Analysis and Improvement of the bRAPID Algorithm and its Implementation

Jacob Benjamin Bartel

Thesis submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Nuclear Engineering

Alireza Haghighat, Chair
Mark Pierson
Roop Mahajan

June 5, 2019
Arlington, Virginia

Keywords: neutron transport, pin-wise burnup, fission matrix, material composition, RAPID, boundary correction

Analysis and Improvement of the *b*RAPID Algorithm and its Implementation

Jacob Benjamin Bartel

(ABSTRACT)

This thesis presents a detailed analysis of the *b*RAPID (burnup for RAPID – Real Time Analysis for Particle transport and In-situ Detection) code system, and the implementation and validation of two new algorithms for improved burnup simulation. *b*RAPID is a fuel burnup algorithm capable of performing full core 3D assembly-wise burnup calculations in real time, through its use of the RAPID Fission Matrix methodology. A study into the effect of time step resolution on isotopic composition in Monte Carlo burnup calculations is presented to provide recommendations for time step scheme development in *b*RAPID. Two novel algorithms are implemented into *b*RAPID, which address: i) the generation of time-dependent correction factors for the fission density distribution in boundary nuclear fuel assemblies within a reactor core; ii) the calculation of pin-wise burnup distributions and isotopic concentrations.

Time step resolution analysis shows that a variable time step scheme, developed to accurately characterize important isotope evolution, can be used to optimize burnup calculations and minimize computation time. The two new algorithms have been benchmarked against the Monte Carlo code system Serpent. Results indicate that the time-dependent boundary correction algorithm improves fission density distribution calculations by including a more detailed representation of boundary physics. The pin-wise burnup algorithm expands *b*RAPID capabilities to provide material composition data at the pin level, with accuracy comparable to the reference calculation. In addition, wall-clock time analyses show that burnup calculations performed using *b*RAPID with these novel algorithms require a fraction of the time of Serpent.

Analysis and Improvement of the *b*RAPID Algorithm and its Implementation

Jacob Benjamin Bartel

(GENERAL AUDIENCE ABSTRACT)

Fuel burnup modeling is an important aspect of nuclear reactor design that provides information about the energy extracted (called burnup) and isotopes created or used in the fuel of a reactor over time. A reactor core is a collection of fuel assemblies, and assemblies are simply a bundle of fuel pins, which contain nuclear fuel such as Uranium. The desire for accurate and fast computer codes to calculate fuel burnup rises each year as engineers working in reactor core design seek to arrange fuel assemblies in an optimal pattern to extract the most energy. State of the art burnup codes exist, however they have certain limitations due to their underlying methodologies.

To satisfy this need, the *b*RAPID algorithm was developed by the Virginia Tech Transport Theory Group (VT^3G). *b*RAPID is a new methodology capable of performing full core fuel burnup calculations in real time. *b*RAPID is able to calculate the criticality and burnup distribution of a reactor orders of magnitude faster than comparable algorithms, while addressing many of the shortcomings seen in other burnup codes.

In this thesis, studies of standard burnup codes are conducted in order to aid in *b*RAPID analysis: first in the form of a detailed study of the reference Monte Carlo model used in this thesis, and secondly in an investigation of the effect of time step selection– or the time intervals used in burnup calculations– on isotope concentration. Both of these studies are conducted using the benchmark code system, Serpent, with the latter study providing useful insight that can be used for *b*RAPID database development. This thesis then presents two new algorithms for *b*RAPID that expand its capability and improve performance. First, an algorithm to more accurately simulate the boundary regions of the core– called the time dependent boundary correction algorithm– is presented and benchmarked. Next, an algorithm to expand *b*RAPID capability from assembly-wise to pin-wise burnup calculations is implemented and tested. These two algorithms are benchmarked against the Serpent Monte Carlo based burnup code.

Acknowledgments

I would like to thank Dr. Alireza Haghighat, my research advisor, whose breadth of knowledge and mentorship have been essential to completing this thesis. I thank the rest of my committee, Dr. Roop Mahajan and Dr. Mark Pierson, for their assistance and interest. My research would not have been possible without the work done by Dr. Nathan Roskoff, an excellent developer and engineer.

I would also like to thank my colleagues Vince Wang, Valerio Mascolino, and Jimmy Butler, whose knowledge of transport theory is surpassed only by their sense of humor.

I owe a great debt of gratitude to my best friend and partner, K'Ehleyr Thai, whose patience, love, and support has sustained me throughout my education. Additionally, thank you to my friends and roommates who have provided much needed relief from the stresses of graduate school. Last but not least, thank you to my parents, Luisa and Michael Bartel, and my brother Matthew, who have supported me every step of the way through this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fuel burnup is a quantity that describes the usage of a fuel rod in a nuclear reactor, in units of energy extracted per unit mass. The primary fuel (initially) in a nuclear fuel rod is Uranium-235, or ^{235}U. When this fuel is "burned" it undergoes fission, which transmutes elements and releases energy. Over time, the material composition of the fuel changes as elements are depleted and generated. After a certain amount of burnup occurs in a reactor core, some of the fuel assemblies would be removed and dubbed "spent nuclear fuel" or SNF, while lower burned fuel assemblies are shuffled around and strategically placed in order to maximize the efficiency or utilization of the fuel. Modeling this process of depletion and fuel utilization is of great importance to core design engineers. Section 1.1 discusses computer codes commonly used to perform these analyses, however the work in this thesis primarily revolves around the fuel burnup methodology of *b*RAPID [1], a burnup algorithm developed and incorporated into the RAPID (Real-time Analysis for Particle transport and In-situ Detection) code system [2].

For criticality or eigenvalue calculation, RAPID uses the Fission Matrix (FM) methodology to perform particle transport calculations extremely quickly, and accurately. RAPID's FM methodology is based on pre-calculation of its FM coefficients by performing a series of fixed-source Monte Carlo simulations. Then, a pin-wise, 3-D fission density distribution

and a corresponding system eigenvalue are obtained by solving a linear system of equations using the pre-calculated coefficients. The real-time solution capability of RAPID comes from solving this system of equations. RAPID overcomes many of the issues in Monte Carlo based codes, such as a lack of fission source convergence, undersampling, and cycle-to-cycle correlation, as well as issues in deterministic codes, such as inaccuracies due to phase space discretization, by using the FM methodology.

The typical fuel burnup methodology couples a particle transport code (Monte Carlo or deterministic) with a depletion code. *b*RAPID on the other hand, is built on top of the RAPID code system, and thus avoids the drawbacks of typical particle transport methodologies when performing its burnup routine. *b*RAPID works by using a RAPID database of coefficients that are power density and irradiation time dependent. These pre-calculated coefficients are used as needed during the burnup calculation. Initially, coefficients describing a fresh fuel system (unless otherwise specified by the user) are used in a RAPID calculation, which produces the k-eigenvalue of the system, as well as quantities such as the fission source distribution. This fission source distribution is then converted into a power density distribution, which is fed into the next RAPID calculation input, and also used to generate material composition data for that burnup step. This process continues until the last step is reached.

The accuracy of a *b*RAPID calculation is highly dependent on database generation– and database generation requires the use of a Monte Carlo burnup pre-calculation. Thus, in order to improve *b*RAPID, analysis was performed on traditional Monte Carlo techniques in order to provide recommendations for database development. First, in Chapter 4, a sensitivity study analyzing the primary Monte Carlo parameters (number of particle histories, number of active cycles, and number of skip cycles) was performed in order to create a reference model for use in the rest of the thesis. Then, in Chapter 5 a detailed study into the effect of time step resolution on material composition in Monte Carlo burnup calculations was performed. This latter study presents a heuristic method for time step scheme development, and metrics for evaluating the efficiency of said schemes.

*b*RAPID fundamentally runs on the fission source distribution data provided by RAPID. RAPID fission source data comes from solving the FM linear equations described above– and hence, the accuracy of this data is dependent on the methodology used to generate FM coefficients. FM coefficients are generated using a center core environment– meaning there are no boundary effects captured by the coefficients. However, most models do have boundary regions, where assemblies are in contact with a reflector or a different material type. In order to correct for the center-core approximation in models with boundary regions, a "boundary correction factor" methodology was developed, described in Chapter 6. These boundary correction factors (or $bnd(x, y)$) resolve much of the issues with boundary regions in static RAPID problems, but for problems with changing material compositions (such as burnup), there was no algorithm to deal with the changing boundary conditions. Thus, this thesis presents a time dependent boundary correction algorithm to deal specifically with burnup problems. The time dependent boundary corrections ($bnd(x, y, t)$) work similarly to $bnd(x, y)$ corrections– however, the new methodology uses material composition data for each assembly, at each time step, to more accurately reflect the changes in boundary behavior over time. The changes made to the boundary correction methodology are reflected in fission source distribution calculations, presented in Chapter 6.

Originally, *b*RAPID was developed at the assembly level– meaning that assembly-wise power densities and material compositions are used, and assembly-wise burnup data is produced. While this is suitable for most applications, there is an interest in the development of accurate pin-wise burnup codes for core design and fuel management as discussed in Section 1.1. Pin-wise fission density is generated by RAPID at each burnup step automatically, however the original version of *b*RAPID does not utilize this information, rather it uses an assembly-wise power distribution for burnup calculations. This thesis presents a new algorithm to expand the capability of *b*RAPID to 2D pin-wise burnup applications. Similar to the assembly-wise burnup algorithm, during each burnup step the 2D pin-wise fission source distribution is converted to a power density distribution. This data can then be converted to pin-wise burnup and material composition distributions using the same interpolation scheme

described in Chapter 3. As a result, pin-wise burnup data can now be produced by *b*RAPID alongside the standard assembly-wise data outputs if greater detail is desired.

1.1 Literature Review

There have been a number of codes developed capable of performing burnup calculations. As described above, burnup codes typically couple either a Monte Carlo or deterministic particle transport code with a depletion code. The most commonly used burnup tools, along with their transport and depletion codes, are listed in Table 1.1.

Table 1.1 Burnup tools and their transport/depletion codes

Burnup Tool	**Transport**	**Transport Type**	**Depletion**	**Ref.**
MONTEBURNS	MCNP	3D Monte Carlo	ORIGEN CINDER90	[3]
Serpent	Serpent	3D Monte Carlo	Serpent	[4]
SCALE/TRITON	KENO-VI NEWT XSDRNPM	3D Monte Carlo 2D S_N 1D S_N	ORIGEN	[5]
PENBURN	PENTRAN MCNP	3D S_N 3D Monte Carlo	PENBURN	[6]
MVP-BURN	MVP	3D Monte Carlo	BURN	[7]

Transport codes perform an eigenvalue calculation at each burnup step to retrieve flux values that are then fed into the depletion code. This depletion code then uses a solver routine to solve the Bateman equations for the specified time step length. Typical Bateman equation solver routines include the "linear chain method" [8], which splits non-linear chains in the Bateman equation into a set of linear chains that are then solved numerically, or the "matrix exponential method" [9], which solves the Bateman equations in matrix form using an exponential power series approximation. The Bateman equations provide new material compositions for fuel regions, which are then fed back into the next burnup step's particle

transport calculation, and so on.

Monte Carlo based burnup calculations are the "gold standard" for modeling in many ways, as they can use continuous phase space variables including exact model geometries [10]. However, MC calculations can be time consuming, have convergence issues in models with a high dominance ratio, and are known to have cycle-to-cycle correlations which cause issues such as underestimating uncertainties on tally values [11]. Deterministic codes have the potential to be faster and they provide detailed solutions within a discretized phase space. However, due to the large amount of memory required for detailed problems, deterministic codes generally make use of approximations which can lead to inaccuracies. In general, 3D determinstic codes are not used for burnup due to large time requirements [1]. RAPID, and *b*RAPID by extension, uses the FM approach described above to overcome many of the issues with Monte Carlo and deterministic based codes. This approach allows for *b*RAPID to converge on a fission source distribution, k_{eff} and calculate burnup values in a fraction of the time taken by similar Monte Carlo reference calculations.

