars and visible oscillations in the flux evolution. These fluctuations are typical of Monte Carlo methods, particularly in time-dependent simulations. While the Monte Carlo method is highly accurate in certain scenarios, its computational cost is notably high, especially in comparison with the numerical and analytical methods. The substantial amount of computational resources required to perform simulations with Serpent results in solutions characterized by high uncertainty and low precision. This inefficiency becomes even more apparent when considering analyses that prioritize speed and require faster results. Therefore, despite the potential accuracy of Monte Carlo simulations, their computational burden and the resulting lack of precision make them less attractive for applications that demand rapid calculations and higher precision. Further improvements in statistics, such as increasing the number of simulated neutrons or cycles, could enhance the clarity and reliability of the results, but would likely further increase the already substantial computational cost.

As a next step, the focus was placed on addressing the limitations observed in the initial Serpent simulation, particularly the high level of statistical noise and the large uncertainty associated with the flux evolution. To improve the accuracy and reduce the oscillatory behavior, some modifications were introduced. First, the number of simulated neutrons as well as the number of neutron batches were increased to enhance statistical reliability. In addition, population control was implemented. This technique, which regulates the neutron population across time steps by redistributing or replicating neutron histories, helps

to reduce variance and smoothen the transient response. Although the use of population control led to an increase in computational time, it significantly improved the quality of the results, which now exhibit reduced noise and more consistent behavior.

The updated simulation results are shown in Figure 4.8. In this case, the transient duration was shortened to 0.1 s to limit computational burden, while still capturing the essential dynamic behavior. The key simulation parameters for this refined case are summarized in Table 4.2.

Table 4.2: Simulation parameters – 0.1 s transient with population control.

Simulation Parameter	**Value**
Simulated Neutrons	1.2M
CPU Time (with pop control)	90d 1h
CPU Time (without pop control)	18d 15h
Number of batches	6000
Number of time bins	100
Fuel Temperature	900 K (fixed)
Control Rod Position	Raised of 25 steps
Reactivity Insertion	535 pcm
Simulation Type	Uncoupled (no thermal feedback)
Transient Duration	0.1 s
Configuration	**Boron Concentration (ppm)**
Critical (steady-state)	468,14
Transient (post-insertion)	426.04

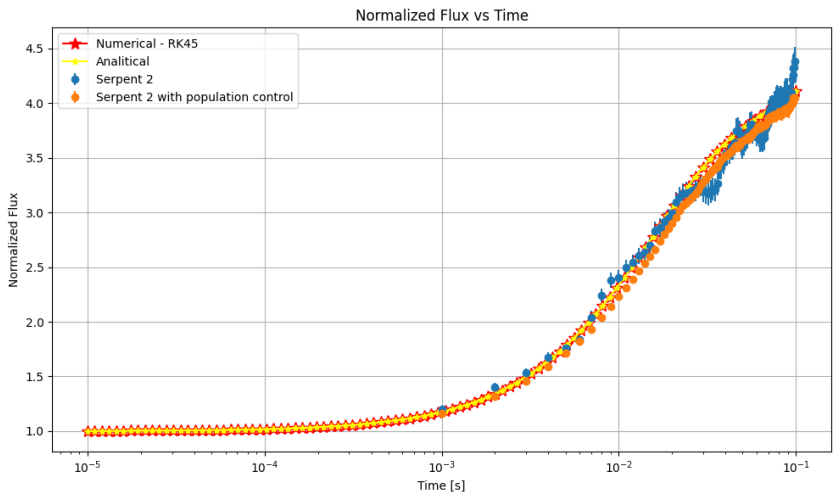


Figure 4.8: Neutron flux evolution during the 0.1 s transient with and without population control enabled.

The results confirm that the Serpent simulations are in overall agreement with both the analytical and numerical solutions. In particular, the orange curve, corresponding to the Serpent simulation with population control, closely follows the expected transient behavior and aligns well with the reference solutions. This improved agreement highlights the effectiveness of population control in reducing statistical fluctuations and uncertainty. On the other hand, the blue curve, representing the simulation without population control, exhibits noticeable oscillations and noise, which obscure the underlying physical trend. The application of population control thus plays a key role in enhancing the reliability and clarity of the Monte Carlo transient results. However, this improvement comes at the cost of significantly increased computational time, which can make the Monte Carlo approach less attractive for routine or large-scale transient analyses where computational efficiency is a critical factor.

Conclusion

This chapter introduced the transient analysis framework by presenting a first test case based on Cinecolo, a numerical code developed by Tractebel that solves the point kinetics equations. The results obtained with Cinecolo were successfully verified through the implementation of an analytical solution. In addition, a numerical method based on the Runge-Kutta scheme was developed and showed consistent agreement with Cinecolo. This numerical approach will serve as the foundation for the implementation of an extended model that includes thermodynamic feedbacks, to be discussed in the following chapter.

Furthermore, the chapter provided a detailed explanation of how to configure and set up a transient simulation in Serpent. Preliminary test results were presented to evaluate the behavior of Serpent in time-dependent mode. The limitations of this approach, mainly associated with statistical noise and uncertainty, were highlighted. To mitigate these weaknesses, some strategies were adopted: increasing the number of simulated neutrons and enabling population control at each time step. These measures significantly improved the accuracy of the neutron flux evolution by reducing statistical fluctuations. However, this came at the expense of a longer computational time and higher resource demand.

Finally, the improved results obtained with these settings were presented for a test case involving a 0.1 s transient triggered by a step reactivity insertion. The outcomes showed enhanced precision and a better agreement with both the analytical and numerical reference solutions, confirming the potential of Serpent for transient studies when appropriately configured.

5 | Fuel Thermal Feedback

Abstract

To move beyond a simplified transient simulation and to better represent the physical behavior of the system, a more realistic model has been developed by introducing thermal feedback effects. Specifically, this chapter focuses on the implementation of thermal feedback from the fuel only, neglecting any heat exchange with the coolant by assuming an adiabatic model. Consequently, thermal feedback contributions from the coolant are not considered.

5.1. Introduction to Neutronic-Thermal Coupling

The system under analysis is once again the fuel assembly previously studied. However, in this case, the fuel temperature is no longer assumed to remain constant. Instead, it is allowed to evolve dynamically in response to a reactivity insertion that leads to an increase in power. The expected behavior is characterized by an initial rapid rise in power, which is then mitigated by the thermal feedback mechanism. As the fuel temperature increases, the Doppler effect enhances neutron absorption in the resonance energy region, thereby reducing the reactivity and consequently lowering the power output. Over time, this feedback drives the system toward a new equilibrium.

It is important to note that, under adiabatic conditions where heat is not removed by the coolant, the power does not stabilize at a new steady-state level. Instead, the continuous accumulation of thermal energy causes the fuel temperature to keep rising, leading to a further decrease in reactivity. This process ultimately drives the reactor power toward zero.

Coupling Methodology

To address the coupling between neutronics and thermal-hydraulics, a fixed-point iterative scheme is employed, reflecting the inherent nonlinearity of the problem. In fact, the

neutron flux and the effective multiplication factor depend on the cross sections of the nuclides, which in turn are influenced by the temperature and density fields provided by the thermal solution.

The most widely used strategy to handle this coupling is the *Picard iteration*, where the neutronic and thermal-hydraulic solvers are executed sequentially while treating the other field as fixed during each step. Mathematically, this iterative process can be written as:

$$X^{n+1} = H(X^n) \tag{5.1}$$

where X^n represents the solution (both neutronic and thermal quantities) at the n-th iteration, and H is the mapping operator that combines the two physics domains.

This operator-splitting approach allows the use of independent solvers for each physics domain, exchanging the necessary quantities such as power distributions from the neutronic side and temperature fields from the thermal side. The iteration continues until a convergence criterion is met, typically based on the difference between successive iterations.

While simple and widely adopted in both Monte Carlo and deterministic frameworks, the Picard method does not guarantee convergence in all cases. To enhance stability, relaxation techniques are often applied to the exchanged thermal-hydraulic fields.

This iterative scheme forms the basis of the coupling strategy used in this work. A detailed description of how this method is implemented specifically in Serpent, both for steady-state and transient simulations, is provided in the following sections.

5.2. The Serpent Code and the Multiphysics Interface

In the coupled framework developed in this work, the neutronic problem is handled by the Serpent 2 Monte Carlo code, while the thermal behavior of the fuel is solved using a simplified external Python script. Serpent provides pin-wise power distributions for each fuel rod in the assembly, which are then passed to the thermal solver. The thermal module calculates the corresponding fuel temperatures, assuming adiabatic conditions, and returns the updated temperature fields to Serpent. These updated temperatures are used to re-calculate the temperature-dependent cross sections and to perform a new transport calculation. This exchange is repeated iteratively until convergence is achieved.

The thermal model is intentionally kept simple: it assumes no heat removal from the fuel to the coolant, and thus only considers the temperature evolution within each fuel rod due to internal power generation. The spatial resolution of the thermal field can be adapted depending on the level of detail allowed by the multiphysics interface with Serpent.

Internal and External Coupling Options

Serpent supports both internal and external coupling schemes. Internally, it can be coupled to the FINIX fuel behavior module (available under separate licensing), which allows for detailed thermo-mechanical feedback. However, this work utilizes **external coupling**, where Serpent interacts with a custom Python-based thermal solver. This choice offers full control over the thermal model and facilitates transparent testing of the coupling logic. The following figure illustrates the two-level coupling scheme in Serpent 2. Built-in solver modules can be used to obtain fast approximations for fuel and coolant behavior, while high-fidelity solutions from advanced external solvers can be imported through the universal multi-physics interface.

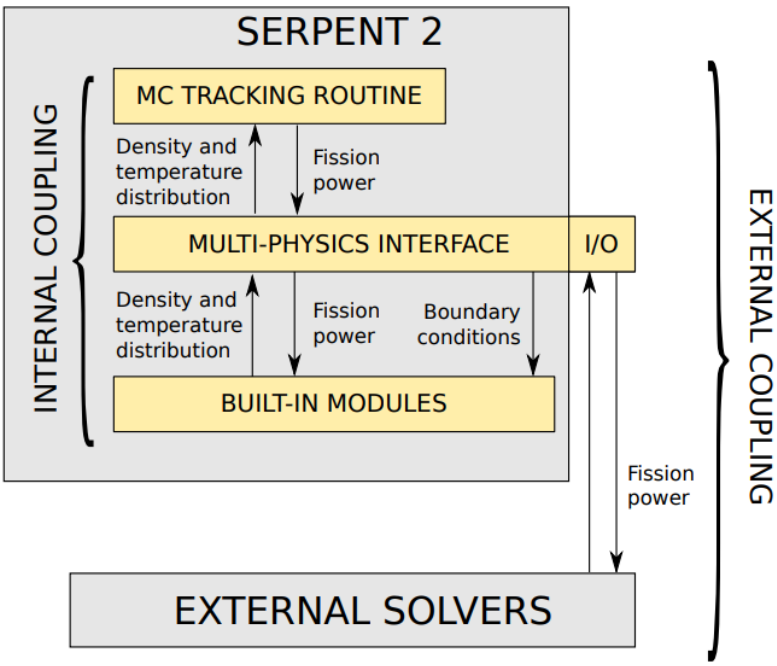


Figure 5.1: The two level coupling scheme in Serpent 2 [29].

The Multiphysics Interface in Serpent 2

Serpent 2 has been designed with multiphysics coupling in mind, enabling the import of externally computed temperature and density fields with arbitrary spatial resolution. These fields can be overlaid directly on top of the base geometry without the need to redefine materials for different conditions. This functionality is achieved through the universal multiphysics interface, which allows Serpent to interact with external solvers such as thermal-hydraulics or CFD codes.

Serpent supports three types of coupling modes:

- **File-based communication** (`set comfile`)
- **POSIX-signal based communication** (`set ppid`)
- **Socket-based communication** (`set pport`)

In this work, the **file-based communication mode** was adopted due to its simplicity and reliability. In this mode, Serpent writes the power distribution to interface files and enters a waiting state. Once the external solver updates the temperature and density fields and writes them to the appropriate files, Serpent reads them and resumes the transport simulation.

The following figure presents the solution flow in externally coupled calculations using the finalized coupled calculation routines implemented in Serpent 2.

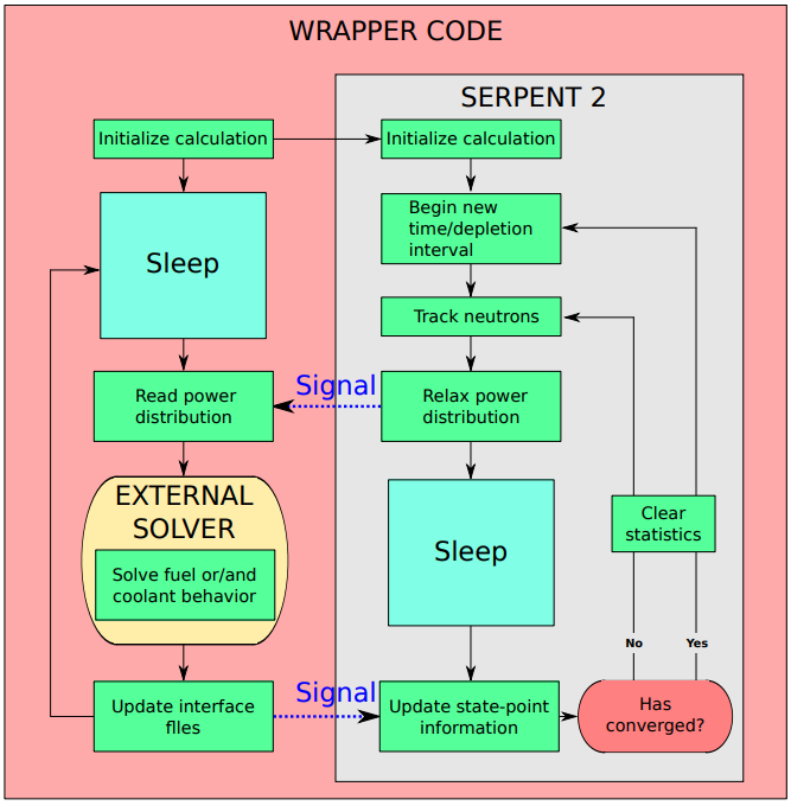


Figure 5.2: Solution flow in externally coupled calculations using the finalized coupled calculation routines implemented in Serpent 2 [14].

Usage of the Multiphysics Interface in Serpent 2

The multiphysics interface in Serpent 2 is activated in the input file using the `ifc` command:

```
ifc FILE
```

where `FILE` refers to the path of the interface definition file. This interface enables the coupling of Serpent with external solvers by allowing the import of temperature and density fields that can be overlaid on top of the base geometry. Multiple interface files can be linked to the same Serpent input to represent different physical fields or materials.

Serpent supports various types of interface formats, each suited to different applications and solver configurations. The available interface types are summarized below:

- **Type 1 – Point-average**: Interpolation of point-wise data.
- **Type 2 – Regular mesh**: For data defined on regular Cartesian or hexagonal grids.

- **Type 3 – Functional dependence**: User-defined functions specified at the source code level.
- **Type 4 – Tetrahedral mesh**: Deprecated, replaced by type 7.
- **Type 5 – Fuel behavior interface (piecewise constant)**: For (z,r) fuel performance models.
- **Type 6 – Fuel behavior interface (radial interpolation)**: Also for (z,r) fuel performance models with interpolated radial resolution.
- **Type 7 – Unstructured mesh**: Based on OpenFOAM file format.
- **Type 8 – Unstructured mesh with many materials**: Also OpenFOAM-based, optimized for multi-material problems.
- **Type 9 – Unstructured mesh with geometry generation**: Imports both the geometry and interface fields from OpenFOAM data.

Each interface type requires a different input format and data structure, tailored to the solver or model it is designed to couple with. However, despite the fact that the input format is structured differently for each interface type, there are common elements. For example, the material to which the temperature and/or density distribution is associated must always be specified. Additionally, the `OUT` flag is used to determine whether the results should be output (1 for yes, 0 for no). The `OUTFILE` parameter specifies the file name to which the output will be printed.

For each interface type, it is also necessary to specify the type of interface, followed by the Cartesian or cylindrical coordinates of the mesh. Afterward, the temperature and/or density distributions should be specified.

The specifications for writing the input for each type of interface are detailed on the VTT website: VTT Multi-physics Interface Specifications. In this work, a **Type 2 (Regular mesh)** interface was used. This choice enabled the spatial discretization of the fuel assembly in both axial and radial directions, allowing for a flexible definition of resolution and level of detail depending on the needs of the analysis.

By using a mesh-based format, the fuel temperature field can be mapped with high fidelity across the entire assembly, facilitating a robust and efficient exchange of data between Serpent and the external thermal solver.

Coupled Calculation Workflow and Signal Exchange

When operating in coupled mode, Serpent follows a dedicated signal-based workflow. After each transport solution, Serpent sends a `SIGUSR1` signal to notify the external solver that the power field is available. It then waits for the external solver to process the thermal problem and return updated temperature and density fields. Once ready, the external solver sends back a `SIGUSR1` to signal that Serpent can proceed with the next iteration at the current time step.

If convergence is reached, the external solver can send a `SIGUSR2` signal to instruct Serpent to advance to the next time step. If Serpent reaches the maximum number of iterations defined by the user (via `set ccmaxiter`), it automatically sends a `SIGUSR2` and waits for updated fields corresponding to the next state point. Upon completion of all time steps and iterations, Serpent sends a `SIGTERM` to indicate the end of the simulation. The algorithm followed by Serpent and the external thermal solver for one depletion step using forward Euler integration and a maximum of two iterations per time step is summarized in the following table.

In transient simulations, the user defines the total number of time steps and the maximum number of iterations allowed per step. The solution proceeds incrementally in time: each time point involves several coupled iterations between Serpent and the thermal solver until convergence is reached. Only then is the simulation advanced to the next time step. This tightly coupled approach ensures accurate feedback modeling at each point in time.

It is important to highlight that detector outputs in transient simulations are handled differently depending on whether the simulation is performed in a standard (non-coupled) transient mode or in a fully coupled mode. In non-coupled transients, Serpent simulates each neutron batch across all time intervals before moving to the next batch, and the resulting detector file aggregates data from all time intervals into a single output. Conversely, in coupled transients, the simulation processes all batches within each time interval sequentially, clearing all tallies between intervals. As a result, one separate detector output file is generated for each time interval (e.g., `[input]_det0.m`, `[input]_det1.m`, etc.).

It should also be noted that, similarly to the standard transient mode, detector outputs in the coupled mode are integrated over time within each time step. This means that quantities such as power are not reported as instantaneous values but rather as time-integrated metrics—for instance, the total energy released during the interval, expressed in joules.