Accurate pin-wise burnup codes are another area of interest in the nuclear community due to their implications for reactor design and spent fuel material composition prediction [1, 12–16]. Pin-wise calculations grant designers a greater ability to optimize fuel placement and utilization in reactor operation, or to optimize the design of the next generation of reactors by increasing the granularity used for analysis. Processes such as pin-by-pin fuel pattern management are made more feasible by detailed pin-wise burnup calculations [15, 17, 18]. Research has been conducted to improve the pin-wise fuel burnup capabilities of Monte Carlo based codes [19], however even with these improvements, calculations are still constrained by the shortcomings of the Monte Carlo methodology. By utilizing the fission matrix approach, the *b*RAPID algorithm shows promise in overcoming the hurdles other burnup codes in the domain of pin-wise burnup calculations.

Chapter 2

Theory

This Chapter introduces the theoretical foundations of this thesis. Since this thesis primarily focuses on expanding and improving the fuel burnup methodology of the *b*RAPID (burnup for RAPID - Real Analysis for Particle-transport and In-situ Detection) algorithm, the foundations of fuel burnup methodologies– namely, the two principle parts of fuel burnup: neutron transport and depletion will first be described. After presenting eigenvalue particle transport methods, the coupling of the neutron transport and depletion processes in traditional burnup codes will be discussed.

2.1 Linear Boltzmann Equation

The Linear Boltzmann Equation (LBE) is the fundamental governing equation of neutral particle transport. The LBE is a particle balance equation, and this thesis consider only the balance equation for neutrons. Within the phase space ($d^3rdEd\Omega$), the balance of production

and loss of neutrons is expressed by Equation 2.1 [20], the time independent LBE:

$$\underbrace{\hat{\Omega}\cdot\nabla\Psi(\vec{r},E,\hat{\Omega})}_{(1)}+\underbrace{\Sigma_t(\vec{r},E)\Psi(\vec{r},E,\hat{\Omega})}_{(2)}=\underbrace{\int_0^\infty dE'\int_{4\pi}d\Omega'\Sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\Psi(\vec{r},E,\hat{\Omega})+}_{(3)}$$
$$\underbrace{\frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\Omega'\nu(E)\Sigma_f(\vec{r},E')\Psi(\vec{r},E',\hat{\Omega})}_{(4)}+\underbrace{q(\vec{r},E,\hat{\Omega})}_{(5)} \tag{2.1}$$

where $\Psi(\vec{r},E,\hat{\Omega})$ is the angular flux, $\Sigma_t(\vec{r},E)$ is the total macroscopic cross-section, $\Sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})$ is the macroscopic differential scattering cross-section for a neutron with energy E' and direction Ω' scattering into phase space $(dEd\Omega)$ about energy E and angle $\hat{\Omega}$, $\chi(E)$ is the fission neutron spectrum, $\nu(E)$ is the average number of neutrons produced from fission by a neutron of energy E, $\Sigma_f(\vec{r},E')$ is the macroscopic fission cross-section, and $q(\vec{r},E,\hat{\Omega})$ is the independent neutron source.

The equation is split into 5 pieces, each of which describes a different expected particle process [21]:

1. Streaming: Removal of neutrons via flow, or leakage, from the phase space
2. Collision: Removal of neutrons via collision interactions (absorptions or scattering *out of* phase space)
3. Scattering: Neutron production via scattering *into* the phase space
4. Fission: Neutron production due to fission
5. Independent Source: neutrons produced via an independent neutron source

One can see that the left hand side of the equation represents neutron loss, while the right hand side represents neutron production.

2.2 Methods for Solving the LBE

The LBE is unsolvable analytically for most reactor applications in any practical sense [22]. This barrier has lead to the creation of various approximate methods for solving the LBE, which are useful for applications given certain assumptions. These methods fall broadly into three camps: deterministic methods, Monte Carlo methods, and their hybrids.

2.2.1 Deterministic

Deterministic methods solve the LBE by discretizing the phase space variables introduced in Section 2.1 and solving the LBE numerically. These three variables– space (d^3r), energy (E), and angle ($\hat{\Omega}$) require different techniques to discretize, which have been the subject of extensive study.

One of the most common methods for treating the angular phase space in the LBE is the Discrete Ordinates method, also called the S_N method [20, 22–24]. This method involves solving the LBE over a set of discrete angular directions (called ordinates), each of which is given a weighting constant, to create what is known as a "quadrature set". The 'order' of the quadrature set, N, gives rise to $N(N+2)$ corresponding ordinates and weighting constants. One of the main concerns in using the S_N method is identifying the proper order to use, which is highly problem dependent.

Spatial discretization can be achieved by using a "differencing scheme", which breaks the spatial portion of the LBE calculation into "mesh" segments [20, 25]. Cell-averaged flux values can be calculated for each mesh cell using the incoming/outgoing flux values with various differencing schemes, such as the Diamond Differencing (DD) [20], Theta-Weighted (TW) [26], and Directional Theta-Weighted (DTW) [27] methods

Energy is discretized by formulating a set of energy intervals, called "groups", and holding energy and other values (i.e. cross-sections) within that group constant. As with the

other phase space variable discretization techniques described, this so called "multigroup" approach is also highly problem specific. Various methods for creating these cross-section group structures have been developed [28, 29].

Deterministic methods have the benefit of giving detailed solutions for a given problem, generating an abundance of information in each spatial volume defined, and (usually) the LBE calculations converge more quickly than they do in Monte Carlo calculations. However, determinsitic methods also have downsides– discretization of the phase space variables can introduce inaccuracies, and calculations can require large amounts of computer memory for detailed problems.

2.2.2 Monte Carlo

The Monte Carlo method is a statistical approach that uses random numbers to sample probability densities to simulate particle transport, i.e., LBE [20,21]. Rather than attempting to discretize the LBE, the Monte Carlo approach is to simulate particles using continuous phase space variables. The uncertainty from Monte Carlo methods therefore comes from statistical uncertainty, rather than from approximations due to discretization. Monte Carlo simulations are similar to running thousands of real life experiments to calculate a given average quantity within a discretized phase space.

Benefits of the Monte Carlo method include continuous cross-section sampling and the ability to use precise model geometries. For these reasons, Monte Carlo is generally regarded as the most accurate solution if the simulation is properly performed. However, there are various drawbacks associated with the Monte Carlo method as well. For instance, Monte Carlo calculations often take more computation time, as millions of particles need to be simulated and tracked. Less information is also produced by the Monte Carlo method, and so results must be 'tallied' in discrete variables– similar to the deterministic method. In addition, the statistical approach opens the door to undersampling, and the Monte Carlo eigenvalue approach can have trouble with fission source convergence [30].

2.3 The Eigenvalue Formulation

In Section 2.1 the Linear Boltzmann Equation for a generalized problem set was introduced. A more specific application of the LBE is the "eigenvalue" formulation, also known as the criticality formulation.

The LBE for an eigenvalue problem [1, 21] is expressed by

$$H\Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k} F\Psi(\vec{r}, E, \hat{\Omega}) \tag{2.2}$$

where H is the transport operator, and F is the fission operator, given by

$$H = \hat{\Omega} \cdot \nabla + \Sigma_t(\vec{r}, E) - \int_0^\infty dE' \int_{4\pi} d\Omega' \Sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega}) \tag{2.3}$$

$$F = \frac{\chi(E)}{4\pi} \int_0^\infty dE' \int_{4\pi} d\Omega' \nu(E) \Sigma_f(\vec{r}, E') \tag{2.4}$$

Solve for the flux in Equation 2.2:

$$\Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k}(H^{-1}F)\Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k} M\Psi(\vec{r}, E, \hat{\Omega}) \tag{2.5}$$

Where M is an operator given by $M = H^{-1}F$. This system of equations has infinitely many solutions, with eigenvectors Ψ_i and corresponding eigenvalues k_i – however the most useful solution for reactor applications is the fundamental mode, k_0 and Ψ_0. k_0 is the largest eigenvalue, such that $|k_0| > |k_1| > |k_2| > ...$

Since the source is initially unknown, an iterative calculation is used to converge on a solution when explicitly solving the equations. The n^{th} eigenvector solution in an iterative calculation

is based on the previous iteration's solution, $(n-1)$, given by:

$$\Psi^{(n)} = \frac{1}{k^{(n-1)}} M\Psi^{(n-1)} \tag{2.6}$$

which leads directly to the eigenvalue associated with the n^{th} iteration eigenvector:

$$k^{(n)} = \frac{\langle M\Psi^{(n)} \rangle}{\langle M\Psi^{(n-1)} \rangle} = \frac{\langle M\Psi^{(n)} \rangle}{k^{(n-1)} \langle \Psi^{(n)} \rangle} \tag{2.7}$$

where the bra-ket ($\langle ... \rangle$) symbols indicate integration over all independent variables.

This iteration process, called the power iteration method, is repeated until a pre-defined "tolerance" is reached. The power iteration method is widely used by Monte Carlo codes to determine the fundamental modes in criticality calculations [31, 32].

One parameter for measuring the convergence of the power iteration method is the "dominance ratio" [21, 33], as defined by:

$$DR = \frac{|k_1|}{|k_0|} \leq 1 \tag{2.8}$$

where k_1 is the second eigenvalue solution to Equation 2.5, and k_0 is the first. A high dominance ratio, or a ratio of $|k_1|$ to $|k_0|$ close to 1, leads to convergence issues within a Monte Carlo eigenvalue calculation due to cycle-to-cycle correlations.

2.3.1 Monte Carlo Eigenvalue Method

As previously mentioned, the Monte Carlo method does not explicitly attempt to solve the LBE. Rather, Monte Carlo codes simulate neutrons and track their interactions in a medium, in a statistical process, to determine the k-eigenvalue of a system.

First, an initial neutron source is placed in the fuel region of a model. These neutrons are transported through the medium and their interactions are tracked. If absorbed, the neutron

may induce fission. If fission is induced, the code calculates how many fission neutrons are born from that interaction, and these neutrons are subsequently transported. Neutrons are tracked in "generations" from which they are born.

To calculate the fission source distribution, a code uses Equation 2.9 [21]

$$max \left| \frac{S_i^{(n)} - S_i^{(n-1)}}{S_i^{(n-1)}} \right| < \epsilon \tag{2.9}$$

where n is the generation number, and ϵ is a pre-defined tolerance. This has a corresponding eigenvalue of

$$K^n = \frac{N^{(n)}}{N^{(n-1)}} \tag{2.10}$$

where N is the number of fission neutrons in generation n.

The accuracy of the source distribution and eigenvalue calculated by this procedure is heavily dependent on three Monte Carlo parameters– the number of particle histories tracked per generation (NPS), the number of generations (also called cycles) skipped (NSK), and the number of active generations (NAC). These parameters largely determine whether a Monte Carlo simulation converges towards the correct fission source distribution, and also impact uncertainties in tallies and eigenvalue. An in depth study of the effect these parameters have on eigenvalue calculations is performed in Chapter 4.

2.3.2 Fission Matrix Eigenvalue Method

Recently, there has been a renewed interest in developing methods for solving eigenvalue problems accurately, while dealing with source convergence issues associated with the Monte Carlo power iteration method. One such method, called the Fission Matrix (FM) methodology, has piqued the interest of researchers due to its effective handling of these issues [34]. The FM approach to eigenvalue calculations begins similarly to the approach outlined in Section 2.3 [1, 21, 35]. First, begin with Equation 2.2 once again:

$$H\Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k} F\Psi(\vec{r}, E, \hat{\Omega}) \tag{2.11}$$

Then rewrite the F operator as

$$F = \chi\tilde{F} \tag{2.12}$$

pulling the fission spectrum, χ, out of the operator F to separate the fission neutron density from the fission spectrum. $\tilde{F}$ is therefore given by

$$\tilde{F} = \frac{1}{4\pi} \int_0^\infty dE' \int_{4\pi} d\Omega' \nu(E) \Sigma_f(\vec{r}, E') \tag{2.13}$$

Equation 2.2 then becomes

$$H\Psi(\vec{r}, E, \hat{\Omega}) = \frac{1}{k} \chi\tilde{F}\Psi(\vec{r}, E, \hat{\Omega}) \tag{2.14}$$

Solving for Ψ,

$$\Psi = \frac{1}{k} H^{-1}\chi\tilde{F}\Psi \tag{2.15}$$

Apply $\tilde{F}$ to both sides of this equation to get

$$\tilde{F}\Psi = \frac{1}{k} (\tilde{F}H^{-1}\chi)\tilde{F}\Psi \tag{2.16}$$

Define two new expressions, $S = \tilde{F}\Psi$ and $A = \tilde{F}H^{-1}\chi$ to rewrite the above equation as

$$S(\vec{r}, E, \hat{\Omega}) = \frac{1}{k} AS(\vec{r}, E, \hat{\Omega}) \tag{2.17}$$

Equation 2.17 can be discretized for use on digital computers by splitting a model into N spatial elements, and using the formulation

$$\vec{S} = \frac{1}{k} \boldsymbol{A}\vec{S} \tag{2.18}$$

or

$$S_i = \frac{1}{k} \sum_{j=1}^{N} a_{i,j} S_j \tag{2.19}$$

where

$$a_{i,j} = \frac{\int_{V_i} d^3r \int_{V_j} d^3r' a(\vec{r}' \rightarrow \vec{r}) S(\vec{r}')}{\int_{V_j} d^3r' S(\vec{r}')} \tag{2.20}$$

and $a_{i,j}$ is defined as the number of fission neutrons produced in cell i, due to a fission neutron born in cell j.

This discretized FM method results in a set of linear equations that can be iteratively solved for the fission source distribution, S, and the eigenvalue, k [35]. The general procedure for this iteration is to first start with an initial source guess

$$S_i^{(0)} = \frac{1}{N}, \qquad i = 1, N \tag{2.21}$$

where N is the number of spatial regions. The iteration then goes through steps such that

$$S_i^{(m+1)} = \frac{1}{k^{(m)}} \sum_{j=1}^{N} a_{i,j} S_j^{(m)} \tag{2.22}$$

where

$$k^{(m)} = \sum_{i=1}^{N} S_i^{(m)} \tag{2.23}$$

2.3.3 Comparison of Monte Carlo and Fission Matrix Eigenvalue Methods

The FM methodology provides a number of benefits compared to the Monte Carlo eigenvalue methodology. Monte Carlo eigenvalue calculations are prone to fission source convergence issues in systems with high dominance ratios. Due to cycle-to-cycle correlations, a Monte Carlo simulation could 'converge' to a biased fission source, and uncertainties may be un-

derestimated in the process. RAPID's implementation of the FM method uses a series of fixed source calculations to create FM coefficients, which are then used to solve a set of linear equations to arrive at a fission source distribution and eigenvalue. This method avoids cycle-to-cycle correlations and poor fission source convergence. Additionally, the fixed source FM method avoids undersampling which Monte Carlo calculations suffer from.

Additionally, there is a wide disparity in the computation time necessary to complete these calculations. Most of the calculation required in the FM methodology is performed ahead of time– during the pre-calculation stage. Once a database of coefficients are pre-calculated, FM calculations can be done in real time, even with changes made to the model geometry or material composition. The Monte Carlo method requires an entirely new calculation each time any parameter is changed.

2.4 Traditional Fuel Burnup Methodology

Fuel burnup is the process of extracting energy from a nuclear fuel source, such as a fuel rod. When fission occurs, an atom (such as the primary fuel source in most reactors, ^{235}U) is split to release energy, but also other particles and daughter nuclides. Other processes, such as isotopic decay or absorption, occur alongside fission during reactor operation. These processes can produce other fissile isotopes as well– in effect, fuel is created while it is being used in a nuclear reactor. Alongside calculating the energy released during reactor operation, the goal of a burnup simulation is to calculate the isotopic composition of fuel regions at a given point in time.

The isotopic changes are determined using the Bateman Equations [36], a set of time dependent coupled first-order differential equations. The Bateman equations are given by Equation 2.24:

$$\frac{dN_j}{dt} = \sum_{i \neq j} \lambda_{ij} N_i - \lambda_j N_j, \quad N_j(0) = N_0, \quad j = 1, ..., n \tag{2.24}$$

where N_j is the atom density of isotope j with decay constant λ_j, N_i is the atom density of isotope i, and λ_{ij} is the general transmutation constant which describe neutron-induced reactions, fission and decay from isotope $i \rightarrow j$. This latter constant (λ_{ij}) can be written as

$$\lambda_{ij} = \phi\sigma_{ij} \tag{2.25}$$

where ϕ is the cell averaged flux and σ_{ij} is the one-group flux-weighted average cross-section, which can be expressed by

$$\sigma_{ij} = \frac{\int_V d^3r \int_E dE\phi(\vec{r}, E)\sigma_{ij}(E)}{\int_V d^3r \int_E dE\phi(\vec{r}, E)} \tag{2.26}$$

For fissionable isotopes, Equation 2.25 changes to

$$\lambda_{ij} = \gamma_{ij}\phi\sigma_{f,ij} \tag{2.27}$$

where γ_{ij} is the fission yield constant for j and $\sigma_{f,ij}$ is the one-group fission cross-section. For decay processes, the expression

$$\lambda_{ij} = l_{ij}\lambda_i \tag{2.28}$$

is used, where l_{ij} is the branching ratio for $i \rightarrow j$

The main methods used for solving the Bateman equations are the Linear Chain Method [8] and the Matrix Exponential Method [9]; the latter is the standard method used by most codes, including SERPENT.

A fuel burnup code couples the Bateman equations 2.24 with a transport calculation (described in Section 2.2), performed iteratively for each burnup step. The transport calculation produces cell-average flux distributions, average cross-sections, and other constants used in the depletion calculation. Once the depletion calculation is performed, the updated material composition data are used in the transport equation to obtain the flux distribution; this process repeats until the last burnup step is reached.

During each time step (t), certain values like cross-sections are held constant for the duration of the depletion step (Δt). Currently in SERPENT, the flux and cross-section values used in the depletion calculation are approximated using the formulation

$$\frac{1}{t_{i+1} - t_i} \int_{t_i}^{t_{i+1}} f(t) dt \tag{2.29}$$

where $f(t)$ is a continuous analytic function of the flux or cross-section created using discrete values of previous burnup steps.

These approximations can lead to inaccuracies– though there are methods to combat this, such as the "predictor-corrector" approach which predicts the reaction rate behavior during the depletion step, then applies a correction to the flux distribution at the end of the burnup step. However, the effect of a low resolution time Δt time step scheme can still be observed even when applying a predictor-corrector scheme. For this reason, the effect of time step resolution on isotopic composition is studied in Chapter 5.

Chapter 3

The *b*RAPID Fuel Burnup Methodology

*b*RAPID is an algorithm that is implemented into the RAPID particle transport code in order to perform 3D burnup calculations. This algorithm can be used to calculate full-core pin-wise fission source distributions, k_{eff} , 3D assembly-wise burnup and material distributions, and in this thesis has been extended to produce 2D pin-wise burnup and material composition distributions as described in Chapter 7. In this chapter, RAPID's Multi-stage Response-function Transport (MRT) methodology and the *b*RAPID algorithm itself is described.

3.1 RAPID MRT Methodology

The RAPID code system [1, 2, 10, 35] is a Multi-stage Response-function Transport (MRT) based approach for solving otherwise difficult particle transport problems very quickly. The MRT methodology partitions a transport problem into constituent parts and generates a set of coefficients; e.g., the coefficients for the FM methodology that are used in eigenvalue calculations. During this process, RAPID can determine a full 3D fission source distribution,

the k-eigenvalue (k_{eff}), detector response, and other quantities for spent fuel pools, casks, or reactor environments. Fission source distributions and k_{eff} values are the most important quantities calculated by RAPID for the analyses in this thesis.

Here, the RAPID methodology and its relevance to this thesis is examined. In particular, the focus will be on how to generate workable databases for use in *b*RAPID. Figure 3.1 shows a flowchart of the general procedure of RAPID broken up into these stages.

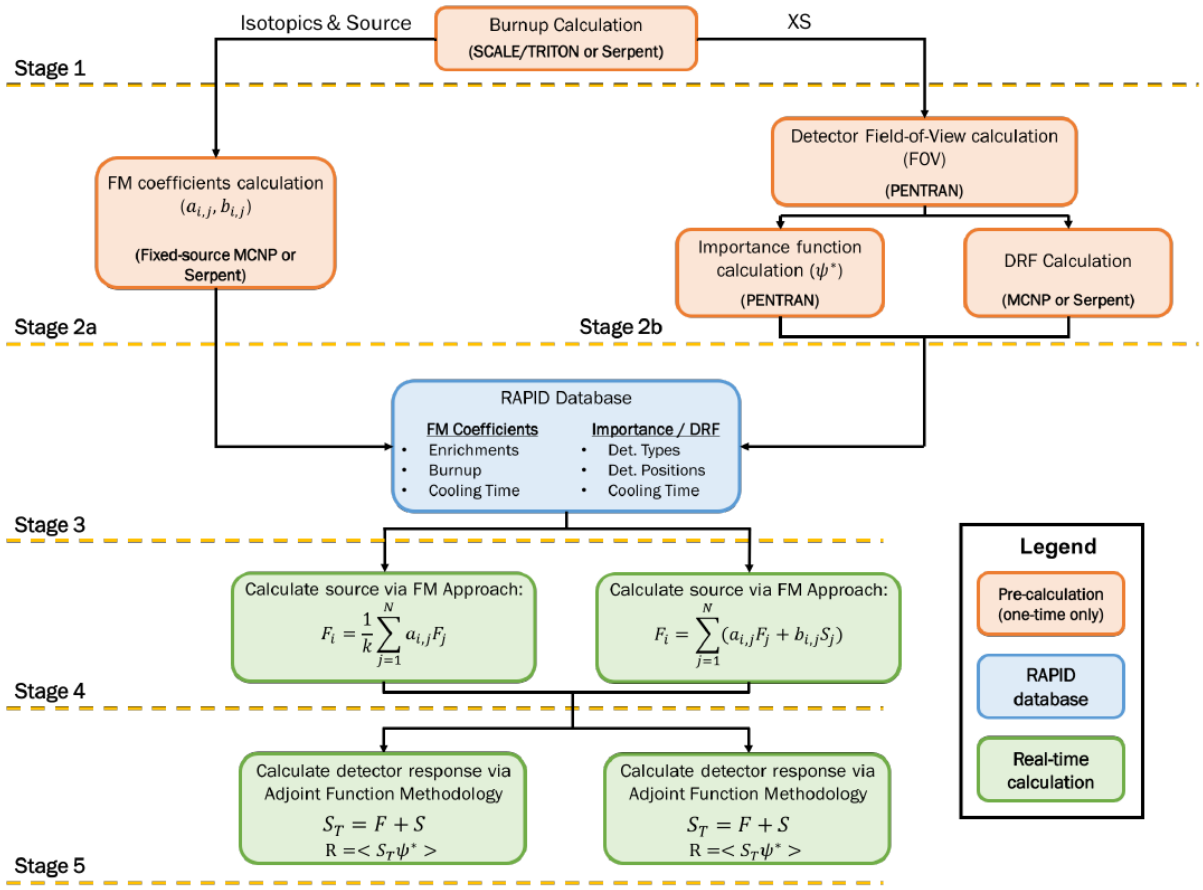


Figure 3.1 RAPID flowchart [1]

3.1.1 Database Development

The first step in using RAPID is to create a database of FM coefficients that are versatile enough to be used in multiple reactor scenarios. These FM coefficients can be generated using the automated pre- and post- processing utility software called pRAPID [1]. When creating a *b*RAPID database, this process is split into two subprocesses: a burnup pre-calculation, and the FM coefficient generation. For *b*RAPID, the burnup pre-calculation is necessary because the FM coefficients contain information not just about initial enrichment and geometry, but also power density and irradiation time. The burnup calculation is used to retrieve material composition and fission spectrum at each burnup step.