Table 5.1: Coupled calculation flow (Euler method, `ccmaxiter = 2`)

Step	Action	Signal
1.1	Neutron transport solution (STEP 1, ITER 1)	–
1.2	Serpent sends signal to external solver	SIGUSR1 (sss → ext)
1.3	External thermal solution (STEP 1, ITER 1)	–
1.4	External solver sends signal to Serpent	SIGUSR1 (ext → sss)
2.1	Neutron transport solution (STEP 1, ITER 2)	–
2.2	Serpent sends signal to external solver	SIGUSR1 (sss → ext)
2.3	External thermal solution (STEP 1, ITER 2)	–
2.4	External solver sends signal to Serpent	SIGUSR1 (ext → sss)
3.1	Depletion solution (STEP 1 → STEP 2)	SIGUSR2 (sss → ext)
3.2	External solver prepares fields for STEP 2	–
3.3	External solver sends signal to Serpent	SIGUSR1 (ext → sss)
4.1	Neutron transport solution (STEP 2, ITER 1)	–
4.2	Serpent sends signal to external solver	SIGUSR1 (sss → ext)
4.3	External thermal solution (STEP 2, ITER 1)	–
4.4	External solver sends signal to Serpent	SIGUSR1 (ext → sss)
5.1	Neutron transport solution (STEP 2, ITER 2)	–
5.2	Serpent sends signal to external solver	SIGUSR1 (sss → ext)
5.3	External thermal solution (STEP 2, ITER 2)	–
5.4	External solver sends signal to Serpent	SIGUSR1 (ext → sss)
6	End of simulation	SIGTERM (sss → ext)

Legend: `sss` = Serpent; `ext` = External solver; `SIGUSR1` = Iterate current step; `SIGUSR2` = Proceed to next step; `SIGTERM` = End of simulation

Iteration Scheme and Power Relaxation

The coupled calculation is performed using a Picard-style fixed-point iteration scheme. At each time step, Serpent and the external solver iterate sequentially until a converged solution for the coupled fields is obtained.

To improve stability and accelerate convergence, Serpent applies a power distribution relaxation method. The relaxed power distribution at iteration n is computed from the previous iterations using a stochastic averaging approach [8].

The coupled criticality and thermal–hydraulic equations represent a nonlinear problem that must be solved through a numerically stable and iterative method. In the context

of steady-state simulations with Monte Carlo neutronics, Serpent adopts a relaxation strategy inspired by the Robbins–Monro algorithm, which guarantees convergence under general conditions.

Governing Equations. The reactor core state is fully described by two fields: the nuclide field $N(\mathbf{r})$, representing the concentration of isotopes in space, and the temperature field $T(\mathbf{r})$. The neutron flux $\phi(\mathbf{r}, \Omega, E)$ is then computed by solving the criticality (eigenvalue) transport equation:

$$B(N,T)\phi \equiv \left[L(N,T) - \frac{1}{k}F(N,T)\right]\phi = 0,$$

where L and F are operators for neutron losses and production, respectively. The solution $\phi = \phi_B(N,T)$ depends on the material and thermal state of the system.

Stochastic Iteration. Since Serpent is a Monte Carlo code, the flux solution is inherently stochastic and includes a noise component ε, whose magnitude decreases with the number of neutron histories s as $\varepsilon \sim \frac{1}{\sqrt{s}}$. The iteration becomes a stochastic fixed-point problem:

$$\phi = \hat{G}(\phi) = G(\phi) + \varepsilon.$$

To address this, Serpent applies the stochastic approximation scheme proposed by Dufek and Gudowski (2006), where the relaxed power distribution at iteration n is computed as:

$$\phi^{(n)} = (1-\alpha_n)\phi^{(n-1)} + \alpha_n \hat{G}_{s_n}(\phi^{(n-1)}),$$

with $\alpha_n \in (0,1]$ as the step-size and s_n the number of particles in iteration n. A common and robust choice for the step-size is the Robbins–Monro rule:

$$\alpha_n = \frac{1}{n},$$

which leads to the averaged solution:

$$\phi^{(n)} = \frac{1}{n}\sum_{i=1}^{n} \hat{G}_s(\phi^{(i-1)}).$$

This formulation ensures convergence of the power distribution in the presence of statistical noise and is particularly well-suited to iterative coupling in Monte Carlo simulations.

Implementation in Serpent. In Serpent, this relaxation process is applied automatically during coupled calculations. The under-relaxation factor d and the relaxation method are controlled through the input keyword `set relfactor`. This mechanism smooths the stochastic fluctuations of the power field across iterations, accelerating convergence and enhancing the stability of the overall multiphysics simulation.

Monitoring the convergence of the coupled solution

To monitor the convergence of the solution during the simulation, it is not enough to follow the multiplication factor of the system. Instead two methods applied to the thermal-hydraulic fields are used in Serpent. They both require the calculation of the local relative difference between subsequent solution fields. Denoting the number of the iteration by n, and the spatial node index (specifying pin, axial zone and radial node) by k the local relative difference in field Q will be

$$\epsilon_k^n = \frac{Q_k^n - Q_k^{n-1}}{Q_k^n} \tag{5.2}$$

The two convergence criteria calculated by Serpent during coupled calculations are the maximum absolute value of ϵ over the whole geometry

$$\epsilon_{\max}^n = \max_k |\epsilon_k^n| = \max_k \left| \frac{Q_k^n - Q_k^{n-1}}{Q_k^n} \right| \tag{5.3}$$

and the ℓ_n^2-norm of the ϵ field:

$$\ell_n^2 = \sqrt{\sum_k (\epsilon_k^n)^2} = \sqrt{\sum_k \left(\frac{Q_k^n - Q_k^{n-1}}{Q_k^n} \right)^2} \tag{5.4}$$

The first of the two criteria (Eq. 5.3) measures the greatest local relative change in the solution field but does not consider the convergence of the field as a whole. The ℓ_n^2-norm, on the other hand, considers the relative changes in the whole solution field. Both of the convergence criteria are calculated for three fields: The unrelaxed (iteration based) power distribution, the relaxed power distribution and the temperature distribution obtained from the coupled solver. [29]

Temperature-Dependent Cross Sections in Coupled Simulations: Traditional vs. On-the-Fly Methods

In coupled simulations, where neutronics and thermal-hydraulics are interlinked, material temperatures are allowed to change during the simulation. This dynamic temperature variation significantly impacts neutron cross sections, as they are temperature-dependent. For this reason, it is crucial for Serpent to adjust the cross sections in real time to account for these temperature changes as the simulation progresses.

To achieve this, two methods are available in Serpent: a traditional approach, where temperature-dependent cross sections are precomputed and stored, and a more innovative and efficient method, known as the **On-the-Fly Treatment (Target Motion Sampling - TMS)**, which calculates these changes dynamically during the simulation. Below, we explore both methods in detail, with a particular focus on TMS, as it provides greater flexibility and efficiency for dynamic simulations.

Traditional Method: Doppler Broadening Preprocessor

The traditional approach used in Serpent is the Doppler Broadening Preprocessor, which is designed to handle temperature-dependent effects on neutron cross sections. This method accounts for the Doppler broadening of resonances caused by the thermal motion of target nuclei and the changes in reaction probabilities at different temperatures.

In this approach:

- **Precomputation:** Cross sections are precomputed for each material at a discrete set of temperatures before the simulation starts.
- **Storage:** These precomputed cross sections are stored in the system's memory during the simulation.
- **Interpolation:** As the simulation runs and material temperatures change, Serpent interpolates between the precomputed cross sections to determine the correct values for each material at any given moment.

Limitations of this approach include:

- **Memory-Intensive:** Storing cross sections for multiple temperatures requires significant memory, particularly in large systems or burnup calculations where the temperature range can be wide.
- **Static Data:** Precomputed cross sections are fixed for the temperatures that were

prepared in advance. This method cannot handle real-time changes in temperature beyond the predefined range of values.

On-the-Fly Treatment (Target Motion Sampling - TMS)

In contrast, TMS offers a more dynamic and computationally efficient approach, eliminating the need to store precomputed cross sections. Instead, TMS calculates the effects of temperature on the cross sections dynamically during the simulation, in response to real-time temperature changes.

Key Concepts of TMS

Thermal Motion of Nuclei: At non-zero temperatures, the target nuclei in materials are in thermal motion, which affects how neutrons interact with them. This motion causes Doppler broadening of neutron cross sections, influencing reaction probabilities and changing the energy distribution of scattered neutrons.

Rejection Sampling: To account for the thermal motion dynamically, TMS uses a statistical technique called rejection sampling. Instead of precomputing Doppler-broadened cross sections, the method dynamically calculates adjustments to cross sections based on the sampled velocity of target nuclei.

Target-at-Rest Frame: Neutron interactions are simulated as if the target nucleus were at rest. The relative velocity of the neutron with respect to the moving nucleus is calculated, and this is used to modify the interaction probabilities dynamically.

Workflow of TMS

1. Majorant Cross Section: A majorant cross section ($\Sigma_{\text{maj}}(E)$) is defined. This is a conservatively large value that bounds the total cross section for all possible target velocities and interactions. It ensures that the rejection sampling method can sample collision points efficiently without missing valid interactions.

2. Sampling the Collision Point: A collision point is sampled based on the neutron's path length and the majorant cross section:

$$l = -\frac{\ln(\xi)}{\Sigma_{\text{maj}}(E)},$$

where ξ is a random number in the range $[0, 1]$. The sampled collision point determines where the neutron might interact.

3. Sampling the Target Nucleus: Among all possible target nuclides at the collision point, one is selected based on the relative contribution of each nuclide's cross section to the majorant cross section:

$$P_n = \frac{\Sigma_{\text{maj},n}(E)}{\Sigma_{\text{maj}}(E)}.$$

4. Sampling the Target Velocity: The velocity of the target nucleus is sampled from a Maxwell-Boltzmann distribution, which depends on the temperature of the material and the mass of the nucleus:

$$f_{\text{MB}}(v) \propto v^2 e^{-\frac{mv^2}{2kT}},$$

where: - v is the velocity of the nucleus, - m is the mass of the nucleus, - T is the temperature, - k is the Boltzmann constant.

5. Adjusting Cross Sections Dynamically: Using the sampled velocity of the target nucleus, the relative velocity between the neutron and nucleus is calculated:

$$v_{\text{rel}} = |v_{\text{neutron}} - v_{\text{target}}|.$$

The neutron's energy in the target-at-rest frame is determined:

$$E' = \frac{1}{2} m_{\text{neutron}} v_{\text{rel}}^2.$$

Cross sections are evaluated dynamically at E' instead of using precomputed Doppler-broadened data.

6. Rejection Sampling: The sampled interaction is accepted or rejected based on the ratio of the actual cross section to the majorant cross section:

$$\text{Acceptance Criterion: } \xi < \frac{\Sigma_{\text{tot},n}(E', T)}{\Sigma_{\text{maj},n}(E)}.$$

If rejected, the sampling process starts again from step 2 (sampling a new collision point).

7. Final Interaction: If the interaction is accepted, the reaction type (e.g., scattering, absorption) is determined using standard Monte Carlo techniques, and the simulation proceeds.

Key Advantages of TMS

- **Reduced Memory Usage:** There is no need to store cross sections for multiple temperatures, which significantly reduces memory requirements.
- **Real-Time Adjustments:** Cross sections are adjusted dynamically during the simulation, allowing for precise handling of continuous temperature changes in the material.
- **Flexibility:** TMS can manage continuous temperature distributions across materials, making it ideal for multiphysics simulations where thermal feedback dynamically influences the system over time.

TMS is the preferred method for simulations involving dynamic and continuously changing temperatures, as it allows for more accurate and computationally efficient modeling in coupled simulations. [18]

Activating On-the-Fly Treatment in Serpent Input

To activate the **On-the-Fly Treatment (TMS)** in Serpent, the Doppler preprocessor should be disabled for materials that receive their temperature distributions from an external interface. This can be done by removing any temperature definitions from the material cards, such as:

```
tmp <temperature>
```

Serpent will then automatically deduce the minimum and maximum temperatures for the TMS treatment directly from the interface files. If this temperature range needs to be extended, the following definition can be used in the material card:

```
tft <TMIN> <TMAX>
```

Where '<TMIN>' and '<TMAX>' define the desired temperature range for the material, allowing flexibility in the simulation.

5.3. Implementation of Coupled Simulations: Steady-State and Transient Analysis

The coupled simulations were executed using an external coupling approach, where both the Serpent code and the thermodynamic calculations were managed by an external script, written in Python. The Python script was responsible for launching the Serpent simula-

tions, performing the thermodynamic calculations, and handling the exchange of signals between the two codes. This Python script was developed based on the one provided on the VTT website, available at: `https://serpent.vtt.fi/mediawiki/index.php?title=MSCS_for_coupled_transients`.

Coupled calculations were carried out through external coupling, involving the interchange of fields via standard output files (usually in ASCII format). Flow control between Serpent and the external solver was managed using semaphores. Specifically, signals were exchanged through the input and output files, as specified in the Serpent input with the command:

```
# Append signalling mode
set comfile com.in com.out
```

In particular, `com.in` is the file used by Serpent to send input to the external solver, while `com.out` is the file through which the solver communicates back with Serpent.

Details of the coupled calculations using this external-coupling approach can be found in Appendix C.

For the discretization of the fuel assembly, an initial mesh of 10 equally spaced axial nodes was used, each 20 cm long, as shown in the figure below. This discretization allowed for a detailed axial analysis. To refine the model, the assembly was later discretized into 44 axial nodes, particularly for studying various transient scenarios, such as rod extraction and boron dilution.

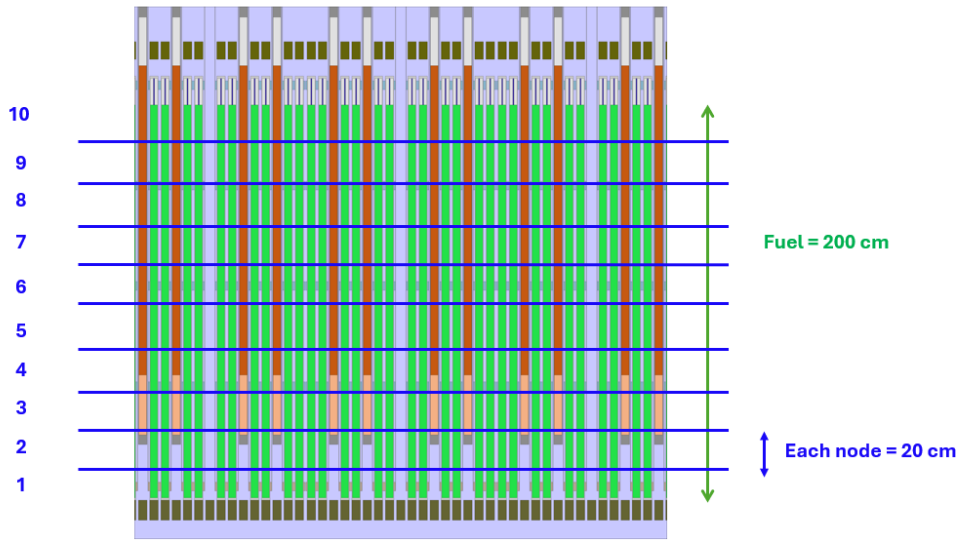


Figure 5.3: Discretization of the fuel assembly within the multiphysic interface.

The two-step process described in Chapter 4.3 remains a mandatory path for transient simulations. However, when dealing with coupled calculations, a slight modification is required. This modification arises because not only the initial sources (for both live neutrons and precursors) must be obtained through a coupled calculation, but also the initial thermodynamic calculation must be started from the converged power distribution.

As a result, the two-step approach is adapted, as depicted in Figure 5.4 for this coupled case.

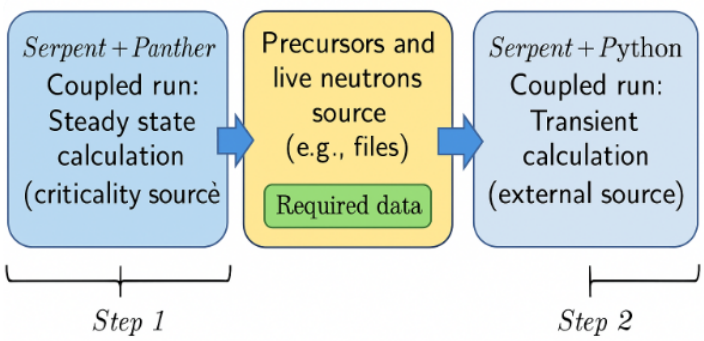


Figure 5.4: Transient two–step approach for coupled calculations for Serpent-Thermodynamic solver.

The following subsections provide a detailed description of the steps involved in setting up and running both steady-state and transient simulations, outlining the necessary assumptions and procedures for each stage.

5.3.1. Steady-State Analysis in Coupled Mode

To obtain the initial distributions for live neutrons and delayed neutron precursors, a steady-state coupled criticality source simulation is first executed. This criticality calculation (the code is found in the appendix) was essential in determining the stable configuration of the system, ensuring criticality through a criticality iteration on boron, which is used to find the critical boron concentration. The calculation was performed based on a given axial temperature distribution of the fuel, which is provided in the table below:

Axial Node Temperature Distribution

Axial Node	Temperature (K)	Axial Node	Temperature (K)
1	587.1	23	782.1
2	603.3	24	764.7
3	632.2	25	756.9
4	657.6	26	746.2
5	681.2	27	731.7
6	703.6	28	715.7
7	724.8	29	698.6
8	744.5	30	680.5
9	762.6	31	651.5
10	779.0	32	626.7
11	791.0	33	613.5
12	797.5	34	605.5
13	795.2	35	599.5
14	809.4	36	595.5
15	814.8	37	593.1
16	816.7	38	591.0
17	816.8	39	589.6
18	815.2	40	588.7
19	812.1	41	588.1
20	807.3	42	587.7
21	801.0	43	587.4
22	793.1	44	587.2

Table 5.2: Temperature distribution at axial nodes in Kelvin

This temperature distribution was obtained from calculations executed with Panther on the same assembly system. It then enabled the calculation of the initial power distribution

using Serpent, which was necessary for the subsequent transient simulations. Moreover, it allowed the generation of the live neutron and precursor sources required to initialize the transient simulations.

The system configuration for the steady-state coupled criticality simulation is shown in the following table:

Steady-State Criticality Simulation Parameters

Parameter	Value
Position of Control Rod	100 step
Simulation Type	Steady-state - coupled - criticality (k=1)
Number of Axial Nodes	44
Critical Boron Concentration	2069 ppm

Table 5.3: System configuration for steady-state criticality simulation

Once a critical configuration is reached, the distributions of live neutrons and precursors are recorded into dedicated files.

The temperature distribution provided by Panther is imposed in the multi-physics fuel interface in Serpent, where the varying temperature profile is reflected in the green shading along the height of the fuel. The different shades represent the temperature gradient along the axial direction, as illustrated in the mesh of the assembly shown below.

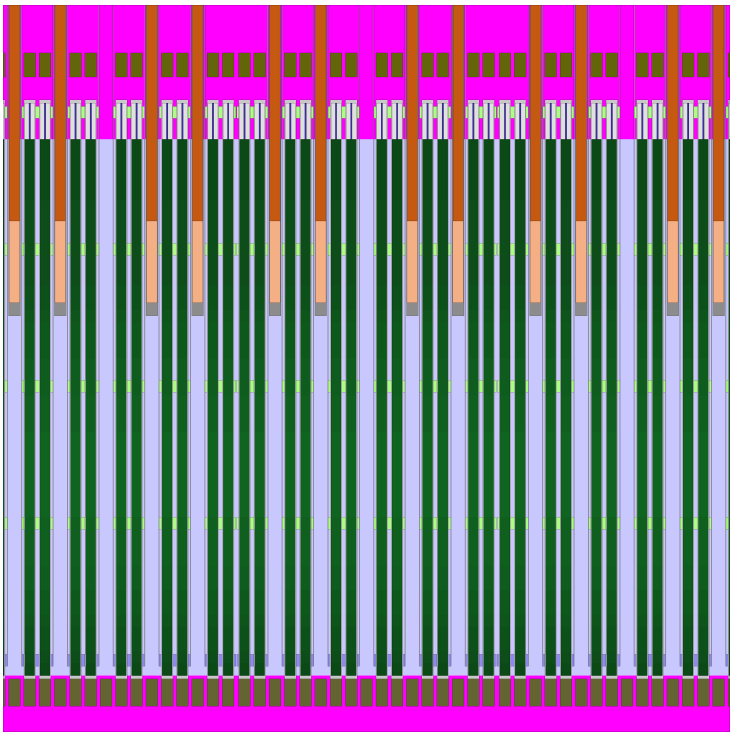


Figure 5.5: Mesh of the assembly showing the fuel temperature distribution.

5.3.2. Coupled Transient Simulation: Methodology and Assumptions

In the second step, which is the transient simulation, the initial live neutron and delayed neutron precursor distributions are set up using the corresponding options. This approach integrates both thermodynamic calculations (with the Python code) and neutronic simulations (with Serpent) for a more realistic simulation of the transient behavior, as schematized in the following image.

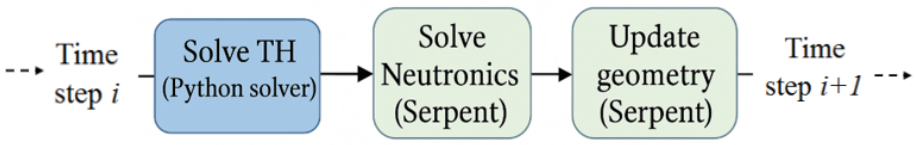


Figure 5.6: Scheme of the code flow for coupled transient calculations.

The following describes the procedure used to update the fuel temperature at each axial node over time, based on the power distribution computed by Serpent.

The equation to describe the energy balance for the fuel is:

$$M_f C_f \frac{dT_f}{dt} = P - K(T_f - T_c) \tag{5.5}$$

where:

- M_f is the mass of the fuel,
- C_f is the specific heat capacity of the fuel,
- T_f is the temperature of the fuel,
- P is the power generated in the system,
- K is the heat transfer coefficient,
- T_c is the temperature of the coolant.

And the equation to describe the energy balance for the coolant is:

$$M_c C_c \frac{dT_c}{dt} = K(T_f - T_c) - GC_c(T_{OUT} - T_{IN}) \tag{5.6}$$

where:

- M_c is the mass of the coolant,
- C_c is the specific heat capacity of the coolant,
- G is the flow rate of the coolant,
- T_{OUT} and T_{IN} are the outlet and inlet temperatures of the coolant.

In the adiabatic model used in this simulation, no heat transfer occurs between the fuel and the coolant, and the temperature of the coolant remains constant. Therefore, the term $K(T_f - T_c)$ in the heat transfer equation is zero, and the second equation, which accounts for the temperature change of the coolant, is not considered. The power $P[i]$, which is calculated by Serpent at each time step for each axial node, is then used to compute the new temperature of the fuel.

The temperature calculation is based on the following energy balance:

$$Q[i] = m[i] \cdot C_f \cdot (T_{\mathrm{EOI}}[i] - T_{\mathrm{BOI}}[i])$$

where:

- $Q[i]$ is the initial heat in each axial node i.
- $m[i]$ is the mass of the fuel in each axial node i.
- T_{EOI} is the final temperature of the fuel at a given time step.
- T_{BOI} is the initial temperature of the fuel (beginning of the iteration).

During each time step, the heat deposited dQ is added to the initial heat, and the new fuel temperature is computed using the following formula:

$$T_{EOI}[i] = \frac{Q[i] + dQ[i]}{m[i] \cdot C_f}$$

where:

- $dQ[i]$ is the heat deposited at axial node i during the current time step, as calculated by Serpent.

This process is repeated for each axial node across all time steps, allowing for an updated power distribution and temperature distribution at each step. The updated temperature influences the cross-sections, which are then used by Serpent to compute the new power distribution for the subsequent time step.

The following Python code snippet illustrates the implementation of this calculation:

Algorithm 5.1 Coupled Transient Simulation Calculation

1: **Start the transient simulation**
2: Define fuel specific heat capacity $C_f = 300\,\mathrm{J/(kg\ K)}$
3: **for** each axial node i from 1 to 44 **do**
4: Calculate mass of this node:

$$m[i] = \left(\frac{\pi \cdot 0.4750^2 \cdot 200.0\,\mathrm{cm}}{44}\right) \cdot 10.5216\,\mathrm{g/cm^3} \cdot 10^{-3}\,\mathrm{kg/g} \cdot 264\,\text{fuel pins}$$

5: Calculate initial heat in this axial node:

$$Q[i] = T_{\mathrm{BOI}}[i] \cdot (C_f \cdot m[i])$$

6: The power calculated by Serpent for this time step is saved in $dQ[i]$ from the multiphysic interface
7: Calculate new temperature based on energy deposition:

$$T_{\mathrm{EOI}}[i] = \frac{Q[i] + dQ[i]}{m[i] \cdot C_f[i]}$$

8: Print new temperature for this node: $T_{\mathrm{EOI}}[i]$
9: **end for**
10: **End of the calculation for this time step**

5.4. Development of a Numerical Code for Comparison

To verify the results of the coupled Serpent and thermodynamic solver, a numerical solution was implemented. This numerical solution is based on the code developed in the previous chapter to study the simple transient calculated by Serpent. However, this time, the solution was expanded to account for the thermodynamic feedback of the fuel.

In this approach, the system is modeled using a lumped parameter model. This simplification assumes that the temperature and heat transfer within the system can be represented by average quantities rather than detailed spatial distributions. This lumped approach is particularly useful for studying the overall behavior of the system without needing to

resolve every detail of the spatial temperature profile.

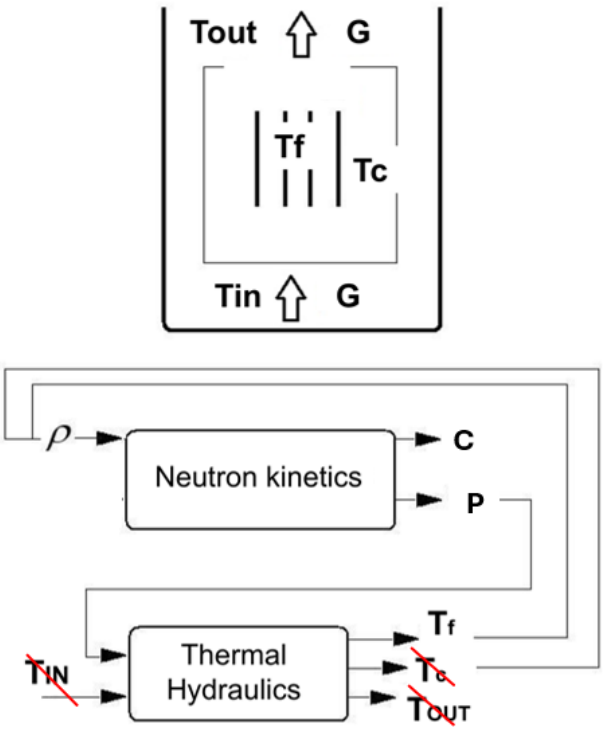


Figure 5.7: Lumped Parameter Model for the Adiabatic Fuel Rod System.

The system consists of a set of coupled equations, as shown in the figure, and includes both the thermodynamic balances for the fuel and coolant, as well as the feedback from the fuel and the point kinetics for neutron transport. The equations governing the system are as follows [2]:

$$\begin{cases} M_f C_f \frac{dT_f}{dt} = P - K(T_f - T_c) \\ M_c C_c \frac{dT_c}{dt} = K(T_f - T_c) - GC_c(T_{OUT} - T_{IN}) \\ \rho = \alpha_f \Delta T_f + \alpha_c \Delta T_c + \rho_0 \\ \frac{dP}{dt} = \frac{\rho - \beta}{\Lambda} P + \sum_i \lambda_i C_i \\ \frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} P - \lambda_i C_i \end{cases}$$

Each equation is explained as follows:

1. **Thermodynamic balances:** - The first equation represents the heat balance for the fuel, where M_f is the mass of the fuel, C_f is its specific heat capacity, T_f is the fuel temperature, P is the power generated, and K is the heat transfer coefficient between the fuel and coolant. - The second equation describes the heat balance for the coolant,

where M_c is the mass of the coolant, C_c is its specific heat capacity, T_c is the coolant temperature, and G is the mass flow rate of the coolant. The term $T_{OUT} - T_{IN}$ represents the temperature difference across the coolant.

2. **Reactivity feedback:** - The third equation models the reactivity feedback from the fuel. α_f and α_c represent the feedback coefficients for the fuel and coolant, respectively, and ΔT_f and ΔT_c are the temperature changes in the fuel and coolant. The term ρ_0 represents the initial reactivity insertion due to factors such as boron dilution or control rod extraction.

3. **Point kinetics:** - The fourth equation represents the neutron kinetics for the system, where n is the neutron population, ρ is the reactivity, β is the delayed neutron fraction, Λ is the neutron generation time, and λ_i and C_i are the decay constants and concentrations of the i-th precursor group. - The fifth equation governs the rate of change of the concentration of each delayed neutron precursor, where β_i represents the fraction of delayed neutrons of the i-th precursor group.

In the case of an adiabatic system assumption, as shown in the previous figure, the heat exchange with the coolant is not considered. Consequently, the system of equations is simplified, and the thermal feedback is only accounted for within the fuel. This leads to the following simplified set of equations:

$$\begin{cases} M_f C_f \frac{dT_f}{dt} = P \\ \rho = \alpha_f \Delta T_f + \rho_0 \\ \frac{dP}{dt} = \frac{\rho - \beta}{\Lambda} P + \sum_i \lambda_i C_i \\ \frac{dC_i}{dt} = \frac{\beta_i}{\Lambda} P - \lambda_i C_i \end{cases}$$

This simplification is valid for an adiabatic system, where the feedback is purely due to the temperature change in the fuel, and no heat transfer occurs to the coolant.

This system consists of a total of 9 equations and 9 unknowns (P, T_f, ρ, C_i, where the C_i represent the concentrations of the 6 delayed neutron precursors), which must be solved numerically. In our case, the Runge-Kutta 45 method is employed to solve the system step by step, updating the temperatures, reactivity, and precursor concentrations at each time step.

Previously, when the feedback was not considered, the system was linear, and the reactivity was constant (a step function). This linearity allowed for a straightforward solution of the point kinetics equations. However, with the inclusion of thermodynamic feedback from the fuel, the system becomes nonlinear. This is because the reactivity now varies

with time due to changes in the fuel temperature, making the system more complex and requiring a more sophisticated numerical approach.

By solving this coupled system between neutronics and thermo-dynamics, we can simulate the behavior of the system with time-dependent feedback, which is essential for accurately modeling the transient response and understanding the effects of fuel temperature on the reactivity.

5.4.1. The Fuel Feedback Coefficient α_f

The fuel feedback coefficient α_f is a critical parameter in modeling the temperature-dependent reactivity of the fuel. It accounts for the changes in reactivity due to temperature variations in the fuel, particularly due to the Doppler effect and the thermal expansion of the fuel material.

Doppler Effect and Its Impact on Fuel Reactivity

The Doppler effect is a physical phenomenon where the resonance absorption cross-section of fuel materials broadens as the temperature increases. As the fuel temperature rises, the atoms in the fuel vibrate more vigorously due to the increased thermal energy. These vibrations cause a shift in the energy levels of the atomic nuclei, leading to the broadening of the resonance peaks in the fuel's absorption cross-section. This broadening means that neutrons across a wider range of energies are absorbed, resulting in an overall increase in neutron absorption by the fuel.

As more neutrons are absorbed by the fuel, the fuel becomes less effective at sustaining the chain reaction, leading to a negative reactivity feedback that helps stabilize the reactor. This feedback mechanism is particularly significant during reactivity-initiated accidents (RIA), as the Doppler effect responds almost instantaneously to changes in temperature, whereas the moderator's temperature effect is slower. Consequently, the Doppler effect is often more influential than the negative temperature coefficient of the moderator, especially during fast transients where rapid changes in reactivity occur.

The physical mechanism at play is that the broadening of resonances leads to an increase in the resonance integral I_{eff}, which contributes to the negative feedback. This effect is represented mathematically in terms of resonance escape probability p, which is the probability that a neutron will be slowed to thermal energy and escape resonance capture. This probability is defined as the ratio of the number of neutrons that reach thermal energies to the number of fast neutrons that slow down.

First, the resonance escape probability for all resonances can be calculated according to the following equation:

$$p = \prod_i p_i \approx \exp\left(-\frac{N_F V_F}{\xi \Sigma_{SM} V_M} \cdot \sum_i \int \frac{\sigma_a(E)}{1 + \frac{\sigma_a(E)}{\sigma_0}} \frac{dE}{E}\right)$$

Where: - p_i is the probability of neutron resonance escape for the i-th resonance, - N_F is the number density of fuel, - V_F is the volume of the fuel, - ξ is the slowing-down coefficient, representing the effectiveness of the moderator in thermalizing neutrons, - Σ_{SM} is the macroscopic cross-section of the moderator, indicating the probability of neutron interactions within the moderator, - V_M is the volume of the moderator, - $\sigma_a(E)$ is the absorption cross-section of the fuel as a function of neutron energy E, - σ_0 is a reference absorption cross-section.

The equation involves an integral over energy E, accounting for all resonances where neutron absorption occurs.

To simplify the expression, the resonance escape probability, I_{eff}, is introduced, which takes into account the broadening of the resonance peaks. The simplified form of the equation becomes:

$$p \approx \exp\left(-\frac{N_F V_F}{\xi \Sigma_{SM} V_M} \cdot I_{\text{eff}}\right)$$

In this expression: - I_{eff} represents the effective resonance integral, which incorporates the Doppler broadening effect, causing the resonances to widen with increasing fuel temperature and thus increasing neutron absorption.

In the literature, empirical relations have been developed for the effective resonance integral I_{eff} of the following form:

$$I_{\text{eff}} = 4.45 + 26.6\sqrt{\frac{4}{\rho D}} \quad \text{for } UO_2 \text{ fuel}$$

Where: - ρ is the material density in g/cm^3, - D is the diameter of the fuel rod in cm.

This equation shows how the resonance escape probability depends on the fuel's temperature: as the temperature increases, the resonance peaks broaden, which leads to an increase in the resonance integral I_{eff}. This, in turn, causes a negative reactivity insertion due to the increased neutron absorption, contributing to the fuel's negative feedback

response. [24]

It is interesting to evaluate the impact of varying fuel density (ρ) on the effective resonance integral I_{eff}. The density of the UO_2 fuel significantly influences the resonance absorption characteristics, with higher fuel densities generally leading to a reduction in the effective resonance integral. The values of I_{eff} were calculated for four different values of fuel density, as shown in the table below.

As observed, the resonance integral I_{eff} decreases as the fuel density increases from 9.5 g/cm^3 to 11 g/cm^3, considering a constant value of 1 cm for the diameter of the fuel rod D. This behavior is consistent with the physical understanding that the absorption characteristics of the fuel are affected by its density, which in turn influences the neutron interaction rates and the broadening of resonance peaks. Additionally, the resonance escape probability p for each value of I_{eff} has been calculated, and the results are shown in the third column of the table.

Fuel Density (ρ) [g/cm^3]	**Effective Resonance Integral I_{eff} [barn]**	**Resonance Escape Probability (p)**
9.5	21.71	0.6129
10.0	21.27	0.6190
10.5	20.87	0.6246
11.0	20.49	0.6301

Table 5.4: Effective Resonance Integral I_{eff} and Resonance Escape Probability p for Varying Fuel Densities

As fuel density decreases, the effective resonance integral (I_{eff}) increases, leading to a decrease in the resonance escape probability (p). This results in more neutron capture, reducing the number of neutrons available for fission. Consequently, this creates a negative feedback mechanism that helps stabilize the reactor by lowering reactivity.

Moreover, the same analysis has been performed for different values of the diameter of the fuel rod (D), considering a constant value of 10 g/cm^3 for the fuel density. The results of the effective resonance integral (I_{eff}) and of the resonance escape probability (p) are shown in the table below.

Fuel Rod Diameter (D) [cm]	Effective Resonance Integral I_{eff} [barn]	Resonance Escape Probability (p)
0.5	28.24	0.5290
1.0	21.27	0.6190
1.5	18.19	0.6636

Table 5.5: Effective Resonance Integral I_{eff} and Resonance Escape Probability p for Varying Fuel Rod Diameters

As expected, increasing the fuel rod diameter leads to both a reduction in the effective resonance integral and an increase in the resonance escape probability, which are indicative of changes in the neutron interaction dynamics within the reactor. These results have important implications for reactor design, particularly in optimizing fuel rod dimensions to balance reactivity and neutron moderation efficiency.

Calculation of the Fuel Feedback Coefficient α_f

In the lumped parameter model, this feedback mechanism is incorporated by modifying the reactivity term in the system of equations. Specifically the reactivity feedback is captured as:

$$\rho = \alpha_f \Delta T_f + \rho_0$$

where: - α_f is the fuel feedback coefficient, - δT_f is the temperature change in the fuel, - ρ_0 is the initial reactivity insertion due to boron dilution or rod extraction.

The coefficient α_f is a function of temperature T and varies with it. However, for practical purposes, an average value of α_f is considered, calculated using the following relationship:

$$\alpha_f = \frac{\Delta \rho}{\Delta T_f}$$

where: - $\Delta \rho$ is the change in reactivity due to a temperature change ΔT_f in the fuel.

In this work, the value of the fuel feedback coefficient α_f was estimated based on detailed simulations of the fuel material's thermal properties and neutron transport. The approach involved performing criticality simulations with Serpent, where the fuel temperature was modified in each case by adjusting the temperature of the fuel material using the "tmp"

card in Serpent. For each simulation, the corresponding k-value was extracted, and the corresponding reactivity ρ was calculated.

This process allowed the creation of a set of ρ values as a function of fuel temperature. These results were then interpolated, as shown in Figure 5.8, to obtain a linear relationship between reactivity and fuel temperature. The slope of this curve, which represents the fuel feedback coefficient α_f, was extracted.

From this interpolation, a value of the slope (coefficient) of -2.609756 pcm/K was obtained, which represents the fuel feedback coefficient α_f.