A *b*RAPID database is created by specifying:

- Number of power densities and their value (indicated as b_1, b_2, ...) in units of kW/g
- Number of irradiation times and their value (indicated as t_1, t_2, ...) in units of days

The selection of power density values and an appropriate time step scheme is very important during the database development stage of a *b*RAPID pre-calculation. Previous work has shown that the number of power density and time step values has a measurable impact on the accuracy of *b*RAPID results [1]. In Chapter 5, the effect of time step resolution on material composition in Monte Carlo burnup calculations is examined– a significant component of the *b*RAPID pre-calculation process.

3.1.1.1 Burnup Pre-calculation

The burnup pre-calculation is based on a 2D, radially infinite quarter assembly model, which is burned at a constant average power density (say, b_1) for all the irradiation times specified (t_1, t_2, ...), for each power density specified. The burnup distribution within the assembly is captured using octally-symmetric pin specification within the assembly, as shown in Figure

3.2. These pins are tracked for the burnup calculation, then translated to fill the entire assembly to capture the radial burnup behavior. This calculation can be accomplished using existing depletion code systems such as SERPENT [4]. The FM coefficient calculation also

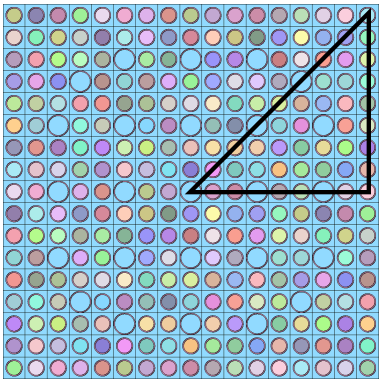


Figure 3.2 Fuel assembly with a single octant highlighted

requires the fission neutron spectrum for the fixed source calculations, for which the Watt fission neutron spectrum given by Equation 3.1 is used [1, 37].

$$\chi(E) = Ce^{-E/a} \sinh \sqrt{bE} \tag{3.1}$$

In order to characterize each fissionable isotope in burned fuel, an average fission neutron spectrum using weighting factors for each isotope is calculated, such that:

$$\bar{\chi}(E) = \sum_i w_i \chi_i(E) \tag{3.2}$$

where i is the index of each isotope, and the weights are calculated using:

$$w_i = \frac{\nu_i \sigma_{f,i} \phi_j}{\sum_i \nu_i \sigma_{f,i} \phi_j} \tag{3.3}$$

where ν_i is the average number of fission neutrons produced for each isotope, $\sigma_{f,i}$ is the fission cross-section for each isotope i, and ϕ_j is the neutron flux in the fuel material (the latter two terms combine to make the expression for reaction rate).

3.1.1.2 FM Coefficient Generation

Recall that FM coefficients ($a_{i,j}$) are defined as the number of fission neutrons produced in cell i, from a fission neutron born in cell j. FM coefficients are generated using a series of fixed-source calculations in the model of interest, for each power density and time step specified. These calculations are performed by placing a fixed neutron source in each fuel segment, e.g., pin (cell j) of one octant of the central assembly– then the resultant fission neutron production for each pin (cell i) in the region of interest (ROI) is tallied. This process is repeated for each power density (b_1,b_2 ...) and time step (t_1,t_2 ...) specified by the user when creating a *b*RAPID database.

Figure 3.3a shows the octant where the the fixed neutron sources are placed. Figure 3.3b shows the region of interest with 1, 2, and 3 relative assembly distances labelled. For this problem, a region of interest comprised of 3 relative assemblies– or a 7x7 assembly area– was chosen.

This process produces a set of FM coefficients:

$$\{a_{i,j} \subseteq \text{FM} \quad | \quad i \in \text{ROI} \quad | \quad j \in 1..39\} \tag{3.4}$$

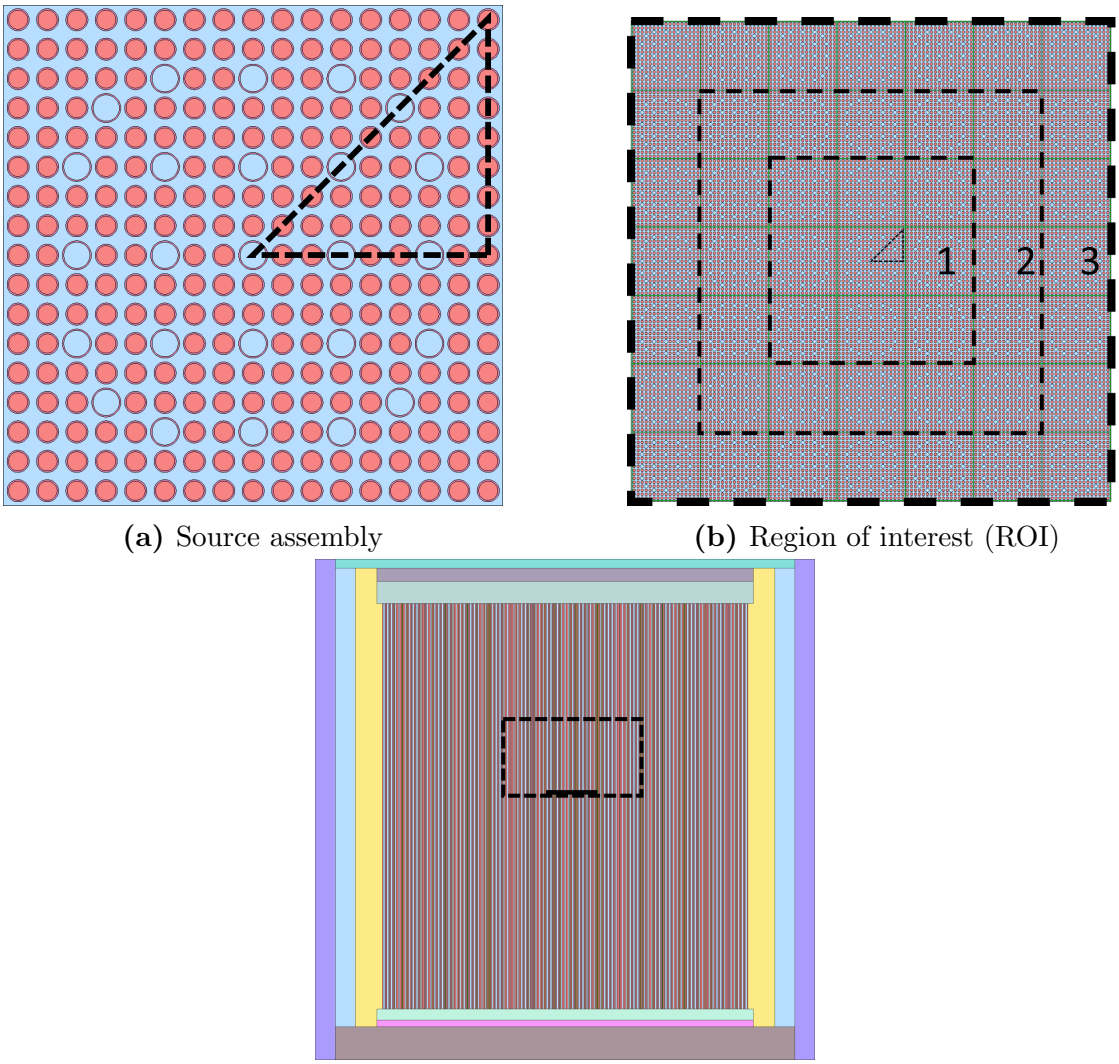
where FM is the full fission matrix, ROI is the region of interest, and 1..39 are the 39 fixed source locations of j in the central assembly.

In order to obtain coefficients for the entire central assembly, octal symmetry is used and coefficients are translated from the first octant to all other octants, illustrated in Figure 3.4 The end result of this process are the coefficients:

$$\{a_{i,j} \subseteq \text{FM} \quad | \quad i \in \text{ROI} \quad | \quad j \in \text{SA}\} \tag{3.5}$$

where SA stands for "source assembly", or the central assembly.

To fill the entire radial portion of the fission matrix, the set of FM coefficients is translated



(a) Source assembly

(b) Region of interest (ROI)

(c) Axial region of interest

Figure 3.3 Source assembly with j octant, region of interest with 1, 2, and 3 relative assembly areas, and axial region of interest for FM coefficient generation outlined in black

by using the geometric similarity of nearby assemblies. In the center of a reactor core, this approximation is valid. However, at the edge of a reactor core or near radial reflectors this approximation becomes problematic, and thus a boundary correction approach (discussed in detail in Chapter 6) is used. Figure 3.5 illustrates the concept of using radial translation to fill FM coefficient elements.

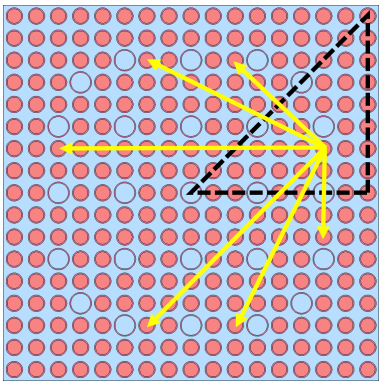


Figure 3.4 Translating FM coefficients to the entire central assembly using octal symmetry

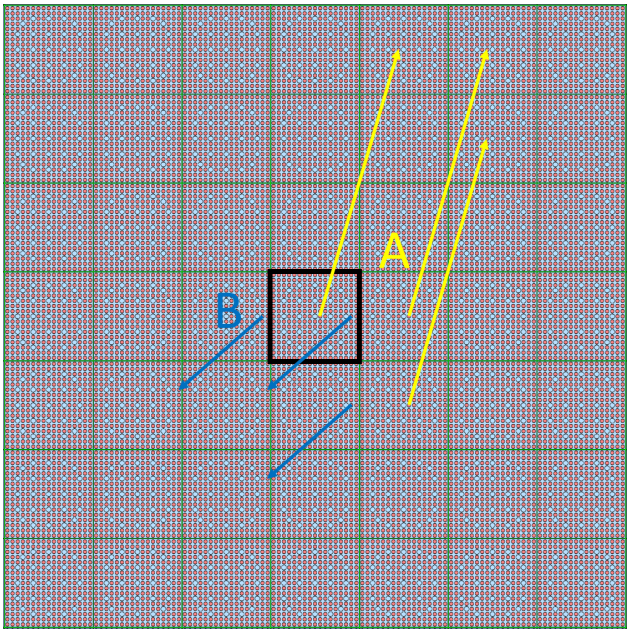


Figure 3.5 Translating FM coefficients radially using geometric similarities. "A" elements are equivalent to each other, and "B" elements are likewise equivalent

The result of this radial translation is a greater portion of the fission matrix being filled, such that:

$$\{a_{i,j} \subseteq \text{FM} \quad | \quad i \in \text{ROI} \quad | \quad j \in \text{RC}\} \tag{3.6}$$

where "RC" is the full radial component of the core.