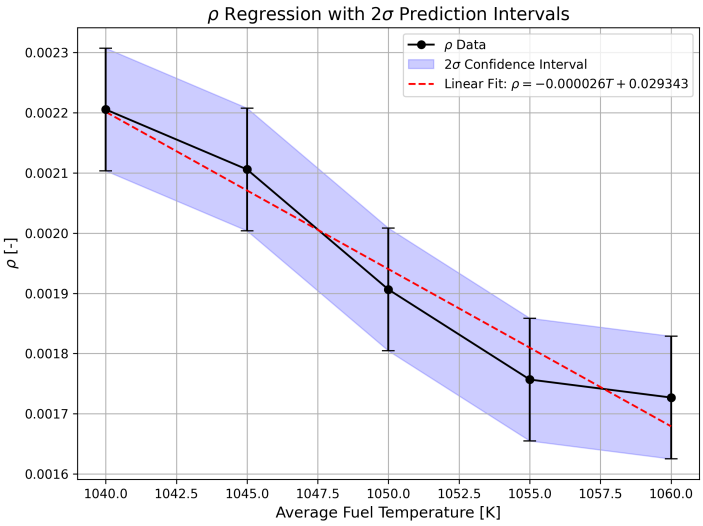


Figure 5.8: Interpolation of reactivity values as a function of fuel temperature to obtain α_f.

The value of α_f was then used as a parameter in the numerical solution code to study the transient and compare the results obtained with the coupled Serpent simulations.

5.5. Results: Boron Diluition Transient

5.5.1. Sensitivity Analysis on Neutron Population

In this section, we present the results of the transient simulation with boron dilution. Two different neutron populations were simulated to analyze the effect on statistical error and computational time. The parameters for both simulations are presented in the table below, showing that the configurations are consistent between the two cases. The only difference between the simulations is the neutron population, which significantly influences

the statistical error and computational time.

Simulation Parameter	**Value**
Neutron Population	20000 / 200000
CPU Time	1d 6h / 12d 12h
Number of Batches per time step	10
Number of Time Steps	100
Fuel Heat Capacity Coefficient	300 J/kg K
Initial Fuel Temperature	900 K (uniform)
Control Rod Position	Raised by 25 steps
Reactivity Insertion	534 pcm (step)
Simulation Type	Coupled (thermal feedback)
Transient Duration	1 s
Boron Concentration (ppm)	468.26 (Critical), 426.02 (Transient)

Table 5.6: Simulation Parameters for Boron Dilution Transients

The results of the two simulations are presented in Figure 5.16 and Figure 5.17. As shown in the figures, the simulation with a smaller neutron population (left) exhibits a much higher statistical error, while the simulation with a larger neutron population (right) produces more precise results. However, the increase in accuracy comes with a significant computational cost, as the simulation with 20,000 neutrons (right) requires considerably more time, as indicated in the table.

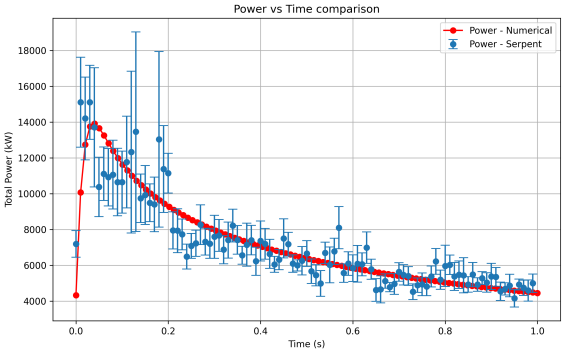


Figure 5.9: Simulation with a smaller neutron population, showing higher statistical error.

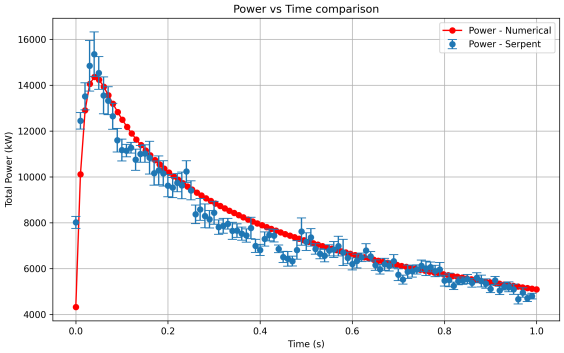


Figure 5.10: Simulation with a larger neutron population, showing more precise results.

The numerical solution shows good agreement with the Serpent results. As seen in both simulations, the fuel temperature feedback mechanism is clearly observed. Following the step reactivity insertion, there is an initial rapid increase in power, after which the system stabilizes. After 1 second of transient, the power returns to approximately its nominal value, demonstrating the stabilizing effect of the fuel feedback.

This confirms that the numerical model effectively captures the system's behavior, particularly the thermal feedback, and provides a reliable solution for simulating transients in the reactor.

Reactivity, Fuel Temperature, and Power Evolution over Time

For the simulation with the larger neutron population, which offers better precision, we can observe the time evolution of reactivity, fuel temperature (parameterized for each axial node), and power (also parameterized for each axial node). As shown in the figures below, the reactivity decreases over time (Figure 5.11) due to the increase in fuel temperature (Figure 5.13), which triggers the thermal feedback mechanism.

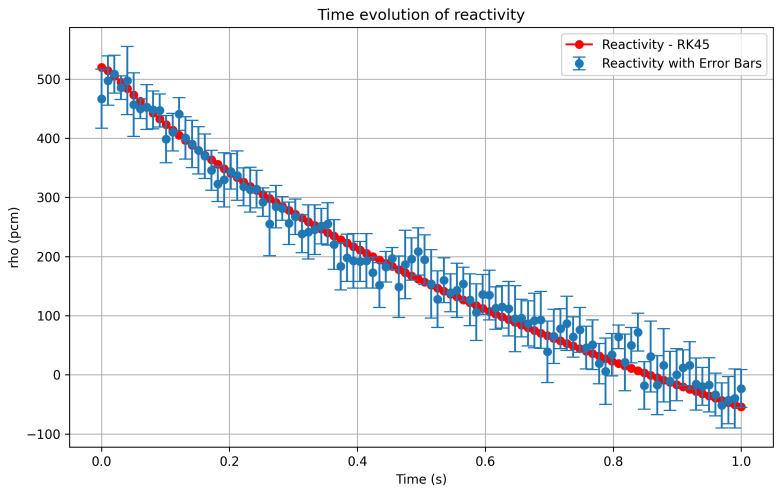


Figure 5.11: Time evolution of reactivity, showing a decrease over time due to the thermal feedback mechanism.

As we can see from the subfigures (Figure 5.12 and 5.13), the power and temperature evolve over time, parameterized for each axial node. The power increases rapidly in the first time steps for the lower axial nodes (node 1 and 2), since the control rods are only raised by 25 steps. This allows only the lower nodes of the assembly to generate power and heat up, which is also reflected in the temperature graphs.

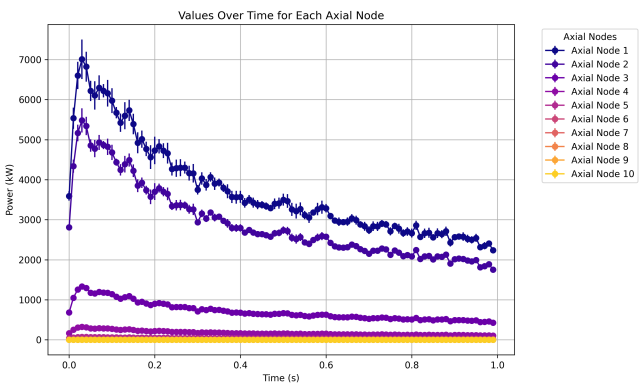


Figure 5.12: Power evolution over time for each axial node.

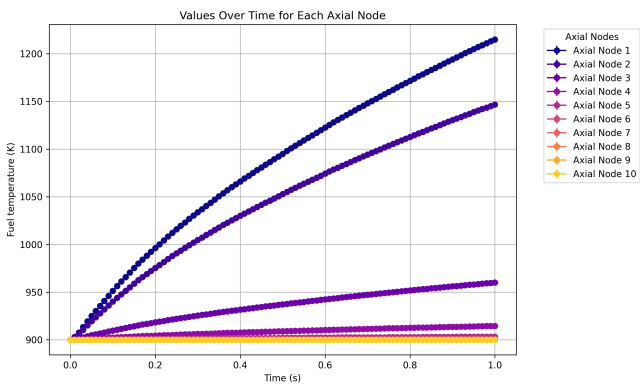


Figure 5.13: Fuel temperature evolution over time for each axial node.

The following graphs Figure 5.14 and 5.15, which are alter-egos of the previous ones, better highlight how only the lower nodes are affected by the transient. The temperature and power profiles evolve over the 100 time steps, with the first and second axial nodes showing higher values due to their location below the control rods. These nodes are able

to generate power because they are not affected by neutron absorption from the control rods, which are raised at 25 steps. In contrast, the nodes near the control rods show zero power, as the fission reactions do not occur in these regions. As time progresses, the power and temperature at the lower nodes stabilize, while the higher nodes (closer to the control rods) remain unaffected.

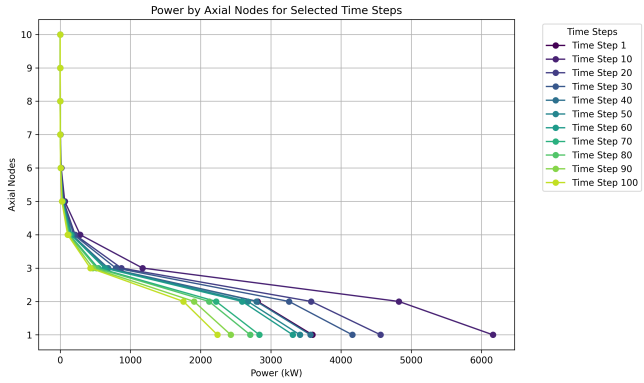


Figure 5.14: Power evolution over time for each axial node.

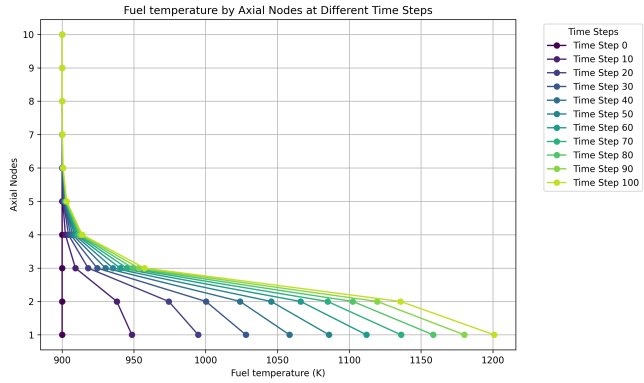


Figure 5.15: Fuel temperature evolution over time for each axial node.

In summary, the time evolution of reactivity, fuel temperature, and power shows the strong influence of the thermal feedback and the impact of the control rod position on the power generation in the upper nodes.

5.5.2. Sensitivity Analysis on Number of Time Steps and Transient Duration

In this section, we present the results of a transient simulation with boron dilution, where the primary focus was on studying the effect of the number of time steps simulated. The simulation parameters are summarized in the table below, where the control rods were raised by 100 steps in both cases. The first simulation was performed over a transient duration of 1 second with 100 time steps, while the second simulation considered a shorter transient of 0.2 seconds with the same number of time steps. The goal of the second simulation was to obtain a finer temporal resolution, as the initial 1-second simulation showed unphysical oscillations. By increasing the number of time steps within a shorter time interval, these oscillations were dampened, as shown in the following figures.

Simulation Parameter	Value
Reactivity Insertion	534 pcm (step)
Control Rod Position	Raised by 100 steps
Initial Fuel Temperature	531.5 K (uniform)
Fuel Heat Capacity Coefficient	300 J/kg K
Transient Time Simulated	1 second / 0.2 second
Neutron Population	100000 / 200000
Number of Batches per Time Step	10
CPU Time	6 days 21h / 12 days 14 h
Simulation Type	Coupled (thermal feedback)
Number of Time Steps	100
Boron Concentration (ppm)	2048.86 (Critical), 1988.23 (Transient)

Table 5.7: Simulation Parameters for Boron Dilution Transients

The results of the two simulations are shown in the following figures (Figure 5.16 and Figure 5.17). As we can observe, the simulation with the shorter transient duration (0.2 seconds) exhibits smoother behavior with reduced oscillations, while the simulation with a longer transient duration (1 second) shows more noticeable oscillations.This difference highlights the importance of temporal resolution for accurately capturing the system's dynamics and minimizing unphysical fluctuations. This is due to the fact that dividing the transient into only a small number of time steps creates oscillations in the curves, as the simulation is unable to accurately track the changing reactivity. Increasing the number of time steps helps to dampen these oscillations, since refining the time allows more frequent population control, which enhances the precision of the results.

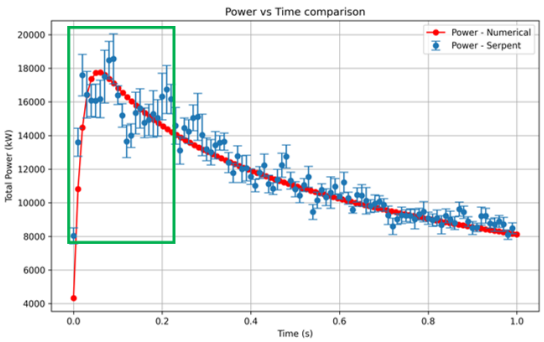


Figure 5.16: Power evolution over time for the first simulation (1 second).

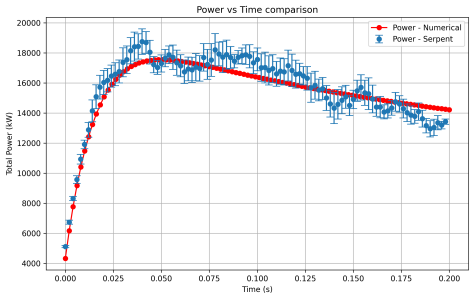


Figure 5.17: Power evolution over time for the second simulation (0.2 seconds).

In conclusion, reducing the transient duration while increasing the number of time steps results in a more detailed and accurate representation of the system's behavior, as evidenced by the smoother evolution of power and the damping of oscillations. This improved temporal resolution, however, comes at the cost of increased computational time, as shown in the simulation parameters table.

Reactivity, Fuel Temperature, and Power Evolution over Time

In this section, the results of the 0.2-second transient simulation are presented, providing a finer temporal resolution compared to the 1-second simulation. This shorter transient time allows for a more detailed examination of the system's behavior.

First, the time evolution of reactivity is illustrated. As shown in Figure 5.18, the reactivity decreases over time due to the thermal feedback from the fuel, reflecting the expected system dynamics.

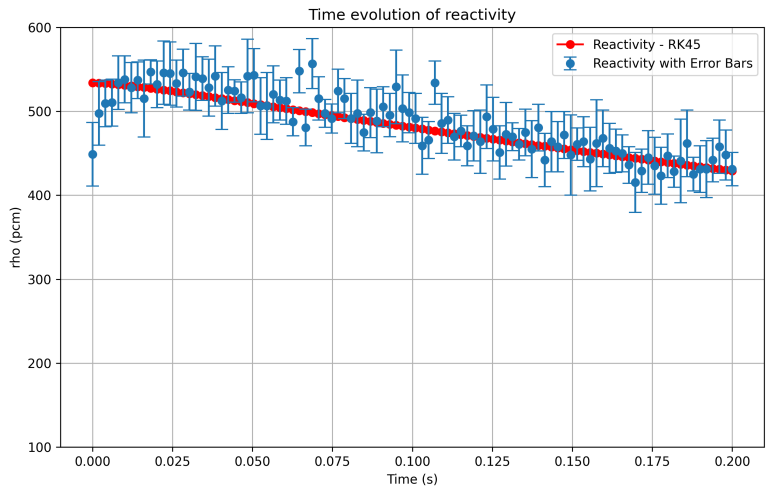


Figure 5.18: Time evolution of reactivity, showing a decrease over time due to fuel thermal feedback.

Next, the temperature evolution is analyzed. The temperature increases over time, consistent with the thermal feedback effect. In Figures 5.19 and 5.20, both the temperature simulated by Serpent and the temperature predicted by the numerical solution, discretized into 10 axial nodes, are presented. The results show good agreement between the two methods.

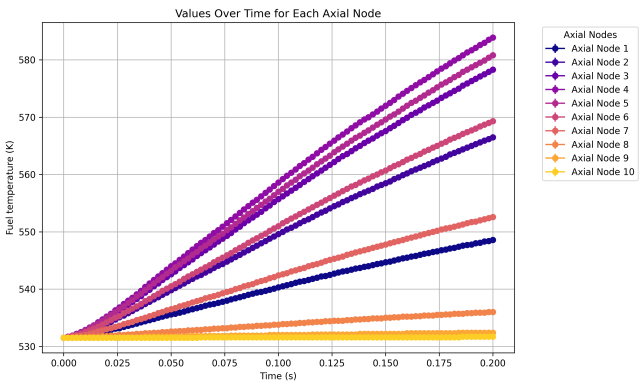


Figure 5.19: Temperature evolution from Serpent simulation.

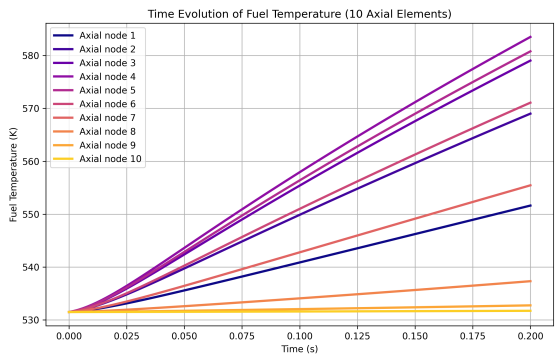


Figure 5.20: Temperature evolution from the numerical solution.

Similarly, the power evolution is shown in Figures 5.21 and 5.22. The power profiles from both Serpent and the numerical solution show excellent agreement, demonstrating that the numerical model accurately captures the power behavior.

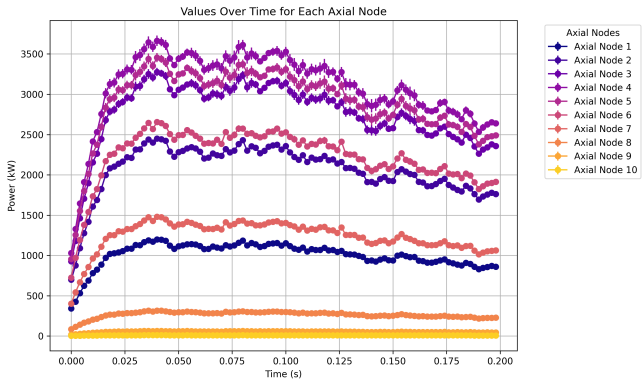


Figure 5.21: Power evolution from Serpent simulation.

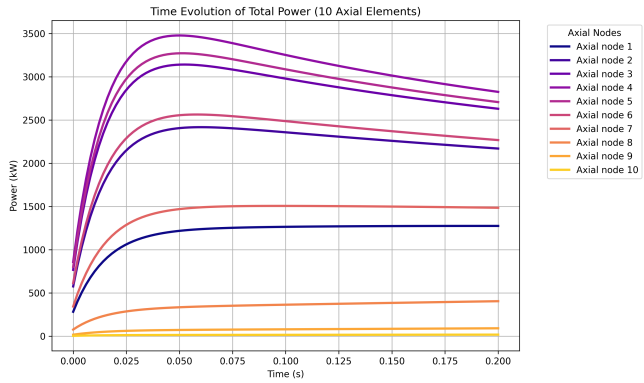


Figure 5.22: Power evolution from the numerical solution.

Lastly, "alter-ego" figures of power and temperature for each axial node, parameterized by the time steps, are presented in Figures 5.23 and 5.24. These figures provide a detailed comparison of the transient across different axial nodes. As can be observed, the higher axial nodes are now also affected by the transient, as the control rods are raised by 100 steps. This allows the upper nodes of the assembly to participate in fission reactions, generating power and increasing in temperature.

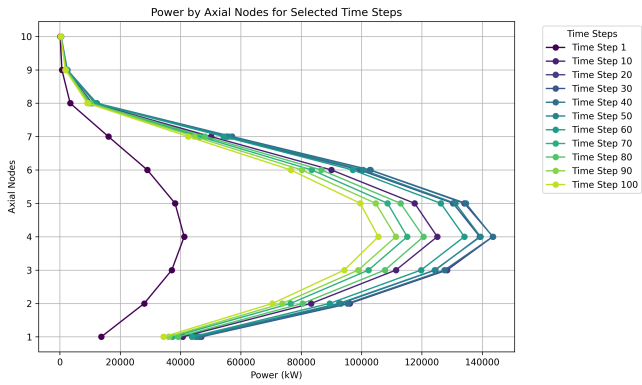


Figure 5.23: Power evolution over time for each axial node.

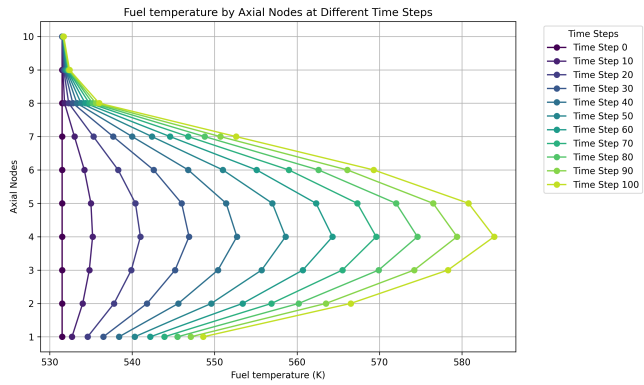


Figure 5.24: Temperature evolution over time for each axial node.

In conclusion, the 0.2-second simulation provides a more detailed view of the transient, reducing unphysical oscillations observed in the 1-second simulation by using a higher number of time steps. The comparison between Serpent results and the numerical solution demonstrates good agreement in terms of power, temperature, and reactivity, confirming the reliability of the numerical model. Furthermore, the results show that, with the control rods raised by 100 steps, even the higher nodes in the assembly are now actively involved

in the transient, participating in fission reactions and exhibiting power and temperature increases.

Convergence Analysis of the Coupled Solution

In this section, the results of the convergence analysis during the simulation are presented. To monitor the convergence of the solution, two methods applied to the thermal-hydraulic fields are used in Serpent. These methods calculate the local relative difference between subsequent solution fields, with the convergence being tracked by the maximum absolute value of ϵ and the ℓ_n^2-norm of the solution fields. Both criteria are evaluated for three fields: the unrelaxed power distribution, the relaxed power distribution, and the temperature distribution.

The graphs shown below illustrate the evolution of ϵ and the ℓ_n^2-norm for the relaxed power distribution and temperature distribution over the 0.2-second transient. These figures provide a visual representation of the convergence behavior.

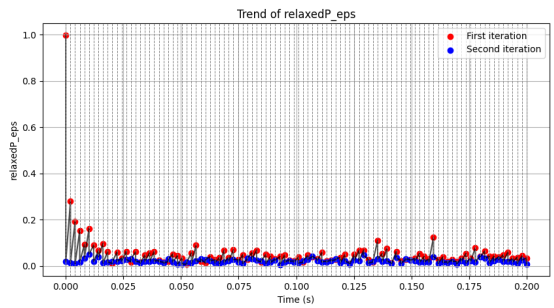


Figure 5.25: Evolution of ϵ for the relaxed power distribution.

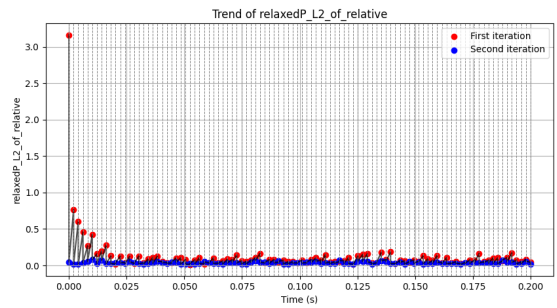


Figure 5.26: Evolution of ℓ_n^2-norm for the relaxed power distribution.

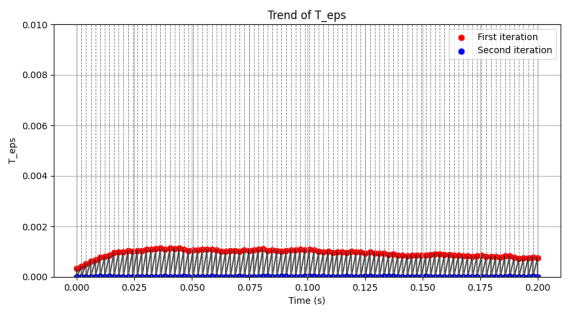


Figure 5.27: Evolution of ϵ for the temperature distribution.

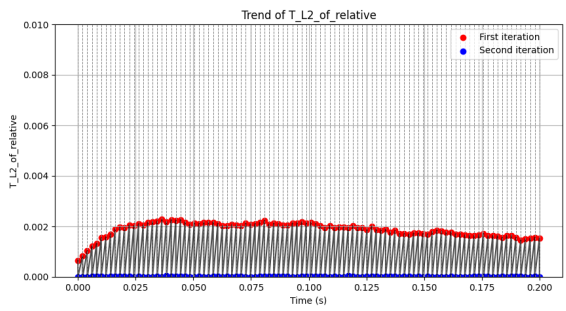


Figure 5.28: Evolution of ℓ_n^2-norm for the temperature distribution.

As can be observed in the figures, for each time step, the first iteration (red dot) shows higher error values compared to the second iteration (blue dot). This discrepancy is due to the convergence process, which is driven by the stochastic approximation implemented in the coupled Serpent-thermodynamic solver. As the iterations progress, the error decreases and the results converge to the true solution.

These results demonstrate the effectiveness of the convergence criteria in monitoring the stability of the solution. The behavior of the relaxed power and temperature distributions aligns well with the expected convergence patterns, as the error reduces over the course of the iterations, ultimately reaching satisfactory accuracy.

5.5.3. Analysis of a 1ms Transient with High Temporal Resolution

In this section, the results of a very brief transient (1 millisecond) are presented, with 100 time steps, resulting in a time grid of the order of 10^{-5} seconds, comparable to the neutron generation time (Λ). This simulation provides a more detailed observation of the behavior of the neutron population at the start of the transient, particularly when the reactivity insertion occurs.

Three different reactivity insertions were simulated: - A low reactivity insertion of 200 pcm - A medium-high reactivity insertion of 450 pcm (approximately 0.7 β) - A super prompt critical case with a reactivity insertion of 1000 pcm (greater than β)

The simulation parameters for each case are summarized in the table below:

Simulation Parameter	Value
Reactivity Insertion	200 pcm / 450 pcm / 1000 pcm
Control Rod Position	Raised by 100 steps
Initial Fuel Temperature	Distribution from Panther
Fuel Heat Capacity Coefficient	300 J/kg K
Transient Time Simulated	1ms
Neutron Population	400000
Number of Batches per Time Step	10
CPU Time	5h 30min (each case)
Simulation Type	Coupled (thermal feedback)
Number of Time Steps	100
Boron Concentration (ppm)	2069 (Critical), 2062.76 / 2027.14 / 1973.53 (Transient)

Table 5.8: Simulation Parameters for 1ms Transients with Different Reactivity Insertions

The analysis allowed for observing the power evolution in the system for all three cases, as shown in Figure 5.29. In this figure, the power for each reactivity insertion is shown together. As observed, in all three cases, the power increases exponentially at the start of the transient. Given the very short transient time, the thermal feedback effect does not have sufficient time to impact the system's behavior during the initial phase.

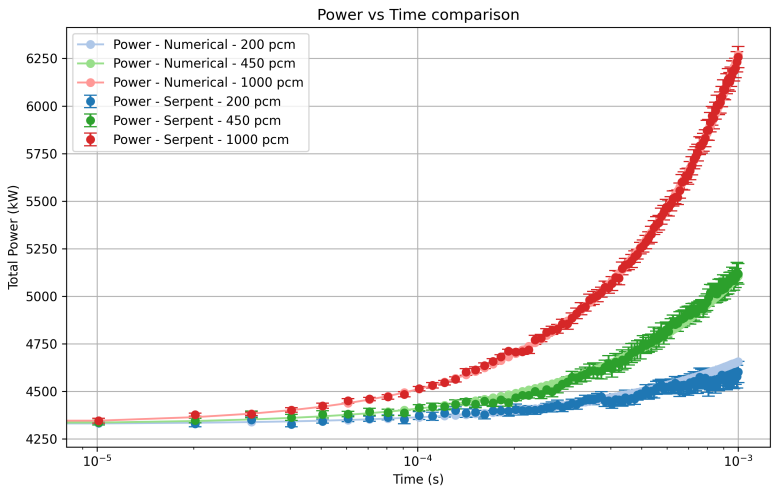


Figure 5.29: Power evolution for all three reactivity insertion cases: 200 pcm, 450 pcm, and 1000 pcm.

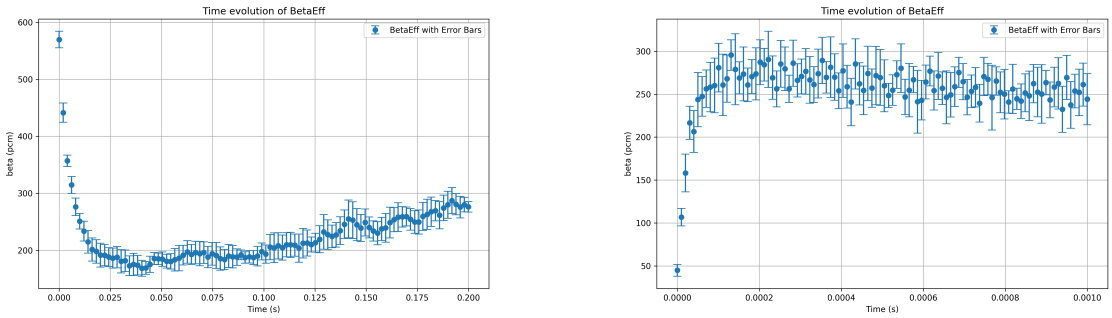
Additionally, the numerical solution for all three cases shows good agreement with the Serpent results, confirming the reliability of the numerical model. This further validates the capability of the numerical solution to predict the system's behavior accurately, even for extremely short transients.

In conclusion, the 1ms transient with 100 time steps provides a high-resolution time grid that allows for a detailed observation of the early-stage dynamics following the reactivity insertion. The results demonstrate how the power increases rapidly at the beginning of the transient, with no significant feedback effect occurring in this very brief time period.

5.5.4. Limitations of Serpent in the Transient Calculation of β_{eff}

The high time resolution achieved in the 1 ms transient simulation allowed for a deeper understanding of an issue observed in Serpent during transient simulations, specifically related to the calculation of kinetic parameters. In particular, the focus was on the parameter β_{eff}, which plays a critical role in understanding the behavior of delayed neutrons.

To investigate this, two transient simulations with different durations and time resolutions were performed. The results are shown in Figure 5.30, where each plot displays the time evolution of the calculated β_{eff} value, including error bars.



(a) 0.2 s transient with 100 time steps

(b) 1 ms transient with 100 time steps

Figure 5.30: Time evolution of β_{eff} during two transient simulations with different time resolutions. The left figure shows an initial drop in β_{eff}, followed by a gradual recovery toward the steady-state value. The right figure shows a low initial value of β_{eff} that progressively increases.

In both cases, the behavior of β_{eff} appears inconsistent with its physical definition, which should remain constant unless the reactor configuration or isotopic composition changes. To understand this behavior, it is important to first recall how β_{eff} is defined and calculated in Serpent.

Definition and Calculation of β_{eff}

The effective delayed neutron fraction is formally defined as the ratio of delayed neutron production to total neutron production. It can be expressed as:

$$\beta_{\text{eff}} = \frac{\sum_{i,j} \left\langle \phi^\dagger \int dE' d\Omega \left(\chi_{d,ij}(E' \to E) \nu_{d,ij} \sigma_{f,i} N_i \phi / 4\pi \right) \right\rangle}{\sum_i \left\langle \phi^\dagger \int dE' d\Omega \left(\chi_{t,i}(E' \to E) \nu_{t,i} \sigma_{f,i} N_i \phi / 4\pi \right) \right\rangle}$$

Where:

- i: fission nuclide index,
- j: precursor group index,
- d: delayed neutrons,
- t: total neutrons (delayed + prompt),
- χ: energy distribution of fission neutrons,
- ν: average number of neutrons per fission,

- σ_f: fission cross-section,
- N_i: number density of nuclide i,
- ϕ: neutron flux,
- $\phi^\dagger$: adjoint flux, representing the importance of a neutron to sustaining the chain reaction.

Adjoint Flux The adjoint flux $\phi^\dagger$ is proportional to the probability that a neutron at a given phase space location contributes to future generations in a fission chain. It weights neutron importance and is crucial in kinetic parameter estimation [10].

The Iterated Fission Probability Method

In Serpent, the calculation of β_{eff} is based on the *iterated fission probability method* [11], which attempts to approximate the adjoint flux through particle tracking. This method is valid under the assumption that the system has reached the fundamental mode of the neutron flux, which is typically satisfied during steady-state (criticality) simulations, but not necessarily in transient conditions.

The method relies on reconstructing the history of neutrons emitted over several generations. Specifically, it defines:

$$\beta_{\text{eff},j}^{(\lambda)} = \frac{\sum_{k\in(\alpha+\lambda)} w_k \cdot \delta_{j,g_k^{(-\lambda)}}}{\sum_{k\in(\alpha+\lambda)} w_k}$$

Where:

- w_k: weight of neutron k,
- $g_k^{(-\lambda)}$: precursor group of the ancestor of neutron k, traced back λ generations,
- $\delta_{j,g}$: Kronecker delta, equal to 1 if $g = j$, 0 otherwise.

Assumptions of the Method

1. The neutron population is traced over multiple generations from an initial ancestor.
2. The flux must have converged to the fundamental eigenmode before meaningful adjoint-weighted results can be obtained.
3. The weighting procedure assumes saturation in neutron population statistics after a fixed number of generations.

Limitations in Transient Simulations The above assumptions do not hold in the early stages of a transient, particularly immediately after a reactivity insertion. During this phase, the neutron flux is rapidly changing and far from the fundamental mode. As a result, the iterated fission probability method provides biased and statistically unstable estimates of β_{eff}, especially in the early time bins.

Numerical Explanation of the Observed Behavior

The non-physical trends observed in Figures 5.30(a) and (b) can be explained through several numerical and statistical factors:

1. Initialization of Delayed Neutron Precursors At the beginning of the transient, Serpent initializes the delayed neutron precursor concentrations based on the steady-state distribution. However, the system requires time to dynamically re-establish a stable balance between precursor production and decay. In very short transients (e.g., 0–1 ms), this equilibration has not yet occurred, resulting in an underestimation of the delayed neutron contribution. As a consequence, the computed value of β_{eff} tends to start at unrealistically low levels and gradually increase as precursor populations accumulate and their influence on the neutron flux becomes statistically visible.

2. Time Binning and Neutron Statistics The duration of each time bin significantly affects the number of neutron histories and reaction events sampled within it. In cases with longer time bins (e.g., 0.2 s total duration with 100 steps, or 2 ms per bin), the statistical sampling is generally sufficient to yield stable estimates of kinetic parameters. Initially, the system still resembles the steady-state configuration, and the computed β_{eff} appears consistent. However, as the transient evolves and the reactor conditions deviate from equilibrium—such as following a reactivity insertion—the balance between prompt and delayed neutrons changes. Serpent's calculation, being sensitive to these evolving conditions, may produce an artificial decrease in β_{eff}, reflecting shifts in fission contributions rather than actual changes in reactor physics.

3. Differences from Steady-State Calculation Methods The calculation of β_{eff} in steady-state simulations relies on well-defined methodologies, such as adjoint-weighted or analog estimators, under the assumption of a stationary neutron flux in the fundamental mode. In transient simulations, this assumption no longer holds. The local flux and neutron importance values vary over time, especially in rapidly changing conditions. As a result, the transiently computed β_{eff} becomes less stable and may lose physical meaning, particularly in cases with high time resolution and short bins, where instantaneous

fluctuations dominate.

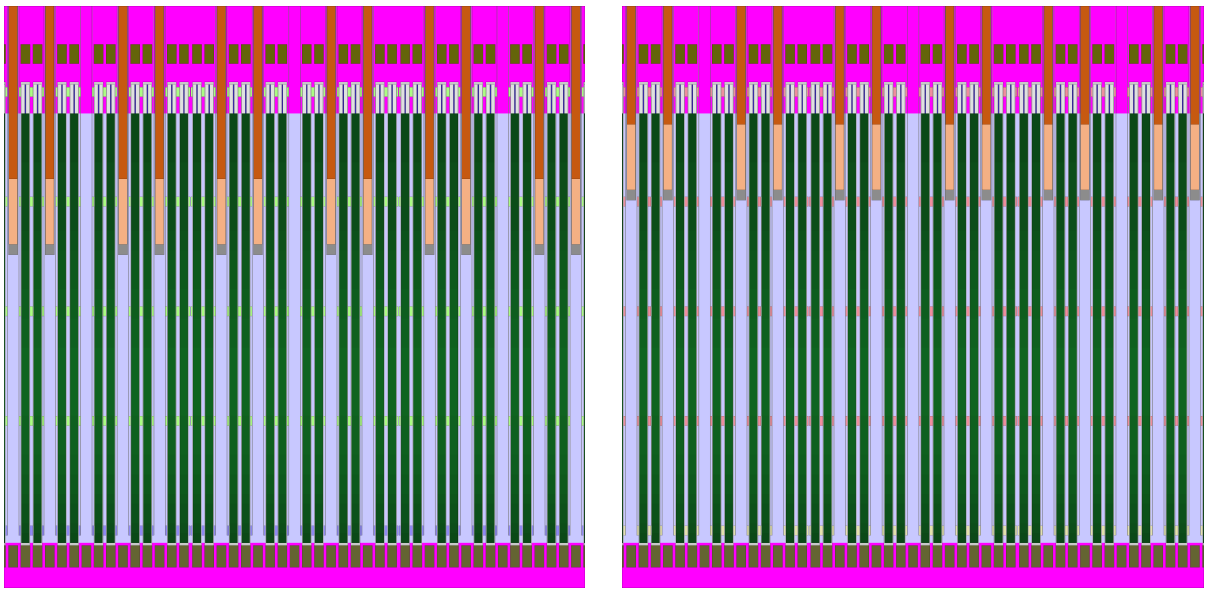
4. Recommendations for Interpretation The values of β_{eff} obtained at each time step during a transient simulation should not be interpreted as physically meaningful variations of the effective delayed neutron fraction. Unless there is a change in geometry, material composition, or core configuration, β_{eff} should remain constant throughout the transient. Instead, it is recommended to use the steady-state value of β_{eff} as a reference and interpret transient outputs with caution, recognizing the numerical and statistical limitations inherent to the Monte Carlo method in dynamic scenarios.