The last step is to reflect and translate the axial component of the fission matrix. FM coefficients are generated with source locations at the $z = 0$ level and tallied in the $+z$

direction; in order to achieve a full fission matrix, the FM coefficients are reflected using axial symmetry. Figure 3.6 illustrates the process of reflecting $+z$ coefficients to their analogous $-z$ locations.

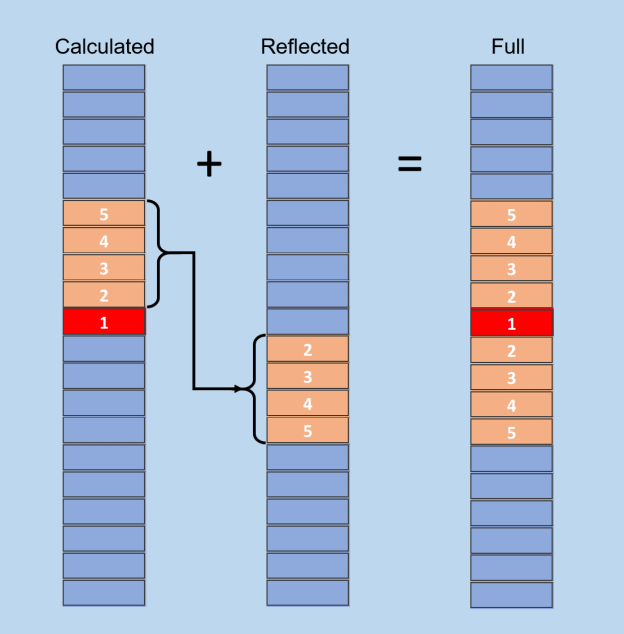


Figure 3.6 Axial reflection, with cell 1 being the source location cell, and cells 2-5 as the ROI

After reflecting the $+z$ coefficients, the coefficients are translated vertically. Figure 3.7 shows this process. The translation process moves FM coefficients in the $+z$ direction (and equivalently in the $-z$ direction) in order to fully fill the axial coefficients.

The net result of all of these reflection and translation processes is the full core fission matrix, which can be used flexibly for reactor applications.

$$\{a_{i,j} \subseteq \text{FM} \quad | \quad i \in \text{ROI} \quad | \quad j \in \text{FC}\} \tag{3.7}$$

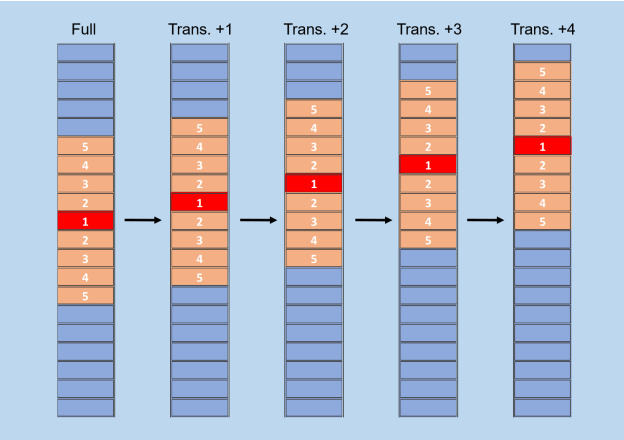


Figure 3.7 Axial translation

3.1.2 Fission Matrix Calculation

Once the database is created, RAPID can be executed to calculate parameters such as the full 3D fission source distribution and k_{eff} . The bulk of the computation time involved in the RAPID process is due to the FM coefficient generation. After generating a database, RAPID solves a linear system of equations in minutes or seconds by using the source iteration scheme presented in Algorithm 1. Recall Equation 2.19:

$$S_i = \frac{1}{k} \sum_{j=1}^{N} a_{i,j} S_j$$

This equation can be solved by using the procedure of Equations 2.21–2.23. The pseudocode of the power iteration algorithm used by RAPID is presented in Algorithm 1 [35]:

Algorithm 1 RAPID power iteration method

Source: $S_i^{(0)}$
Eigenvalue: $k_i^{(0)}$
Source tolerance: ϵ_S
Eigenvalue tolerance: ϵ_k
Max iterations: N_k

while $n \leq N_k$ **do**
$n = n + 1$
$S_i^{(n)} = \frac{1}{k^{(n-1)}} \sum_j a_{i,j} S_j^{(n-1)}$ ▷ Calculate Updated Quantities
$k^{(n)} = \frac{\sum_i S_i^{(n)}}{\sum_i S_i^{(n-1)}}$

$\Delta_{S_i} = \frac{|S_i^{(n)} - S_i^{(n-1)}|}{S_i^{(n-1)}}$ ▷ Calculate Relative Differences
$\Delta_k = \frac{|k^{(n)} - k^{(n-1)}|}{k^{(n-1)}}$

if $\Delta_{S_i} < \epsilon_S$ **and** $\Delta_k < \epsilon_k$ **then** ▷ Convergence Check
return $S_i^{(n)}$, $k^{(n)}$
exit()
end if
end while

3.1.2.1 Material to Material Correction Factor

RAPID performs calculations based on assembly-wise quantities, specified by the user. From a given input, RAPID will interpolate coefficient values based on the pre-calculated database to obtain the correct coefficients for the FM calculation. The fixed source calculations used to create the coefficients, however, are performed assuming uniform pin-wise material compositions in the region of interest. In other words, "source" j cells and "destination" i cells have the same material composition. This assumption works for sysems with uniform distributions, but oftentimes a reactor will have a wide variety of material compositions in neighboring assemblies.

To account for this effect, RAPID uses a "material-to-material correction factor", which modifies coefficients such that

$$a_{i,j} = \frac{c_i}{c_j}\hat{a}_{i,j} \qquad for \quad i \in \text{DA} \tag{3.8}$$

where $\hat{a}_{i,j}$ is the unmodified FM coefficient, c_i and c_j are coefficients describing the material composition of cells i and j respectively, and DA is the destination assembly. The c correction factors are calculated as a sum of all coefficients for that material. For example, a database entry at p_1, t_1 would have a coefficient factor of

$$c_l(p_1, t_1) = \sum_i a_l(p_1, t_1) \tag{3.9}$$

where l is either the i or j index.

Therefore, a ratio of c_i to c_j is an approximate measure of difference made by generating $a_{i,j}$ coefficients in an environment with the uniform material composition of cell i versus the material composition of cell j. Multiplying the FM coefficient by this ratio corrects for the material differences of cells i and j in the eigenvalue calculation.

3.1.2.2 Boundary Correction

In addition to material-to-material correction factors, RAPID utilizes boundary correction factors to account for the fact that all assemblies are not at core center. While this does not generally cause issues in large reactor calculations, in smaller systems or systems with large reflector boundaries this approximation can lead to inaccuracy in the boundary region.

To correct for this, a boundary correction factor $bnd(x, y)$ was developed, where a uniform fixed source calculation is performed on the model of interest and an infinite model. The

fission neutron production rate is tallied for these two scenarios and divided such that

$$bnd(x,y) = \frac{f_{ref}(x,y)}{f_{inf}(x,y)} \tag{3.10}$$

This boundary correction factor is then applied to the FM coefficients with the expression

$$a_{i,j} = bnd(x_i, y_i)\tilde{a}_{i,j} \tag{3.11}$$

where $\tilde{a}_{i,j}$ is the uncorrected FM coefficient. It is important to note that x and y (as indicated by the i index) are discrete pin locations in the reactor, not continuous variables.

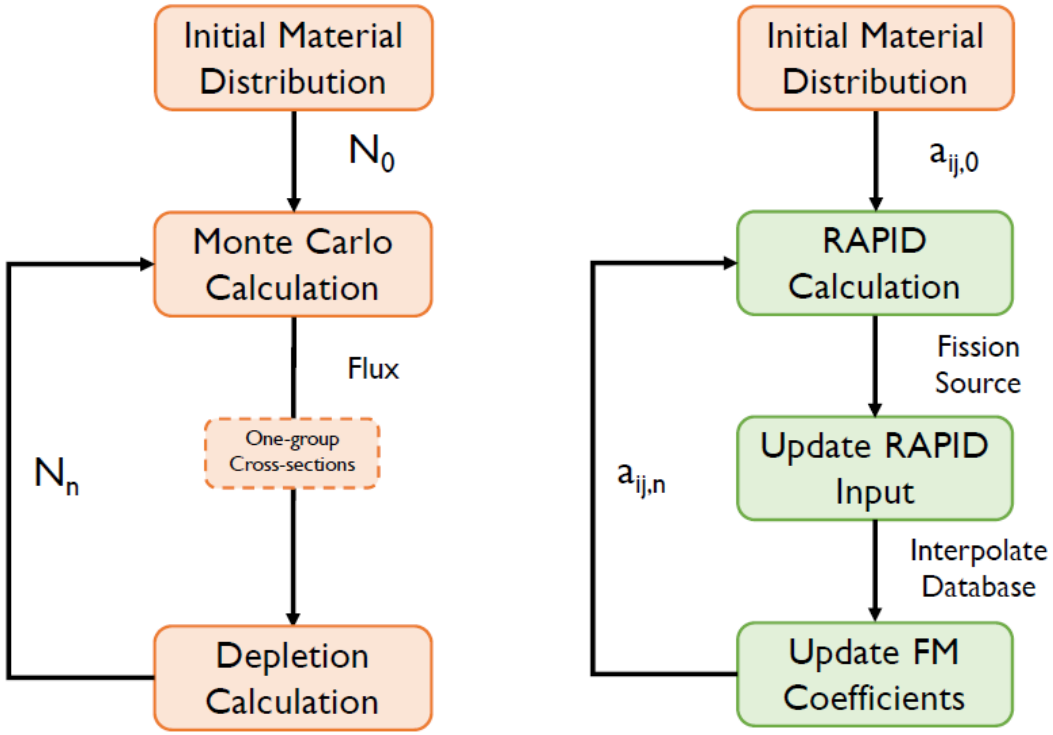
More detail into this process can be found in Chapter 6.

3.2 *b*RAPID Burnup Methodology

The *b*RAPID methodology uses the existing RAPID structure and automates it in a manner analogous to the traditional burnup methodology. Traditional burnup methodologies, as described in Chapter 2, couple either a deterministic or Monte Carlo eigenvalue calculation with a depletion calculation in order to continuously update the material composition of the model. By comparison, *b*RAPID uses a RAPID fission matrix eigenvalue calculation to continuously update the power density distribution of the model, which is fed back into RAPID.

Figure 3.8 [1] compares these two approaches side by side. One can see that both of these processes begin with an initial material distribution. In a traditional Monte Carlo approach, this would consist of a reactor geometry with cells filled with a particular material composition or initial enrichment, signified by N_0. This initial material composition would be used in the eigenvalue calculation to produce a flux distribution. *b*RAPID uses the fission matrix approach, which utilizes an initial set of FM coefficient $a_{ij,0}$ which contain geometric,

material, and flux distribution information inherently. These FM coefficients are used to produce a fission source distribution.



(a) Traditional **(b)** *b*RAPID

Figure 3.8 Simplified flowcharts of the traditional and *b*RAPID burnup methodologies [1]

The traditional approach then takes the flux distribution and calculates one-group cross-sections which are used to solve the Bateman equations explicitly. This produces a new material composition that can be fed back into the Monte Carlo eigenvalue calculation, to repeat the process until the last burnup step is reached. *b*RAPID uses the fission source to produce a new power density distribution, which then tells RAPID to determine the FM coefficients $a_{ij,n}$ as a function of power density and exposure time for the next burnup step. Again this process is repeated until the last burnup step of interest.

While these two methodologies are similar, the fission matrix approach of *b*RAPID provides significant advantages over the standard Monte Carlo/depletion coupling scheme. Firstly, the fission matrix approach uses pre-calculated coefficients to generate a set of linear equa-

tions that can be solved using the power iteration scheme described in Section 3.1.2. This effectively off-loads the vast majority of the calculation time into database creation, which makes the actual *b*RAPID calculation time very fast. This database pre-calculation also adds a level of flexibility– the *b*RAPID algorithm can be run with a different initial burnup distribution without having to re-run the bulk of the calculation. In contrast, the Monte Carlo approach is orders of magnitude slower due to the transport calculation having to be re-run if any change is made to the system.

Monte Carlo eigenvalue calculations are known to have fission source convergence issues for problems with a high dominance ratio, or DR, and under-sampling, as described in Section 2.3.1. In addition, cycle-to-cycle correlation is a considerable issue in Monte Carlo eigenvalue calculations, as they have been shown to cause under-prediction of uncertainty values on tallies [11]. The FM approach of RAPID solves both of these issues by using a set of coefficients that are derived from fixed source calculations, not a MC eigenvalue calculation, hence bypassing the cycle-to-cycle correlation issue.

While *b*RAPID avoids solving the Bateman equations explicitly during execution, it does rely on a Monte Carlo burnup pre-calculation to retrieve material composition values, and thus this portion of the calculation may be susceptible to issues caused by the Monte Carlo method. However, this pre-calculation is performed on a 2D single assembly model with reflected boundary condition, which avoids the issues associated with MC eigenvalue calculations in large systems and undersampling.

3.3 *b*RAPID Algorithm

At each burnup step (n) of a *b*RAPID run, RAPID is executed to solve the fission matrix equations, expressed by

$$F_i^n = \frac{1}{k} \sum_j a_{i,j}^b F_j^n \tag{3.12}$$

where i and j indicate cell locations, k is the eigenvalue, F_i^n is the fission source in cell i for step n, and $a_{i,j}^b$ is the fission matrix coefficient representing neutrons produced in cell i induced by a neutron born in j.

The $a_{i,j}^b$ coefficients seen here are distinct from the standard RAPID FM coefficients, $a_{i,j}$– they are produced as a function of irradiation time (it) and power density (ip). These coefficients are interpolated values, produced via linear interpolation between database elements. Equation 3.13 shows this formulation for a given power density p and irradiation time t

$$a_{i,j}^b = \frac{(p - p_{ip-1})a_{i,j}(ip,t) + (p_{ip} - p)a_{i,j}(ip-1,t)}{p_{ip} - p_{ip-1}} \tag{3.13}$$

Equation 3.12 is used to obtain a normalized assembly-wise fission source distribution f^n, such that

$$f_i^n = \frac{F_i^n}{\sum_i F_i^n} \tag{3.14}$$

which is then used to calculate the power density distribution for the next step RAPID calculation, p^{n+1}, in the formulation

$$p_i^{n+1} = (\bar{P}^{n+1} N_b) f_i^n \tag{3.15}$$

where N_b is the number of burnable regions in the reactor.

This power distribution is normalized to the user-defined system average power density, or

$$\frac{1}{N_b} \sum_i p_i^n = \bar{P}^n \tag{3.16}$$

The irradiation time for each assembly is given by

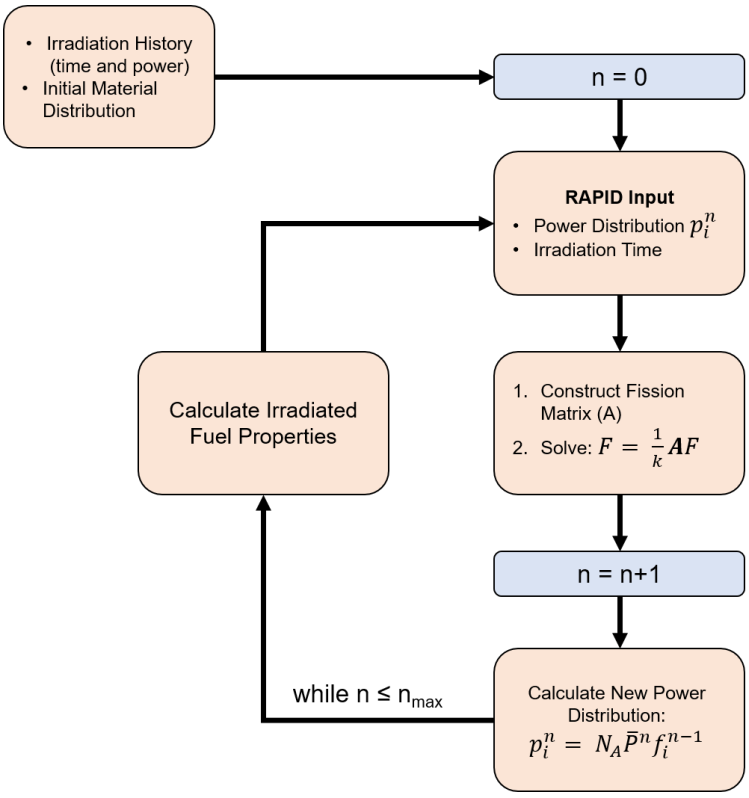
$$t_i^{n+1} = t_i^n + (T^{n+1} - T^n) \tag{3.17}$$

where t_i^n is the current time step, and $(T^n - T^{n-1})$ is the time interval the user specifies for

step n.

Each of the values used in the *b*RAPID calculation (p_i^n, f_i^n, t_i^n, etc.) are *assembly-wise* quantities. This is due to the fact that currently, RAPID is built to perform calculations on assembly-wise input specifications. This limitation is overcome somewhat by the methodology introduced in this thesis (Chapter 7), which describes a 2D pin-wise burnup algorithm. However, in order to perform 'true' pin-wise RAPID burnup calculations, major changes would be made to the RAPID code system to allow it to specify pin-wise axially dependent FM coefficients for calculation, which is beyond the scope of this work.

Figure 3.9 illustrates the *b*RAPID algorithm described here.



Name	Variable
Fission source	F
Normalized, assembly-wise fission source distribution	f_i^{n-1}
Eigenvalue	k
Step number	n
Average power density	$\bar{P}^n$
Assembly-wise power density distribution	p_i^n
Number of fuel assemblies	N_A

Figure 3.9 Generalized *b*RAPID flowchart

3.3.1 Material Composition Interpolation

*b*RAPID uses RAPID to generate a power density distribution at each step that can be used to calculate burnup, material composition, and the RAPID input for the next burnup step. However, *b*RAPID database entries are generated as a function of discrete variables– power densities $(p_1, p_2, ...)$ and irradiation times $(t_1, t_2, ...)$. If the calculated specific power density or irradiation time for a given assembly is not equal to that of a database entry, the *b*RAPID algorithm must use interpolation to retrieve the correct material composition values.

The algorithm utilizes a two-factor linear interpolation scheme to accomplish this. Figure 3.10 illustrates this interpolation process.

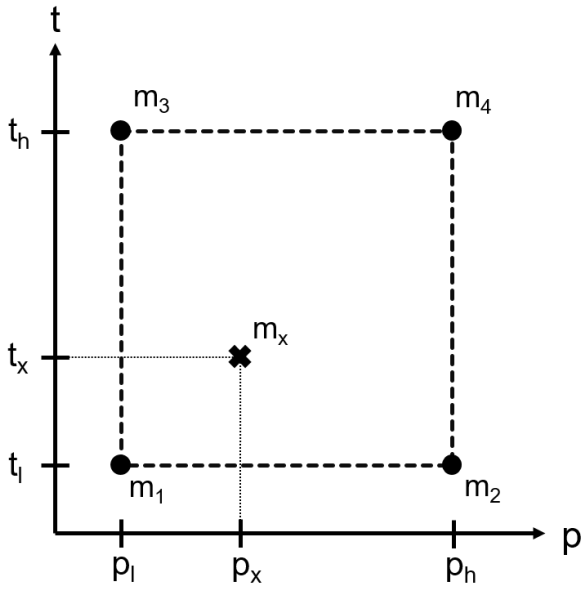


Figure 3.10 Two-factor linear interpolation scheme. The axes are labeled as p for power density and t for irradiation time, where l and h are the low and high p and t values for database entries. The "X" in the center marks the value provided to interpolate between these database entries.

The "x" values are the power density and irradiation time values that come out of a *b*RAPID calculation, which the algorithm uses to identify the corresponding material composition. The process begins by identifying the power density uppper and lower limit such that

$p_l < p_x < p_h$, and similarly for irradiation time ($t_l < t_x < t_h$). The material composition associated with "x" is then expressed as a summation of weights w_i and the 4 material compositions associated with the 4 database values, m_i, or

$$m_x = \sum_i w_i m_i \qquad where \quad i = 1, ..4 \tag{3.18}$$

These weights are normalized, such that

$$\sum_i w_i = 1 \tag{3.19}$$

Each weight is calculated based on the "distance" they are to the interpolation point, x. The following Equations 3.20 – 3.23 describe each of the four weights used in the calculation.

$$w_1 = \left(\frac{p_h - p_x}{p_h - p_l}\right)\left(\frac{t_h - t_x}{t_h - t_l}\right) \tag{3.20}$$

$$w_2 = \left(\frac{p_x - p_l}{p_h - p_l}\right)\left(\frac{t_h - t_x}{t_h - t_l}\right) \tag{3.21}$$

$$w_3 = \left(\frac{p_h - p_x}{p_h - p_l}\right)\left(\frac{t_x - t_l}{t_h - t_l}\right) \tag{3.22}$$

$$w_4 = \left(\frac{p_x - p_l}{p_h - p_l}\right)\left(\frac{t_x - t_l}{t_h - t_l}\right) \tag{3.23}$$

This linear interpolation technique is a decent approximation for the material composition behavior between database entries, particularly if an optimal time step scheme is used during the database generation. However, future work will look at non-linear interpolation schemes for this step to better approximate material distributions in *b*RAPID calculations.

3.4 Using *b*RAPID

Thus far, the general algorithm of *b*RAPID has been discussed, but its operation has not. This section will briefly discuss the procedure for generating a *b*RAPID database using pRAPID [38]– the automated pre- and post-processing algorithm for RAPID database generation. An overview of the execution of *b*RAPID and its outputs will follow in the latter half of the section.

3.4.1 Creating a *b*RAPID Database Using pRAPID

Section 3.1.1 describes the process of *b*RAPID database development, where FM coefficients are generated as a function of power density p and irradiation time t. This is distinct from a regular RAPID database, which is a function of the initial enrichment of the model. Generating a *b*RAPID database requires more steps than a RAPID database. These steps are described in detail in Appendix A.1

3.4.2 *b*RAPID Execution

The process for executing *b*RAPID is described in Appendix A.2. One of the major changes to the input format of *b*RAPID made in this thesis is the addition of a time dependent boundary correction specification line. The time dependent boundary correction option (described in Chapter 6) adds the following line to `burn.inp`:

```
t_bnd = [ ]
```

The execution of *b*RAPID produces a number of output files, most of which relating to the fission source distribution. At each time step, files containing assembly-wise, pin-wise, 2D and 3D fission source information is produced by RAPID. These are named `<name>_bN_X.inp`,

where "N" indicates the burnup step, and "X" indicates the specific information type. A summary of the k_{eff} convergence for each step is also provided.