Conclusion

This analysis highlights a known limitation in Serpent's treatment of kinetic parameters during transient simulations. The requirement for flux convergence to the fundamental mode leads to inaccuracies in the calculation of β_{eff} during the early stages of a transient. Such behavior, while not physically meaningful, can be explained by a combination of statistical under-sampling, invalid assumptions regarding flux shape, and methodological limitations of the iterated fission probability approach. For this reason, values of β_{eff} obtained during transients should be interpreted with caution, and steady-state results should be used as the physical reference whenever possible.

5.6. Results: Rod Extraction Transient

This section presents the results of the transient simulations involving a positive reactivity insertion via control rod extraction. The transient is modeled as a step change in control rod position, from 100 steps (critical configuration) to 120 steps (transient configuration), as illustrated in Figure 5.31.



(a) Control rod position: 100 steps (Critical) (b) Control rod position: 120 steps (Transient)

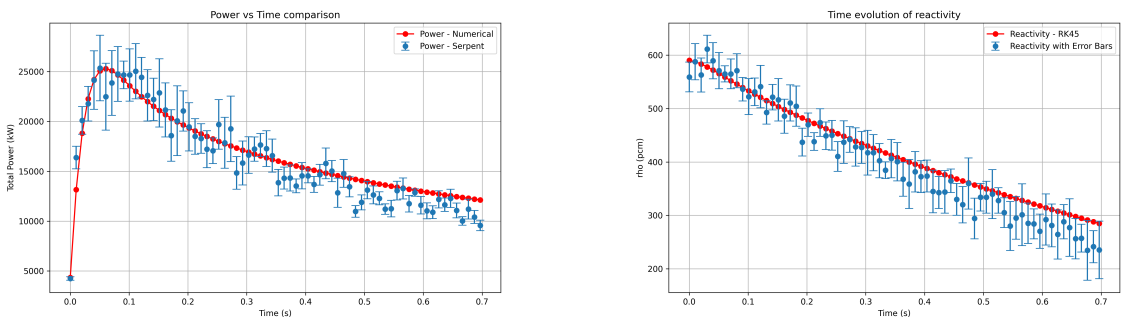
Figure 5.31: Control rod positions used in the transient simulations.

The main simulation parameters are summarized in Table 5.9. The transient is characterized by a 600 pcm reactivity insertion, with thermal feedback enabled and constant boron concentration. The simulations are performed using a coupled neutronics-thermal feedback model, with an improved axial discretization of 44 nodes.

Simulation Parameter	Value
Reactivity Insertion	600 pcm
Boron concentration	2069 ppm (kept constant)
Initial Fuel Temperature	Distribution from Panther
Fuel Heat Capacity Coefficient	300 J/kg K
Transient Time Simulated	0.7 s
Neutron Population	80000
Number of Batches per Time Step	10
CPU Time	9 days 8 h
Simulation Type	Coupled (thermal feedback)
Number of Time Steps	100
Number of Axial Nodes	44
Control rod position	100 steps (Critical), 120 steps (Transient)

Table 5.9: Simulation parameters for rod extraction transient.

Figure 5.32 shows the time evolution of reactor power and reactivity. Serpent results are compared with the numerical solution, and a good agreement is observed between the two, confirming the accuracy of the transient modeling and coupling approach.

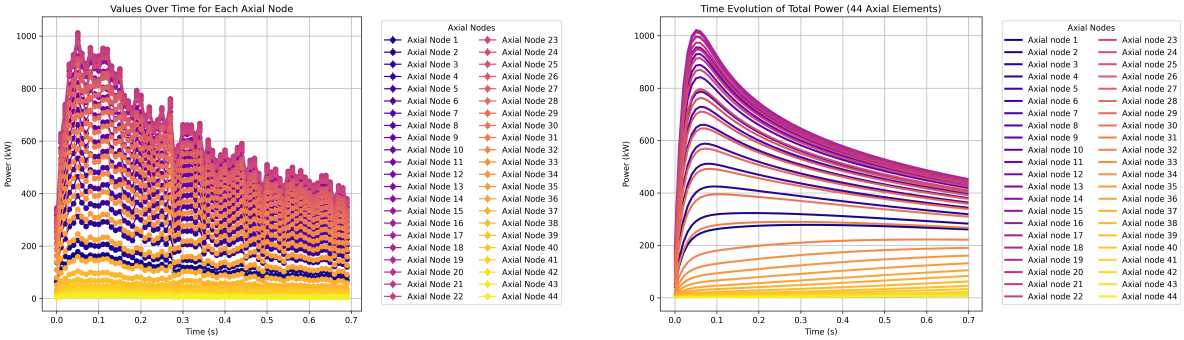


(a) Reactor power evolution (b) Reactivity evolution

Figure 5.32: Time evolution of reactor power and reactivity during the rod extraction transient.

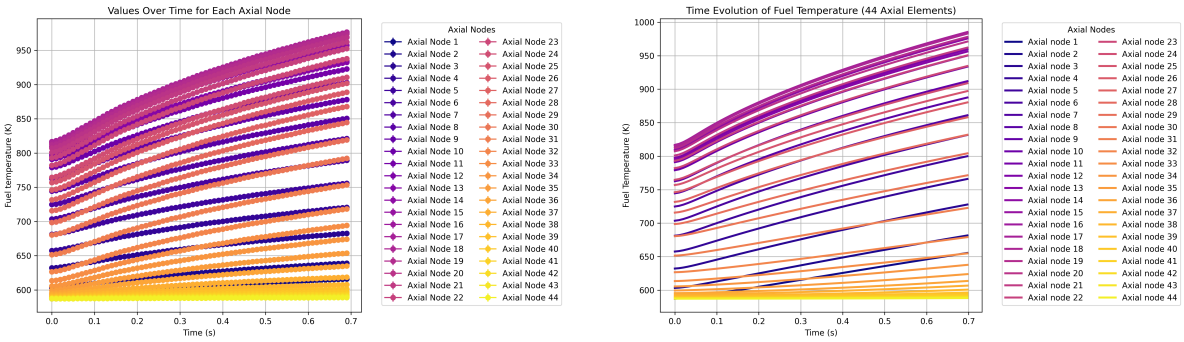
To assess the spatial behavior of the transient, the power and temperature distributions

along the fuel height are plotted for each axial node in both the Serpent and numerical models. These are reported in Figures 5.33 and 5.34.



(a) Serpent results (b) Numerical solution

Figure 5.33: Axial power distribution over time for each axial node (Serpent vs numerical).



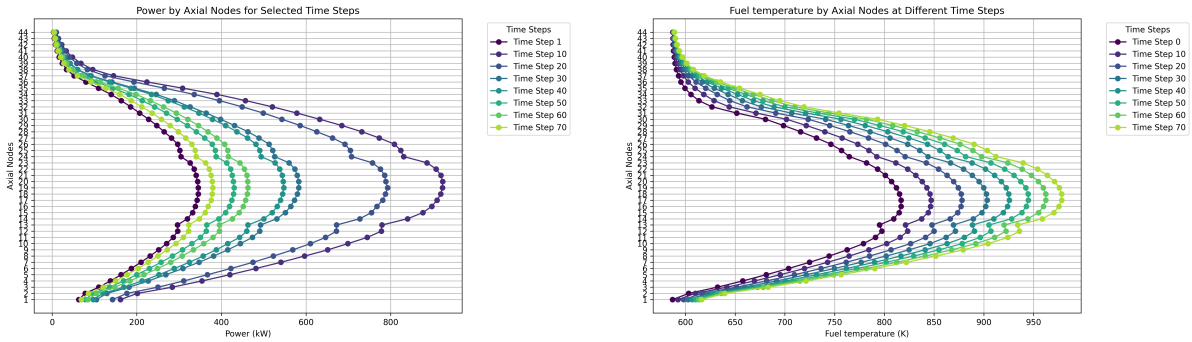
(a) Serpent results (b) Numerical solution

Figure 5.34: Axial fuel temperature distribution over time for each axial node (Serpent vs numerical).

As can be observed from the plots, the upper axial nodes—located near the position of the control rods—show very low or even zero power levels throughout the transient. This behavior is expected, as these regions remain shielded by the inserted control rods and are not significantly affected by the reactivity insertion. In contrast, the lower axial nodes, situated below the control rods, exhibit a noticeable increase in power during the transient. These regions are more active in terms of fission reactions, as they are outside the influence of the absorber material and thus respond more strongly to the positive reactivity insertion caused by the rod withdrawal.

Figure 5.35 presents an alternative perspective by showing the time evolution of power

and temperature for each axial node individually. These plots reveal localized drops in power and temperature along the fuel column.



(a) Power vs time (all nodes) (b) Zoomed view of central nodes

Figure 5.35: Time evolution of reactor power for each axial node.

Compared to the previous boron dilution simulations, which employed only 10 axial nodes, the refined spatial discretization adopted in this case (44 axial nodes) provides a much more detailed view of the axial behavior during the transient. Thanks to this increased resolution, localized reductions in power and temperature can be observed along the fuel column. These drops are associated with the presence of spacer grids—structural elements placed between fuel segments that do not generate power and are typically cooler than the surrounding fuel. As such, their presence leads to identifiable depressions in the spatial profiles of both power and temperature, enhancing the physical interpretability of the model results.

5.6.1. Transient Comparison for Two Different Control Rod Configurations

In this final analysis, a comparison is presented between the previously discussed rod extraction transient and a second simulation, where the control rods are initially positioned much deeper within the core. In this new configuration, the control rods start at 25 steps and are extracted to induce a reactivity insertion of approximately 700 pcm.

Unlike the previous case—where the control rods were initially positioned in the upper-middle region of the fuel assembly—this configuration affects mainly the lower axial nodes. Since the absorber material is located until the bottom of the assembly, only the lower part of the core is exposed to the reactivity change, leading to localized transient effects.

Table 5.10 summarizes the key parameters for both simulations.

Simulation Parameter	Upper-Mid CR Position	Lower CR Position
Initial CR Position	100 steps	25 steps
Final CR Position	120 steps	26 steps
Reactivity Insertion	600 pcm	700 pcm
CPU Time	7d 12h	8d 21h
Transient Time	0.2 s	
Initial Fuel Temperature	Distribution from Panther	
Fuel Heat Capacity Coefficient	300 J/kg K	
Neutron Population	200000	
Number of Batches per Time Step	10	
Number of Time Steps	20	
Axial Nodes	44	
Simulation Type	Coupled (thermal feedback)	
Boron Concentration	2069 ppm	522.5 ppm

Table 5.10: Comparison of simulation parameters for two rod extraction transients.

Figure 5.36 shows the time evolution of reactor power for both transients, with 20 time steps.

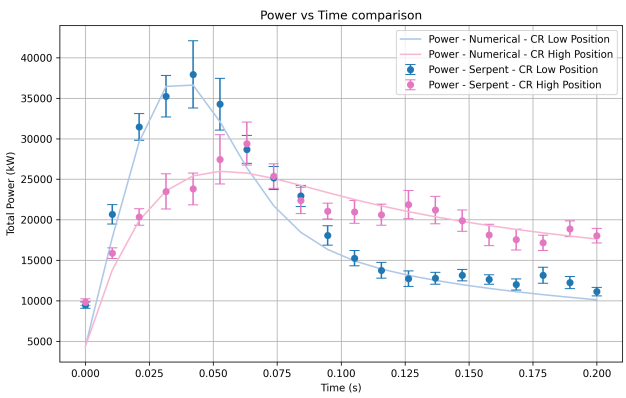


Figure 5.36: Comparison of power evolution in the upper-mid and lower CR configuration.

The unexpected results from the two transient simulations can be explained by the differ-

ent boron concentrations in the two scenarios. Both cases started from a critical reactor condition, but the boron concentration was significantly different: 2069 ppm in the first case and 522.5 ppm in the second case. In the first case, where the control rods were raised by 20 steps (adding 600 pcm of reactivity), the higher boron concentration helped counterbalance the reactivity insertion, leading to a more controlled transient. In contrast, in the second case, where the control rods were raised by only 1 step (adding 700 pcm of reactivity), the much lower boron concentration failed to adequately counteract the reactivity increase. This resulted in a disproportionately large reactivity insertion in the second case, as the lower boron concentration was unable to mitigate the reactivity contribution from the control rods as effectively as in the first case.

Conclusion

In this chapter, we have discussed the final step of my thesis, which involves the coupling between neutronics, performed by Serpent, and a thermodynamic code to integrate the thermal feedback from the fuel due to the Doppler effect, in accidental transient scenarios such as boron dilution and rod ejection. Initially, Serpent's method for implementing coupling with external codes through a multi-physics interface was presented. This interface allows the user to specify temperature and density distributions along the system, using various mesh types, and keeps track of temperature variations during the transient due to changes in geometries or system concentrations.

For the purposes of this work, very short transient scenarios were considered due to the high computational cost of the Monte Carlo code. As a result, only the thermal feedback from the fuel, primarily due to the Doppler effect, was taken into account, modeling an adiabatic system that does not consider temperature variations in the coolant and its associated thermal feedback, which becomes significant only after several seconds. In contrast, the thermal feedback from the fuel due to the Doppler effect manifests almost instantaneously.

The results obtained showed that the use of Serpent's multi-physics coupling method is effective in studying the behavior of transients and the feedback that reduces the power after the peak caused by the step insertion of reactivity, bringing it back towards the steady-state value. It was observed that increasing the simulated neutron population improves result accuracy, though at the cost of higher computational time and expense. Furthermore, it was demonstrated that increasing the number of time steps used to divide the transient time allows for a finer temporal grid, damping non-physical oscillations and providing a better description of the transient behavior.

Additionally, improving the discretization of the multi-physics interface mesh allowed for better spatial detail, revealing localized reductions in temperature and power along the assembly, which were lost with a coarser grid. However, some limitations of Serpent in transient simulations were also observed. Specifically, Serpent does not calculate kinetic parameters, particularly the effective beta (β_{eff}), accurately in transient scenarios. This leads to non-physical results, likely due to Serpent's routines being optimized for criticality mode rather than transient analysis. Despite these limitations, the Monte Carlo method has demonstrated good performance, and when compared with the numerical solution of point kinetics, it showed good agreement. Despite the still long simulation times required, the high accuracy of the results thanks to the Monte Carlo method is a valuable trade-off.

6 | Conclusions and Future Developments

The main scientific question stated in this dissertation is the investigation of advanced methodologies to develop the coupled neutronic-thermal-hydraulic calculations within nuclear systems, specifically focusing on the study of Monte Carlo-based neutronics coupled with thermodynamics in transient scenarios for SMR. A series of objectives were accomplished to provide a concise answer to this question, where the main findings and conclusions are summarized here.

6.1. Main Conclusions

The scientific question addressed in this thesis led to the selection and verification of suitable calculation codes, followed by the implementation of a versatile coupling between them. To overcome the major physical approximations inherent in the industry-standard approach, the cell-core scheme was replaced by a direct method. The system under investigation consists of an assembly in an infinite core, employing pin-wise neutronic and thermodynamic representations, which provide a more accurate depiction of the spatial dependence of the involved physical phenomena. Specifically, the method incorporates MC-based neutronics, coupled with a simplified adiabatic thermodynamic model, implemented in Python, which eliminates common neutronic approximations and enables direct pin-by-pin feedback on temperatures. This allows for a more precise representation of thermal feedback effects, particularly those from the fuel due to the Doppler effect, which is crucial in transient scenarios.

The combination of Serpent for neutronic calculations and the adiabatic thermodynamic model for feedback proved to be a suitable approach. Both transient and steady-state scenarios were explored, with the steady-state analysis providing key kinetic parameters that were used to feed the point kinetics model, essential for comparing the transient simulations. Additionally, these steady-state results were crucial for initializing the transient simulations.

Regarding the coupled transient simulations, good agreement was found with the numerical solution from point kinetics. However, despite the positive results, a major drawback of the Monte Carlo method remains the high computational resources required for these simulations. This trade-off between accuracy and computational cost is a significant challenge, but the consistency of the results obtained encourages further work along these lines, paving the way for industry-like applications.

Aspect	**Monte Carlo (MC)**	**Point Kinetics (PK)**
Computational Cost	High (especially for transient simulations)	Low (faster to compute)
Spatial Dependence	3D spatial analysis, captures spatial effects.	0D model, does not capture spatial dependence.
Accuracy	High accuracy, provides confidence intervals for uncertainty.	Sufficient accuracy for short transients but lacks spatial detail.
Kinetic Parameters	Essential for accurate kinetic parameter estimation.	PK models need kinetic parameters calculated by MC, as they cannot provide these parameters themselves.
Feedback Consideration	Due to the high computational cost, Serpent is impractical for simulating long transients to include coolant feedback.	Can model more detailed feedback (e.g., coolant). PK models can include fuel feedback, but need the α_f (fuel feedback coefficient) from Serpent.

Table 6.1: Comparison between Monte Carlo and Point Kinetics for studying nuclear system transients.

6.2. Future Developments

In this section, potential future developments to improve the efficiency of the Monte Carlo-based simulations will be discussed.

One important direction for improvement is the implementation of socket-based communication instead of the current file-based mode. This change would significantly enhance the

efficiency of data exchange between the neutronics and thermodynamic models, reducing memory usage and speeding up the simulation process.

Another key development is expanding the model to include coolant feedback, which would provide a more realistic representation of reactor behavior. However, this would require simulations with longer time frames, thus demanding greater computational resources. The inclusion of coolant feedback would improve the accuracy of the thermal-hydraulic response, particularly in long-term transient scenarios.

To optimize computational efficiency, strategies for balancing accuracy and computational cost must be explored. Approaches such as parallelization, where simulations are divided into smaller tasks and run concurrently, and adaptive time-stepping, which adjusts the time step based on the transient's characteristics, could be used to optimize the trade-off between performance and precision.

Additionally, the use of multi-physics interfaces with customizable, user-defined meshes could allow for greater detail and accuracy in simulations. By providing finer control over mesh resolution, these interfaces would be particularly beneficial in regions of the reactor where small-scale effects significantly impact the overall behavior.

Further investigation into Serpent's ability to simulate rod extraction with ramp reactivity insertion is also an important area for future work. This feature should be explored in more detail, especially with the use of the dedicated transient mode card, which allows for specifying rod withdrawal speeds. Accurate modeling of rod extraction is essential for understanding transient behavior during operational changes such as shutdowns or accidents.

While the current model simplifies the fuel assembly design, future studies may incorporate additional effects such as burnup, xenon transients, and the presence of burnable absorbers. These factors would further refine the model and provide a more complete and accurate simulation of the transient response.

Although the Monte Carlo method provides a high level of accuracy for transient analysis, its computational cost remains a significant limitation. The simulations conducted in this work demonstrated the need for substantial computational resources, especially when dealing with complex reactor models and transient scenarios. To improve the feasibility and efficiency of such analyses, future developments should focus on integrating machine learning (ML) algorithms and reduced order methods (ROM) techniques into Monte Carlo simulations.