At the end of a *b*RAPID execution, multiple log files (`bRAPID.log`, `rapid.burn`, etc.) are created, describing each step of the *b*RAPID and RAPID run. In addition, assembly-wise burnup and material composition data can be accessed with the `mats.out`. Recently, an option to output pin-wise burnup and material composition data was added, which produces the file `mats_pinwise.out`. This pin-wise burnup process is described in detail in Chapter 7.

Chapter 4

Monte Carlo Model Description and Sensitivity Study

In order to perform the tests laid out in this thesis and to develop new algorithms for use in RAPID, it is necesssary to create a well tested reference model using a separate code system. For this, the Serpent particle transport code system [4] for both eigenvalue and depletion calculations has been used. One requirement when creating a model for Monte Carlo eigenvalue calculations is to perform tests to determine the proper values to use for eigenvalue parameters. These tests, called sensitivity studies, are used to gauge the accuracy of the calculation and improve performance by minimizing unnecessary calculation time. Section 4.1 discusses the Serpent Monte Carlo model, and Section 4.2 discusses the tests performed to determine the three primary parameters (NSK, NAC, and NPS).

4.1 Model Description

A reference model based on the NEA/OECD pressurized water reactor (PWR) Monte Carlo performance benchmark model [39] is developed. This model is a 5x5 'mini-core' configura-

tion shown in Figure 4.1. The 'mini-core' is surrounded by water, and a steel pressure vessel. Figure 4.2 shows the 21 assemblies in the core, and Figure 4.3 shows a close-up of a single assembly. For greater detail into the model itself, refer to Reference [39]. The reference model has the following specifications:

- **Model Type** - 2D PWR model arranged in 5x5 mini-core, corner assemblies missing
- **Assemblies** - 21 total assemblies with assembly pitch of 21.42 cm and 25 guide tubes filled with water per assembly
- **Fuel** - 264 fuel pins arranged in a 17x17 array, UO_2 fuel, 4.19 wt% enrichment
- **Power Density** - 25 kW/kg

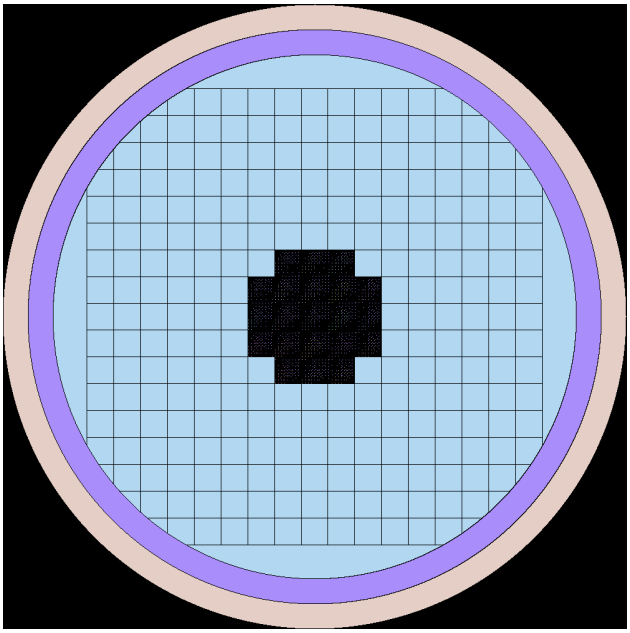


Figure 4.1 Full 5x5 mini-core with pressure vessel

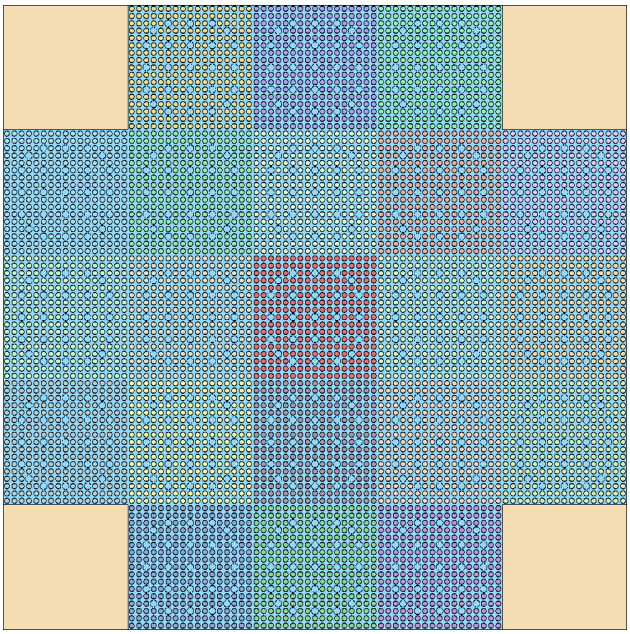


Figure 4.2 21 fuel assemblies in core

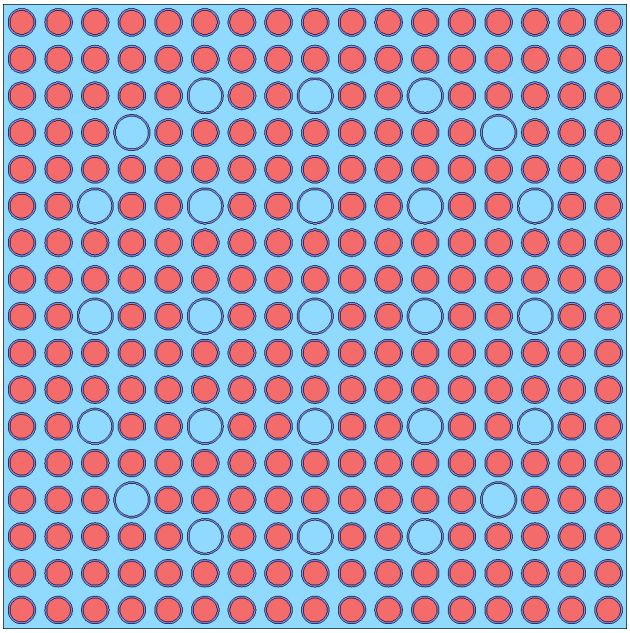


Figure 4.3 Single assembly with guide tubes

4.2 Sensitivity Study

There are three eigenvalue parameters; the number of particle histories per cycle (NPS), number of active cycles (NAC), and number of skip cycles (NSK). All three parameters have

impact on the aforementioned issues associated with eigenvalue Monte Carlo calculations.

To determine the appropriate Monte Carlo parameter values, one must vary one individual parameter while holding the other two constant; then, once settling on a value for that parameter, the other parameters are examined. Beginning with NPS— pick an initial value deemed appropriate given a priori knowledge of other systems. For the model tested (a 2D model with 5,544 fuel regions) a reasonable NPS of 100,000 to give us 18 neutrons per rod will be chosen. Next, an appropriately large NAC is chosen to start out with; 500 active cycles. With these two parameters chosen, the NSK parameter can be tested.

A metric for determining a sufficient number of skip cycles is the Shannon entropy convergence. Briefly, the Shannon entropy is a quantity derived in information theory that measures the convergence of a fission source distribution [21]. In general, the goal is to skip the cycles where Shannon entropy is not converged when choosing an NSK value. To determine NAC and NPS, the k_{eff} uncertainty over time and fission source distribution will be examined. Shannon entropy will also be used as a metric for determining an appropriate NPS. For each test, it is important to keep in mind the run time of the simulation in order to determine whether the additional time spent calculating with larger parameter values is necessary.

Due to the fact that future tests will involve fuel burnup over time, the following convergence trials were performed with an initial burnt fuel distribution, at a core average fuel burnup of 2.28 GWd/tHM. The fuel was then depleted at a core average power density of 0.025 kW/kg for an additional 547 days using 6 burnup steps. Here, the Shannon entropy convergence cycles and the fission source distribution are shown for the 0^{th} step, or the initial burnt fuel distribution only. The k_{eff} uncertainty values are shown for all burnup steps.

4.2.1 NSK

Six trials with varying NSK, described in Table 4.1 were performed to select an adequate number of skip cycles. Figure 4.4 shows the Shannon entropy converging over time for each trial. One can see that the entropy of the system converges quickly, well within 200 cycles. Values such as k_{eff} , k_{eff} uncertainty and fission source distribution do not show significant differences between trials. The run time for the models are relatively similar despite the large changes in NSK. In order to ensure a converged fission source distribution, an NSK value of 200 cycles will be chosen.

Table 4.1 NSK trial values

	NSK	**NPS**	**NAC**	**Run Time***
Trial 1	5	10^5	500	3.91
Trial 2	20	10^5	500	4.56
Trial 3	50	10^5	500	4.75
Trial 4	100	10^5	500	4.38
Trial 5	200	10^5	500	5.80
Trial 6	400	10^5	500	6.16

* Run Time has units of wall-clock hours, performed on 8 processors

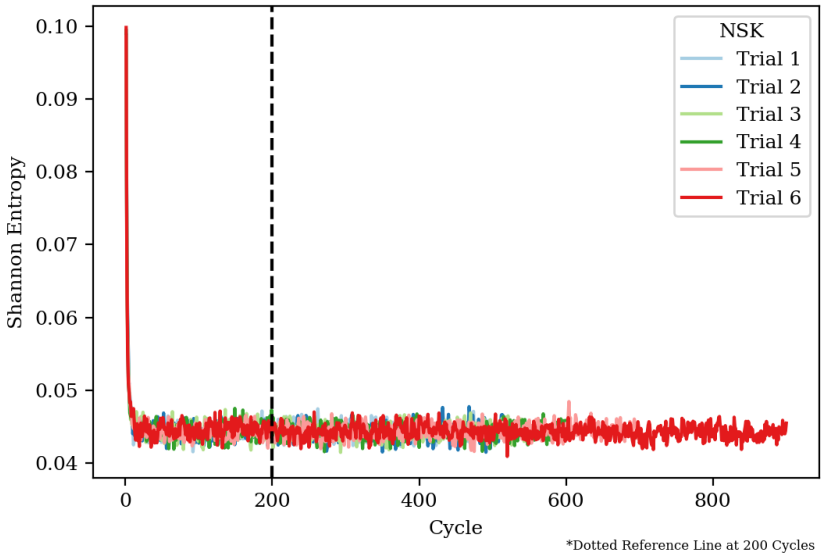


Figure 4.4 Shannon entropy of each trial

4.2.2 NAC

The goal of NAC selection is to select a number of active cycles long enough to allow your system to converge, but not so long as to waste computation time. To that end, it is useful to look at a number of different parameters, including k_{eff} uncertainty and fission source distribution. Table 4.2 describes the different values tested for NAC, holding NSK and NPS constant. Figure 4.5 shows the k-effective uncertainty for each trial. Figure 4.6 shows the relative difference of the fission neutron distribution in each pin between Trials 1 through 8, and Trial 9. Trial 9 was chosen as a reference, as it has the largest number of active cycles. Equation 4.1 describes the percent relative difference calculation performed to achieve this distribution.

$$\%\ Relative\ Difference_{ij} = 100 \times \frac{Trial\ N_{ij} - Trial\ 9_{ij}}{Trial\ 9_{ij}}, \tag{4.1}$$

Where $Trial\ N$ is the fission source of Trial N, $Trial\ 9$ is the fission source of Trial 9, and i,j signifies the position of the pin where the calculation is being done. By examining these

two figures, one can see that Trial 7 yields an average k_{eff} uncertainty of 10 pcm, and a fission source distribution that agrees well with the most accurate case. Thus, an NAC of 500 is selected for this study.