Machine learning has shown great potential in enhancing computational efficiency by au-

tomating the discovery of patterns and relationships within large datasets. In the context of Monte Carlo simulations, ML can be applied to surrogate modeling where machine learning models approximate the results of computationally expensive simulations, thus significantly reducing the time needed for analysis. Algorithms such as neural networks, decision trees, and support vector machines can be trained on existing simulation data to predict outcomes for new cases with a reduced computational effort.

In addition, reduced order methods ([22], [21], [20]), which aim to reduce the complexity of the physical models involved, can help decrease the number of degrees of freedom in simulations without compromising accuracy. Techniques such as proper orthogonal decomposition (POD) or balanced truncation can be used to approximate the full-scale system with lower-dimensional models. These reduced models retain essential dynamic behaviors, enabling faster simulations while maintaining accuracy in predicting transient events. The combination of machine learning and model order reduction has the potential to revolutionize the application of Monte Carlo methods for transient analysis, allowing for real-time or high-frequency simulations in complex nuclear safety studies.

Finally, an important application of the developed methodology was the code-to-code comparison between Serpent and WIMS, initiated in collaboration with Luc Kooijman, at Tractebel. This analysis focused on the behavior of a fuel rod under both steady-state and transient conditions. The objective of this study was to compare the results obtained from the two codes, assessing their accuracy and performance in modeling the fuel rod behavior under different operating conditions. These simulations provided valuable insights into the strengths and limitations of each code, which can guide future improvements in reactor modeling.

Bibliography

[1] L. Agostin. Monte-carlo modelling of the nuscale small modular reactor: A comparative approach. Technical report, IMT Atlantique - Tractebel, 2024.

[2] A. B. d. T. G. M. M. P. D. C. E. P. Antonio Cammi, Roberto Ponciroli. A zero dimensional model for simulation of triga mark ii dynamic response. *Progress in Nuclear Energy*, 2013.

[3] F. B. Brown. Physor-2006, ans topical meeting on reactor physics. In *On the Use of Shannon Entropy of the Fission Distribution for Assessing Convergence of Monte Carlo Criticality Calculations*, Vancouver, BC, Canada., 2006. Canadian Nuclear Society.

[4] D. G. Cacucci. *Handbook of Nuclear Engineering.* Springer, Boston, MA, USA., 2010. ISBN 978-0-387-98149-9.

[5] R. T. C. A. C. F. de Santo. Review of kaganove's solution for the reactor point kinetics equations. *Instituto de Engenharia Nuclear*, 1993.

[6] J. J. Duderstadt and L. J. Hamilton. *Nuclear Reactor Analysis.* John Willey sons Inc., New York, USA, 1976. ISBN 0-471-22363-8.

[7] D. E. Ferraro. *Monte Carlo-based multi-physics analysis for transients in Light Water Reactors.* PhD thesis, Karlsruhe Institute for Technology (KIT), 2021.

[8] J. D. . W. Gudowski. Stochastic approximation for monte carlo calculation of steady-state conditions in thermal reactors. *Nuclear Science and Engineering*, 2017.

[9] A. Haghighat. *Monte Carlo Methods for Particle Transport.* Taylor Francis Group, LLC, 2021.

[10] M. Hayashi. Analytical solution of diffusion adjoint equation for homogeneous bare critical reactor. *Annals of Nuclear Energy*, 2018.

[11] E. F. R. R. S. v. d. M. Jaakko Leppänen, Manuele Aufiero. Calculation of effective

point kinetics parameters in the serpent 2 monte carlo code. *Annals of Nuclear Energy*, 2013.

[12] Y. N. . T. KAMEYAMA. Development of calculation technique for iterated fission probability and reactor kinetic parameters using continuous-energy monte carlo method. *Journal of Nuclear Science and Technology*, 2012.

[13] J. Leppänen. Input syntax manual - serpent wiki, 2004. URL `https://serpent.vtt.fi/mediawiki/index.php?title=Input_syntax_manual`.

[14] J. Leppänen. VTT website, 2004. URL `https://serpent.vtt.fi/serpent/`.

[15] J. Leppänen. *Development of a New Monte Carlo Reactor Physics Code*. PhD thesis, Helsinki University of Technology, Espoo, Finland., 2007.

[16] J. Leppänen. Development of a dynamic simulation mode in serpent 2 monte carlo code. Technical report, VTT Technical Research Centre of Finland, 2013.

[17] J. Leppänen. *Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code*. VTT Technical Research Centre, Finland, 2015.

[18] T. V. . J. Leppänen. Tms on-the-fly temperature treatment in serpent. Technical report, VTT Technical Research Center of Finland, 9 2014.

[19] E. Lewis and W. Miller. *Computational methods of neutron transport*. John Willey sons Inc., New York, USA, 1984. ISBN 0-471-09245-2.

[20] S. Lorenzi. An adjoint proper orthogonal decomposition method for a neutronics reduced order model. *ScienceDirect*, 2018.

[21] M. F. Manuele Aufiero. Stabilization and convergence acceleration in coupled monte carlo–cfd calculations: the newton method via monte carlo perturbation theory. *ResearchGate*, 2017.

[22] M. F. Manuele Aufiero, Michael Martin. Xgpt: Extending monte carlo generalized perturbation theory capabilities to continuous-energy sensitivity functions. *ScienceDirect*, 2016.

[23] D. V. Nance. Notes on an elementary point kinetics problem for a nuclear reactor. Technical report, Independent Research Scientist, 9 2022.

[24] N. Power. Resonance escape probability, 2025. URL `https://www.nuclear-power.com/nuclear-power/reactor-physics/nuclear-fission-chain-reaction/resonance-escape-probability/`.

[25] D. E. C. C. J. C. R. Procassini and R. C. Little. Static and dynamic criticality: Are they different? Technical report, U.S. Department of Energy, Lawrence Livermore National Laboratory, 2003.

[26] B. L. SJENITZER. *The Dynamic Monte Carlo Method for Transient Analysis of Nuclear Reactors.* PhD thesis, Delft University of Technology, Delft, the Netherlands., 2013.

[27] H. M. Valtavirta, V. and J. Leppänen. Delayed neutron emission model for time dependent simulations with the serpent 2 monte carlo code - first results. Technical report, PHYSOR 2016, Sun Valley, Idaho, 2016.

[28] V. Valtavirta. *Development and applications of multiphysics capabilities in a continuous energy Monte Carlo neutron transport code.* PhD thesis, Aalto University, School of Science, Department of Applied Physics, Aalto, Finland., 2017.

[29] T. V. Ville Valtavirta, Jaakko Leppänen. Coupled neutronics-fuel behavior calculations in steady state using theserpent2 monte carlo code. *Elsevier*, 2016.

[30] E. F. H. H. L. C. L. M. K. Ville Valtavirta (VTT), Anthime Farda (CEA). Specifications for the reactivity transients scenarios in the four smr cores. Technical report, VTT, 2020.

A | SERPENT input file

Criticality simulation

```
% --- NAME
set title "NuScale Power LLC"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link additional input files  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

include "/nfs/users/mazzoni/serpent/input/pool_assemblyB01_sqc"
include "/nfs/users/mazzoni/serpent/input/fuelrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/emptyposition_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/guidetube_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/controlrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/materials_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/lattice_assemblyB01"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control rod movement  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Transformation for the control rod movement
% The values given are transformations in (x, y, z)
% Transformation of 187.96 means fully withdrawn

trans u CRSH3 0.0 0.0 187.96

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run parameters  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Boundary condition (1 = black, 2 = reflective, 3 = periodic)
set bc 2

% --- Xenon options, without xenon in this case
set xenon 0

% --- Power normalization with power density (in kW/g, of the entire assembly)
set powdens 17.081e-3

% --- Set the volumes of material regions
set mvol
fuel250         0 27315.41988E+00

```

```
% --- Set number of neutrons and cycles, and source definition to help the sampling
set pop 100000 200 50
src external_source sz 0.0 200.0

% --- Shannon entropy
set his 1
set entr 100 100 20

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Detectors, Meshes, Plots definition  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Two-dimensional mesh (total fission heat deposition) for assembly (radial)
det radpow_dist
dr -8 void
dz -20 240 1
dx -10.7518 10.7518 19
dy -10.7518 10.7518 19

% --- Two-dimensional mesh (total fission heat deposition) for assembly (axial)
det axpow_dist
dr -8 void
dx -10.7518 10.7518 1
dy -10.7518 10.7518 1
dz -41 260 1

% --- Two-dimensional mesh (flux distribution) for assembly
det flux dx -10.7518 10.7518 30 dy -10.7518 10.7518 30
de MyEnergyGrid1
ene MyEnergyGrid1 1 1e-11 0.625e-6 10e-3 2e1

% --- Neutron flux in the energy spectrum in the whole assembly
det EnergyDetectorAssembly
de MyEnergyGrid
ene MyEnergyGrid 3 10000 1e-11 2e1

% --- Neutron flux in the energy spectrum in the fuel of the assembly
det EnergyDetectorFuel
dm fuel250
de MyEnergyGrid

% --- Neutron flux in the energy spectrum in the water of the assembly
dm water
de MyEnergyGrid

% --- XY-meshplot and YZ-meshplot (3 and 1), 5000 by 5000 pixels and cover the assembly
    geometry
mesh 3 5000 5000 0 -10.7518 10.7518 -10.7518 10.7518 0 200
mesh 1 5000 5000 0 -10.7518 10.7518 -10.7518 10.7518 -21 300

% --- Detector that calculates the elastic scattering reaction rate in the system
%     Name of the detector is elastic.
%     The detector uses response number -3 (total elastic scattering cross section)
%     of the material at the interaction site (keyword: void)
det elastic dr -3 void
mesh 8 2 elastic 3  5000  5000
```

```

% --- A meshplot that shows the value of a detector response throughout the geometry (type 8)
%     The colormap that is used is "cold" (colormap 2)
%     The detector linked to this meshplot is named "elastic"
%     Orientation is xy (3), and output size is 200 by 200 pixel
mesh 8 2 elastic 3  5000  5000

% --- Detector that calculates the capture reaction rate in the system
%     Name of the detector is capture.
%     The detector uses response number -2 (total capture cross section)
%     of the material at the interaction site (keyword: void)
det capture dr -2 void

% --- A meshplot that shows the value of a detector response throughout the geometry (type 8)
%     The colormap that is used is "cold" (colormap 2)
%     The detector linked to this meshplot is named "capture"
%     Orientation is xy (3), and output size is 200 by 200 pixel
mesh 8 2 capture 3  5000  5000

% --- Cross sections plot
set xsplot 10000 1e-11 2e1

% --- XY-plot of assembly
plot 3 5000 5000 100 -10.7518 10.7518 -10.7518 10.7518

% --- YZ-plot of assembly
plot 1 5000 5000 0 -30 30 -21 250

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Sets up the creation of an initial source to be used in a dynamic simulation with delayed neutron emission  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Save neutron and precursor source to "./source"-files
set savesrc "./source"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optional setting  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Do not generate group constants
set gcu -1

% --- Do not use implicit capture, nxn, or fission
set impl 0 0 0
```

Listing A.1: Serpent Input File – NuScale Configuration

B | SERPENT input file

Dynamic simulation

```
% --- NAME
set title "NuScale Power LLC"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link additional input files  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

include "/nfs/users/mazzoni/serpent/input/pool_assemblyB01_sqc"
include "/nfs/users/mazzoni/serpent/input/fuelrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/emptyposition_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/guidetube_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/controlrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/TOTAL_BORO_REACTIVITY_INSERTION/
    materials_assemblyB01_total_boron_reactivity_insertion"
include "/nfs/users/mazzoni/serpent/input/lattice_assemblyB01"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control rod movement  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Transformation for the control rod movement
% The values given are transformations in (x, y, z)
% Transformation of 127.0 means rods raised of 100 steps

trans u CRSH3 0.0 0.0 127.0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run parameters  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Boundary condition (1 = black, 2 = reflective, 3 = periodic)
set bc 2

% --- Xenon options, without xenon in this case
set xenon 0

% --- Power normalization with power density (in kW/g, of the entire assembly)
set powdens 17.081e-3

% --- Set the volumes of material regions
set mvol
fuel250         0 27315.41988E+00
```

```
% --- Increase size of precursor buffer
set pbuf 10000

% --- Set up time structures for simulation and detectors:

%     1 time interval for simulation (no population control)
tme simutime 2 1 0 1

%     100 time bins for detector
tme dettime 2 100 0 1

%     1 time bin for Energy Detector Assembly at the start of the transient
tme dettime_start 2 1 0 0.1

%    1 time bin for Energy Detector Assembly at the end of the transient
tme dettime_end 2 1 0.9 1

% --- Neutron population: in external source mode with time binning for dynamic mode
set nps 1200000 600 simutime

% --- Shannon entropy
set his 1
set entr 100 100 20

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Detectors, Meshes, Plots definition  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Neutron population as a function of time
det 1 dr -15 void di dettime

% --- Fission energy deposition as a function of time
det 2 dr -8 void di dettime

% --- Neutron flux in the energy spectrum in the whole assembly at the start of the transient
det EnergyDetectorAssemblyStart
di dettime_start
de MyEnergyGrid
ene MyEnergyGrid 3 500 1e-11 2e1

% --- Neutron flux in the energy spectrum in the whole assembly at the end of the transient
det EnergyDetectorAssemblyEnd
di dettime_end
de MyEnergyGrid

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Links previously generated steady state source distributions to be used in a transient simulation with delayed neutron emission  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Link source (use point-wise precursor tracking)
set dynsrc "./source" 1

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optional setting  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Do not generate group constants
set gcu -1

% --- Do not use implicit capture, nxn, or fission
set impl 0 0 0
```

Listing B.1: Serpent Input File – NuScale Configuration

C | SERPENT input file - PYTHON code

Coupled simulation

```
% --- NAME
set title "NuScale Power LLC"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Link additional input files  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

include "/nfs/users/mazzoni/serpent/input/pool_assemblyB01_sqc"
include "/nfs/users/mazzoni/serpent/input/fuelrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/emptyposition_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/guidetube_assemblyB01"
include "/nfs/users/mazzoni/serpent/input/controlrod_assemblyB01"
include "/nfs/users/mazzoni/serpent/SIMULAZIONI_TESI/COUPLING_BORO_ShortTransient/200_PCM
    /STEADY_STATE/materials_assemblyB01"
include "/nfs/users/mazzoni/serpent/COUPLING_25STEP/lattice_assemblyB01_coupling"

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control rod movement  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Transformation for the control rod movement
% The values given are transformations in (x, y, z)
% Transformation of 127.0 means rods raised of 100 steps

trans u CRSH3 0.0 0.0 127.0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run parameters  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% --- Boundary condition (1 = black, 2 = reflective, 3 = periodic)
set bc 2

% --- Xenon options, without xenon in this case
set xenon 0

% --- Power normalization with power density (in kW/g, of the entire assembly)
set powdens 17.081e-3

```

```
% --- Set the volumes of material regions
set mvol
fuel250         0 27315.41988E+00

% --- Increase size of precursor buffer
set pbuf 10000

% --- Set up time structures for simulation:

%     100 time interval for simulation (in coupled simulation there is always population control)
tme simutime 2 100 0 1

% --- Neutron population: in external source mode with time binning for dynamic mode
set nps 400000 10 simutime

% --- Maximum number of coupled calculation iterations set to 2 (more than 2 Serpent has memory problems)
set ccmaxiter 2

%%%
```

Materials input

In the context of coupled neutronics-thermal feedback simulations, only the materials directly involved in the feedback process require modification compared to uncoupled simulations. For this reason, only the definitions of the `fuel250` and `water` materials are reported below, as they are the most relevant. All other material definitions remain unchanged with respect to the uncoupled configuration.

In particular, for the `fuel250` material, the keyword `tft` must be enabled. This command activates the on-the-fly Temperature-Dependent Material Sampling (TMS) treatment and is necessary for multi-physics coupling. It defines the valid temperature range (minimum and maximum in Kelvin) over which the cross sections are sampled and interpolated during the simulation. This is crucial for properly capturing the temperature feedback in the fuel.

The `water` material is also reported to show how the concentration of boron-10 and boron-11 isotopes can be adjusted depending on the desired reactivity insertion. This allows for simulating different boron dilution scenarios while maintaining a physically consistent coolant composition.

```
/*************************
 * Material definitions *
 *************************/

/* UOX */

 % NUSCALE 2,50 wt-% enriched uranium dioxide fuel
%      Density of the fuel 10.5216 g/cm3.

mat fuel250   -10.5216 tft 531.5 2000.0 rgb 35 225 73
92235.03c   5.94084E-04
92238.03c   2.28766E-02
 8016.03c   4.69414E-02

%Water in active core, T 531.5 K, boron 1000 ppm, from SPECS  : Specs+
    Serpent_reference_results excel files (PANTHER/WIMS)
mat water                   sum
                            moder lwtr 1001
                            rgb 200 200 255
1001.03c     5.02932E-02
O-16.03c     2.51573E-02
B-10.03c     1.7159389071447135e-05
B-11.03c     6.906908370681679e-05

therm lwtr 550 lwe7.10t lwe7.12t
```

Listing C.2: Serpent Input File – Materials definition

Lattice input

In coupled simulations with thermal feedback, the full geometry of the assembly lattice must be explicitly defined, as symmetry operations such as reflections and translations are not supported in this simulation mode. This limitation is due to the need for consistent and complete spatial discretization, which is essential for the proper exchange of information between the neutronics and thermal-hydraulic solvers.

In contrast, during criticality calculations or uncoupled dynamic simulations, symmetry operations (e.g., via the `usym` card) can be used to reduce computational cost by defining only a portion of the lattice and replicating it through reflections and rotations. However, in the coupled mode, each region of the geometry must be explicitly described.

Therefore, the full 19×19 lattice is defined explicitly in the input, as shown below, to ensure compatibility with the coupled multi-physics framework.

```
/************************
 * Assembly definitions *
 ************************/

% 19x19 lattice SH3 with 2.50 wt% U235
lat latB011 1 0.0 0.0 19 19 1.2598
w0 w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 SH3 B01 B01 B01 B01 B01 B01 B01 B01 B01 SH3 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 SH3 B01 B01 SH3 B01 B01 GT  B01 B01 SH3 B01 B01 SH3 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 SH3 B01 B01 B01 B01 B01 B01 B01 B01 B01 SH3 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 SH3 B01 B01 SH3 B01 B01 SH3 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 B01 w0
w0 w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0  w0

%set usym latB011 3 2 0.0 0.0 270 45
```

Listing C.3: Serpent Input File – Lattice definition

Python solver

```
#################################################################
#                                                               #
```

```
#           Minimal Serpent Coupling Script v 0.3            #
#           For transient example                            #
#                                                            #
# Created by:  Ville Valtavirta              2016/05/06      #
# Modified by: Giada Mazzoni               2024/10/26        #
#                                                            #
##############################################################

import os
import signal
import math
import time

# Path to Serpent executable

sssexe = '/nfs/soft/serpent/2.2.1/src/sss2'

#######################################################
# Create the Serpent input-file for this run          #
# (process id or communication file must be appended) #
#######################################################

# Open original input for reading

file_in = open('./assemblyB01_coupling','r')

# Open a new input file for writing

file_out = open('./coupledinput','w')

# Write original input to new file

for line in file_in:
    file_out.write(line)

# Close original input file

file_in.close()

# Append signalling mode

file_out.write('\n')
file_out.write('set comfile com.in com.out\n')

# Append interface names

file_out.write('\n')
file_out.write('ifc cool.ifc\n\n')
file_out.write('ifc fuel.ifc\n')

# Close new input file

file_out.close()

# Write initial coolant temperatures and densities

```

```
nodes = 44
T_c_0 = 531.5 #K
T_f =  [
    587.1, 603.3, 632.2, 657.6, 681.2, 703.6, 724.8, 744.5, 762.6, 779.0,
    791.0, 797.5, 795.2, 809.4, 814.8, 816.7, 816.8, 815.2, 812.1, 807.3,
    801.0, 793.1, 782.1, 764.7, 756.9, 746.2, 731.7, 715.7, 698.6, 680.5,
    651.5, 626.7, 613.5, 605.5, 599.5, 595.5, 593.1, 591.0, 589.6, 588.7,
    588.1, 587.7, 587.4, 587.2
] #K

##############################################
# Write the initial coolant interface file   #
# (Coolant conditions will be held constant) #
##############################################

file_out = open('./cool.ifc','w')

# Write the header line (TYPE MAT OUT)

file_out.write('2 water 2\n')

# Write the output line (OUTFILE NZ ZMIN ZMAX NR)

file_out.write('coolifc.out\n')           #10 0 200 1

# Write the mesh type

file_out.write('1\n')

# Write the mesh size (5:X0 Y0 PITCH ZMIN ZMAX NX NY NZ; 1:NX XMIN XMAX NY YMIN YMAX NZ
    ZMIN ZMAX))

file_out.write('1 -10.7518 10.7518 1 -10.7518 10.7518 44 0.0 200.0\n')

# Write initial coolant temperatures and densities

for i in range(nodes):
    file_out.write(f'0.07548775253000001  {T_c_0}\n')

# Close interface file

file_out.close()

##############################################
# Write the initial fuel interface file      #
# (Fuel temperature will be updated)         #
##############################################

file_out = open('./fuel.ifc','w')

# Write the header line (TYPE MAT OUT)

file_out.write('2 fuel250 0\n')

# Write the mesh type

```

```
file_out.write('1\n')

# Write the mesh size (X0 Y0 PITCH ZMIN ZMAX NX NY NZ)

file_out.write('1 -10.7518 10.7518 1 -10.7518 10.7518 44 0.0 200.0\n')
# Write initial fuel temperatures and densities

for i in range(nodes):
    file_out.write(f'-10.5216 {T_f[i]}\n')

# Close interface file

file_out.close()

# Archive the initial fuel interface

# Archive the initial fuel interface
output_dir2 = "./fuel_interfaces/"
os.makedirs(output_dir2, exist_ok=True)
os.system('cp ./fuel.ifc {}/fuel.ifc0'.format(output_dir2))

################################
# Start the Serpent simulation #
################################

# Create a command string that will start the Serpent simulation

runcommand = 'mpirun -np 1 '+sssexe+' -omp 3 ./coupledinput > ./coupledinput.log 2>&1 &'

# Execute the command string

os.system(runcommand)

##########################################
# Initialize the fuel temperature solution #
##########################################

TBOI = []
TEOI = []

#for i in range(nodes):
#    TBOI.append(531.5)
#    TEOI.append(531.5)

TBOI = T_f.copy()
TEOI = T_f.copy()

# Reset time step

curtime = 0

#########################
# Loop over time steps #
#########################

simulating = 1
```

```
170
171 while simulating == 1:
172
173     ##########################
174     # Picard iteration loop #
175     ##########################
176
177     iterating = 1
178
179     while iterating == 1:
180         ###################
181         # Wait for signal #
182         ###################
183
184         sleeping = 1
185
186         while sleeping == 1:
187
188             # Sleep for two seconds
189
190             time.sleep(2)
191
192             # Open file to check if we got a signal
193
194             fin = open('./com.out','r')
195
196             # Read line
197
198             line = fin.readline()
199
200             # Close file
201
202             fin.close()
203
204             # Check if the line is not empty and can be converted to int
205             if line:  # Ensure the line is not empty
206                 signal_value = int(line)
207
208                 # Check signal
209                 if signal_value != -1:
210                     if signal_value == signal.SIGUSR1.value:
211                         # Got the signal to resume
212                         sleeping = 0
213
214                     elif signal_value == signal.SIGUSR2.value:
215                         # Got the signal to move to the next time point
216                         iterating = 0
217                         sleeping = 0
218
219                     elif signal_value == signal.SIGTERM.value:
220                         # Got the signal to end the calculation
221                         iterating = 0
222                         sleeping = 0
223                         simulating = 0
224
225                     else:
```

```
                        # Unknown signal
                        print("\nUnknown signal read from file, exiting\n")
                        quit()

                # Reset the signal in the file

                file_out = open('./com.out','w')

                file_out.write('-1')

                file_out.close()
            else:
                # If the line is empty or not a valid number
                print("Error: Invalid value in 'com.out', retrying...")

        # Check if simulation has finished and break out of iterating
        # loop

        if (simulating == 0):
            break

        ###########################
        # Read power distribution #
        ###########################

        # Reset power distribution

        P = []

        # Open output file

        file_in = open('./coolifc.out','r')

        # Loop over output file to read power distribution

        for line in file_in:
            # Split line to values

            strtuple = line.split()

            # Store power

            P.append(float(strtuple[0]))

        file_in.close()

        ###########################
        # Calculate TH-solution   #
        ###########################

        print("\nStarting calculation of new temperature\n")

        # Fuel specific heat capacity

        cp = 300 # J/(kg*K)

```

```
        # Calculate EOI temperatures at (nz) axial nodes
        # No heat transfer, just deposition

        for i in range(nodes):

            # Calculate EOI temperature based on BOI temperature
            # and energy deposition during current time interval

            # Calculate mass of this node (in kg)

            m = (math.pi*0.4750**2*200.0/44)*10.5216*1e-3*264              #number of
                fuel pins=264

            # Calculate initial heat in this axial node

            Q = TBOI[i]*(cp*m)

            # The interface output is joules in case of time dependent
            # simulation, no need to multiply with time step

            dQ = P[i]

            # Calculate new temperature based on new amount of heat

            TEOI[i] = (Q + dQ)/(cp*m)
            print("\n New temperature: {} \n".format(TEOI[i]))

        ###########################
        # Update interface        #
        ###########################

        print("\nStarting updating interface\n")

        file_out = open('./fuel.ifc','w')

        # Write the header line (TYPE MAT OUT)

        file_out.write('2 fuel250 0\n')

        # Write the mesh type

        file_out.write('1\n')

        # Write the mesh size (X0 Y0 PITCH ZMIN ZMAX NX NY NZ)

        file_out.write('1 -10.7518 10.7518 1 -10.7518 10.7518 44 0.0 200.0\n')

        # Write updated fuel temperatures

        for i in range(nodes):
            # Use the base density throughout the simulation
            # Write density and temperature at this layer

            file_out.write('-10.5216 {}\n'.format(TEOI[i]))

        file_out.close()
```

```
        print("\nFinished to update interface\n")

        ############################
        # Signal Serpent (SIGUSR1) #
        ############################

        ##########################################################
        # We could actually check if the temperature changed by  #
        # a significant margin and send a SIGUSR2 to indicate    #
        # that Serpent should not iterate the transport solution #
        ##########################################################

        file_out = open('./com.in','w')

        file_out.write(str(signal.SIGUSR1.value))

        file_out.close()

    # Increment time step

    curtime += 1

    ########################
    # Do some archiving... #
    ########################

    # Copy the interface and interface output to
    # archive files for later examination
    # Note: These are EOI fields

    output_dir2 = "./fuel_interfaces/"
    os.makedirs(output_dir2, exist_ok=True)
    os.system('cp ./fuel.ifc {}/fuel.ifc{:d}'.format(output_dir2, curtime))

    output_dir1 = "./cool_interfaces/"
    os.makedirs(output_dir1, exist_ok=True)
    os.system('cp ./coolifc.out {}/coolifc.out{:d}'.format(output_dir1, curtime))

    output_dir3 = "./coupledinput_res/"
    os.makedirs(output_dir3, exist_ok=True)
    os.system('cp ./coupledinput_res.m {}/coupledinput_res{:d}.m'.format(output_dir3,
        curtime))

    output_dir4 = "./fuel_Tconv/"
    os.makedirs(output_dir4, exist_ok=True)
    os.system('cp ./fuel.ifc_Tconv0.m {}/fuel.ifc_Tconv{:d}.m'.format(output_dir4,
        curtime))

    output_dir5 = "./cool_Pconv/"
    os.makedirs(output_dir5, exist_ok=True)
    os.system('cp ./cool.ifc_Pconv0.m {}/cool.ifc_Pconv{:d}.m'.format(output_dir5,
        curtime))

    #output_dir6 = "./coupledinput_det/"
    #os.makedirs(output_dir6, exist_ok=True)
```

```
    #os.system('cp ./coupledinput_det{:d}.m {}/coupledinput_det{:d}.m'.format(curtime-1,
        output_dir6, curtime-1))

    ####################################
    # Check if simulation has finished #
    ####################################

    if (simulating == 0):
        break

    ############################
    # Moving to next time step #
    ############################

    # Copy EOI temperatures to BOI vector

    for i in range(nodes):
        TBOI[i] = TEOI[i]

    ##################################################
    # Here we could calculate an initial guess for   #
    # the the EOI temperatures of the next time      #
    # interval and update the interface.             #
    # But to keep the script minimal, we won't do it #
    ##################################################

    ############################
    # Signal Serpent (SIGUSR1) #
    ############################

    file_out = open('./com.in','w')

    file_out.write(str(signal.SIGUSR1.value))

    file_out.close()
```

Listing C.4: Python code – Thermodynamic solver

List of Figures

1.1 Thesis flow chart. 3
1.2 Schematic view of a NuScale reactor module [30]. 4
1.3 Placement of burnable absorbers and guide tubes in fuel assemblies [30]. . 5
1.4 Loading pattern of the initial core (left) and positioning of the control rod banks (right) [30]. 6
1.5 Sketch of the sampling of a random variable x. 9
1.6 Sketch of the difference between accuracy vs. precision in MC neutronics. 11
1.7 Thermal, Epithermal and Fast neutron fluxes inside the fuel assembly. . . . 16
1.8 Fission rate and thermal flux colormaps. 17
1.9 Thermal, Epithermal and Fast neutron fluxes inside the fuel assembly. . . . 17

2.1 Position and direction variables to characterize the neutrons distribution [6]. 19
2.2 Scheme of precursors decay approach within Serpent [7]. 35
2.3 Two-step approach for transient calculations 36
2.4 Energy dependence of the neutron flux in the assembly 37
2.5 Most relevant nuclear cross sections plotted with Serpent 2. 38
2.6 An increase in temperature from T1 to T2 causes the broadening of spectral lines of resonances. Although the area under the resonance remains the same, the broadening of spectral lines causes an increase in neutron flux in the fuel $\phi_f(E)$, increasing the absorption as the temperature increases [24]. 39

3.1 Geometry plots of the Serpent model. 44
3.2 Evolution of Shannon entropy and k_{eff} over the simulation cycles. 46
3.3 Radial power distributions for reflective (a) and black (b) boundary conditions. 48
3.4 Axial power distributions along the assembly for reflective (a) and black (b) boundary conditions. 49

4.1 Step insertion of reactivity and Comparison of Cinecolo with Analitical solutions results. 64

4.2 Step insertion of reactivity and Comparison of Cinecolo with Analitical and Numerical solutions results. 66
4.3 Ramp insertion of reactivity and Comparison of Cinecolo with Numerical solution result. 67
4.4 Two-step approach for transient calculations [7]. 69
4.5 Schematic representation of precursor weight evolution during a time interval in a transient simulation [14]. 70
4.6 The scheme for performing a dynamic Monte Carlo calculation [26]. 78
4.7 Comparison of neutron flux evolution following the 535 pcm reactivity insertion. 80
4.8 Neutron flux evolution during the 0.1 s transient with and without population control enabled. 82

5.1 The two level coupling scheme in Serpent 2 [29]. 87
5.2 Solution flow in externally coupled calculations using the finalized coupled calculation routines implemented in Serpent 2 [14]. 89
5.3 Discretization of the fuel assembly within the multiphysic interface. 99
5.4 Transient two–step approach for coupled calculations for Serpent-Thermodynamic solver. 100
5.5 Mesh of the assembly showing the fuel temperature distribution. 103
5.6 Scheme of the code flow for coupled transient calculations. 103
5.7 Lumped Parameter Model for the Adiabatic Fuel Rod System. 107
5.8 Interpolation of reactivity values as a function of fuel temperature to obtain α_f. 113
5.9 Simulation with a smaller neutron population, showing higher statistical error. 115
5.10 Simulation with a larger neutron population, showing more precise results. 115
5.11 Time evolution of reactivity, showing a decrease over time due to the thermal feedback mechanism. 116
5.12 Power evolution over time for each axial node. 116
5.13 Fuel temperature evolution over time for each axial node. 116
5.14 Power evolution over time for each axial node. 117
5.15 Fuel temperature evolution over time for each axial node. 117
5.16 Power evolution over time for the first simulation (1 second). 119
5.17 Power evolution over time for the second simulation (0.2 seconds). 119
5.18 Time evolution of reactivity, showing a decrease over time due to fuel thermal feedback. 120

5.19 Temperature evolution from Serpent simulation. 120
5.20 Temperature evolution from the numerical solution. 120
5.21 Power evolution from Serpent simulation. 121
5.22 Power evolution from the numerical solution. 121
5.23 Power evolution over time for each axial node. 121
5.24 Temperature evolution over time for each axial node. 121
5.25 Evolution of ϵ for the relaxed power distribution. 122
5.26 Evolution of ℓ_n^2-norm for the relaxed power distribution. 122
5.27 Evolution of ϵ for the temperature distribution. 122
5.28 Evolution of ℓ_n^2-norm for the temperature distribution. 122
5.29 Power evolution for all three reactivity insertion cases: 200 pcm, 450 pcm, and 1000 pcm. 125
5.30 Time evolution of β_{eff} during two transient simulations with different time resolutions. The left figure shows an initial drop in β_{eff}, followed by a gradual recovery toward the steady-state value. The right figure shows a low initial value of β_{eff} that progressively increases. 126
5.31 Control rod positions used in the transient simulations. 130
5.32 Time evolution of reactor power and reactivity during the rod extraction transient. 131
5.33 Axial power distribution over time for each axial node (Serpent vs numerical). 132
5.34 Axial fuel temperature distribution over time for each axial node (Serpent vs numerical). 132
5.35 Time evolution of reactor power for each axial node. 133
5.36 Comparison of power evolution in the upper-mid and lower CR configuration. 134

List of Tables

1.1 Thermal boundary conditions for the NuScale reactor core. 5
1.2 Materials used for different components of the NuScale core model. 7
1.3 Fuel characteristics of assembly B01. 8

2.1 Main neutronic quantities and their corresponding formulas and units, including neutron current density. 20
2.2 Indicative values (taken by Agostin's work [1]) for the six-factor formula parameters obtained with Serpent for NuScale SMR under hot conditions. The associated uncertainty ($\pm 2\sigma$) is on the order of 10^{-4}. 25

3.1 Comparison of criticality results between Panther and Serpent (ARO configuration, T = 900 K, ENDF/B-VII.1). 50
3.2 Kinetic Parameters extracted using the IFP method, reported with a statistical uncertainty of $\pm 2\sigma$. 51
3.3 Precursor Group Parameters using the IFP method, reported with a statistical uncertainty of $\pm 2\sigma$. 52
3.4 Six-Factor Formula parameters extracted for the three different cases (ARI, ARO, and Critical 100 Steps), reported with a statistical uncertainty of $\pm 2\sigma$. 52

4.1 Transient simulation parameters – 535 pcm step reactivity insertion 79
4.2 Simulation parameters – 0.1 s transient with population control. 81

5.1 Coupled calculation flow (Euler method, `ccmaxiter = 2`) 92
5.2 Temperature distribution at axial nodes in Kelvin 101
5.3 System configuration for steady-state criticality simulation 102
5.4 Effective Resonance Integral I_{eff} and Resonance Escape Probability p for Varying Fuel Densities 111
5.5 Effective Resonance Integral I_{eff} and Resonance Escape Probability p for Varying Fuel Rod Diameters 112
5.6 Simulation Parameters for Boron Dilution Transients 114
5.7 Simulation Parameters for Boron Dilution Transients 118

5.8 Simulation Parameters for 1ms Transients with Different Reactivity Insertions 124
5.9 Simulation parameters for rod extraction transient. 131
5.10 Comparison of simulation parameters for two rod extraction transients. . . 134

6.1 Comparison between Monte Carlo and Point Kinetics for studying nuclear system transients. 138

List of Symbols

Variable	Description	SI Unit
C	Precursor concentration	n/m^3
D	Diffusion coefficient	m^2/s
K	Multiplication factor	-
v	Neutron speed	cm/s
σ_t	Total microscopic cross section	m^{-1}
σ_s	Scattering microscopic cross section	m^{-1}
σ_f	Fission microscopic cross section	m^{-1}
ϕ	Neutron flux	$n/m^2/s$
Λ	Neutron generation time	s
λ	Decay constant	1/s
β	Beta fraction (delayed neutrons)	-
ρ	Reactivity	-
P	Power production	W
α_f	Fuel feedback coefficient	-
T	Temperature	K
T_f	Fuel temperature	K
T_c	Coolant temperature	K
S	Source term	$n/m^3/s$
ψ	Angular flux	$n/m^2/s$
∇	Gradient operator	m^{-1}
n	Neutron count per unit volume	n/m^3
J	Neutron current density	$n/m^2/s$
Σ_t	Total macroscopic cross section	m^{-1}
Σ_s	Scattering macroscopic cross section	m^{-1}
χ	Fission spectrum	-

Continued from previous page		
Variable	**Description**	**SI Unit**
ν	Neutrons per fission	neutrons/fission
Σ_f	Fission macroscopic cross section	m^{-1}
C_i	Precursor concentration at position r and time t	n/m^3
η	Neutrons produced from fission absorption in fuel	-
f	Absorption in fuel (total absorption)	-
P_F^{NL}	Probability of fast neutrons not leaking from the reactor	-
P_T^{NL}	Probability of thermal neutrons not leaking from the reactor	-
ξ	Random number for neutron track length sampling	-
x	Neutron track length	cm
W_n	Neutron weight	-
W_n^{new}	Updated neutron weight after absorption	-
W_n^{old}	Previous neutron weight before absorption	-
w_{tal}	Neutron weight after time absorption	-
w_{emit}	Neutron weight contributing to delayed neutron emission	-
w_0	Initial precursor weight at time t^*	-
B^2	Geometric buckling	m^{-2}
t	Time	s

Acknowledgements

I would like to thank my thesis supervisor, Bart, for his guidance and support throughout the project. His advice and feedback were very helpful in completing my work.

I also want to thank the entire team at Tractebel for the opportunity to work with them during my 7-month internship in Brussels. This experience was very valuable, as it allowed me to learn new skills and gain knowledge from industry experts. I am grateful for the warm welcome and the opportunity to be part of the team.

Finally, I would like to thank my professor Antonio Cammi, for his support and valuable advice, as well as for giving me the chance to get more involved in the professional world.