Table 4.2 NAC trial values

	NSK	**NPS**	**NAC**	**Run Time***
Trial 1	200	10^5	5	1.26
Trial 2	200	10^5	20	1.38
Trial 3	200	10^5	50	1.93
Trial 4	200	10^5	100	2.36
Trial 5	200	10^5	200	2.71
Trial 6	200	10^5	400	4.95
Trial 7	200	10^5	500	5.85
Trial 8	200	10^5	1000	8.76
Trial 9	200	10^5	2000	15.7

* Run Time has units of wall-clock hours, performed on 8 processors

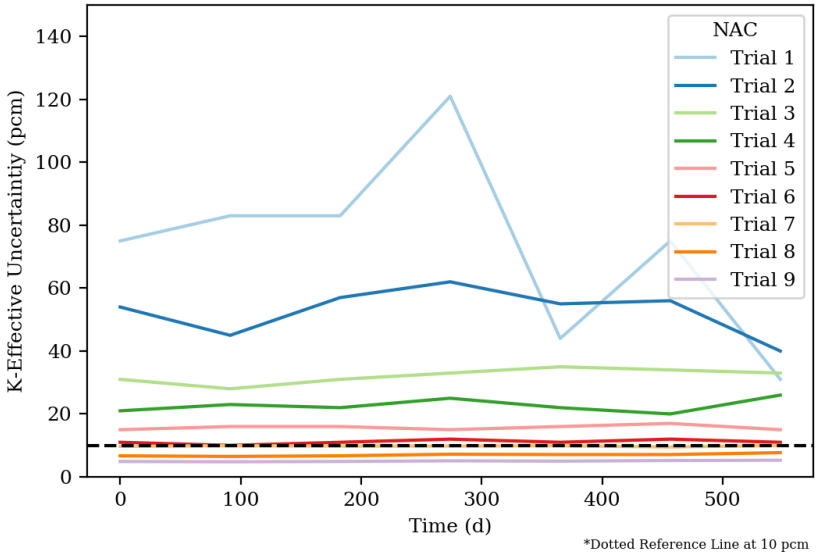
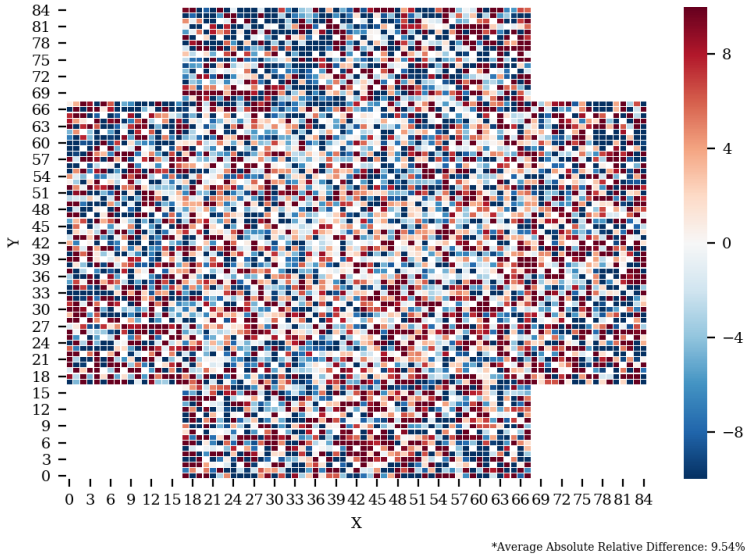
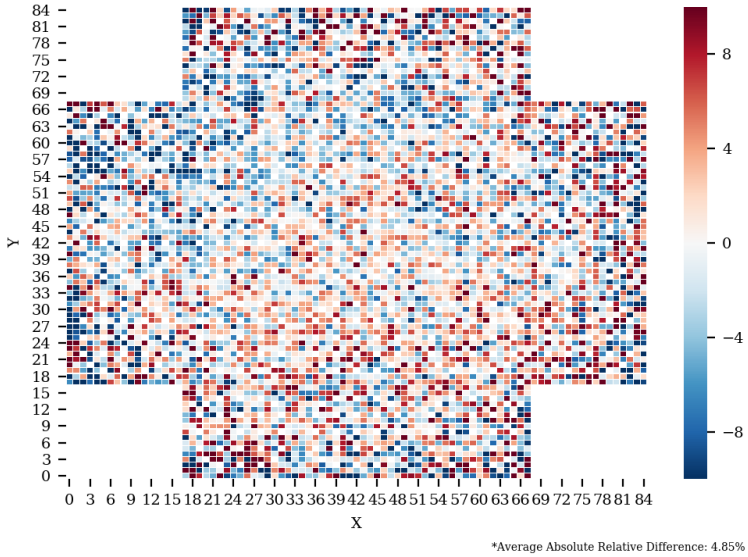


Figure 4.5 k_{eff} uncertainty of each trial over time

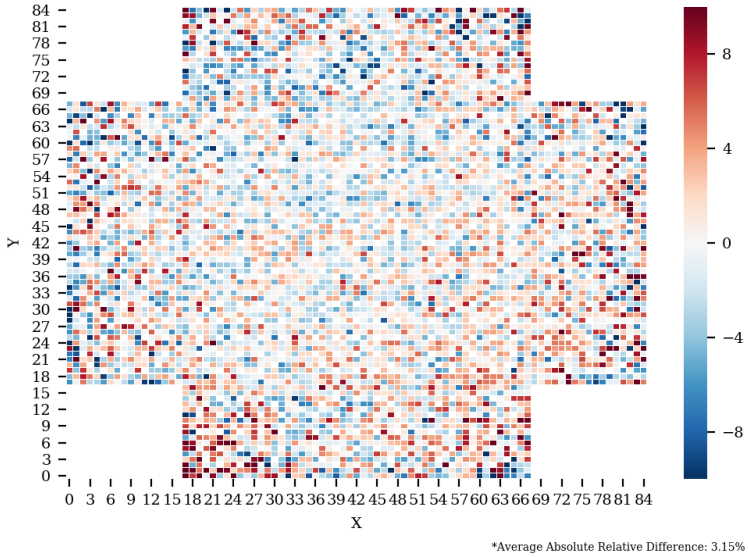


(a) Trial 1 vs. Trial 9

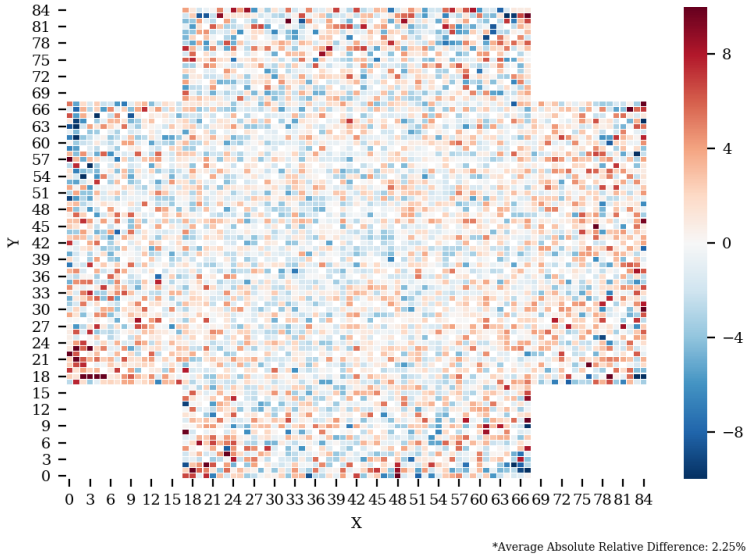


(b) Trial 2 vs. Trial 9

Figure 4.6 Relative difference of fission neutron distribution between Trials 1 - 8 and Trial 9

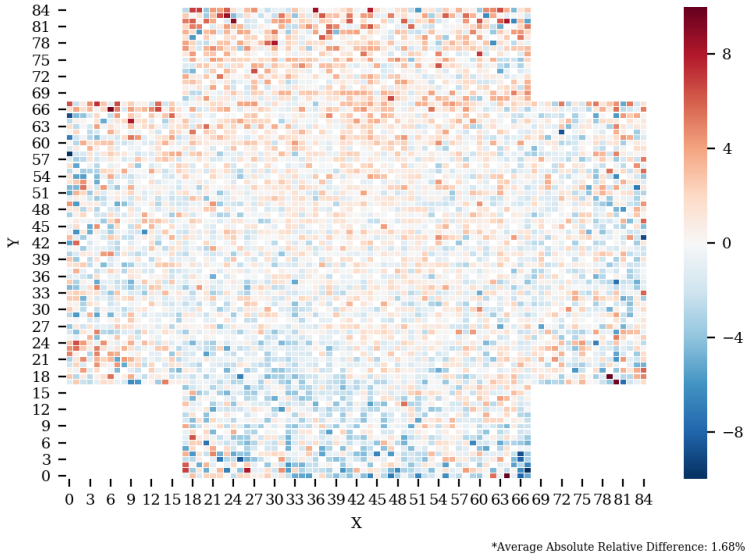


(c) Trial 3 vs. Trial 9

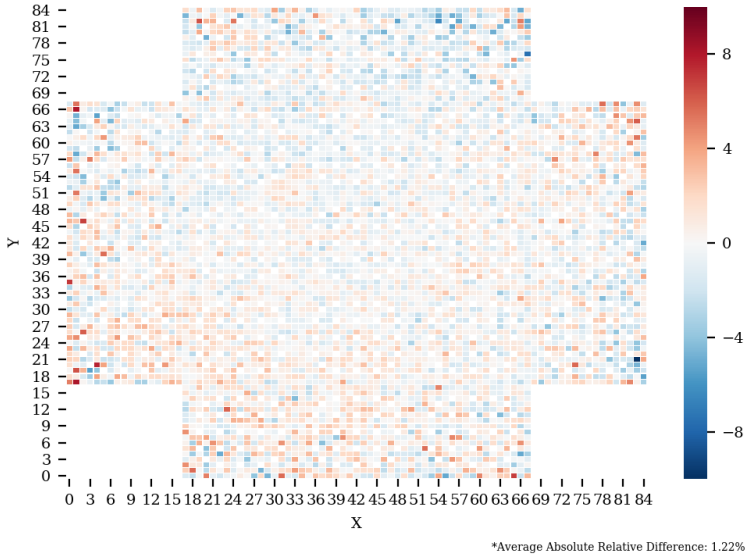


(d) Trial 4 vs. Trial 9

Figure 4.6 Relative difference of fission neutron distribution between Trials 1 - 8 and Trial 9 (cont.)

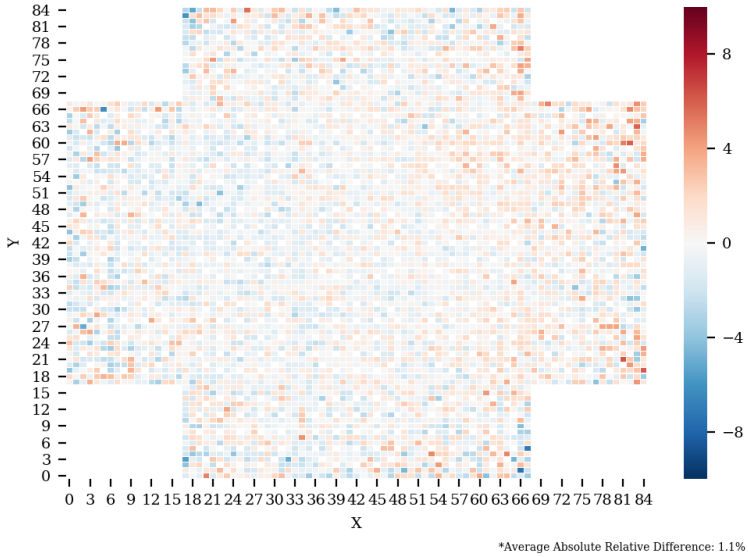


(e) Trial 5 vs. Trial 9

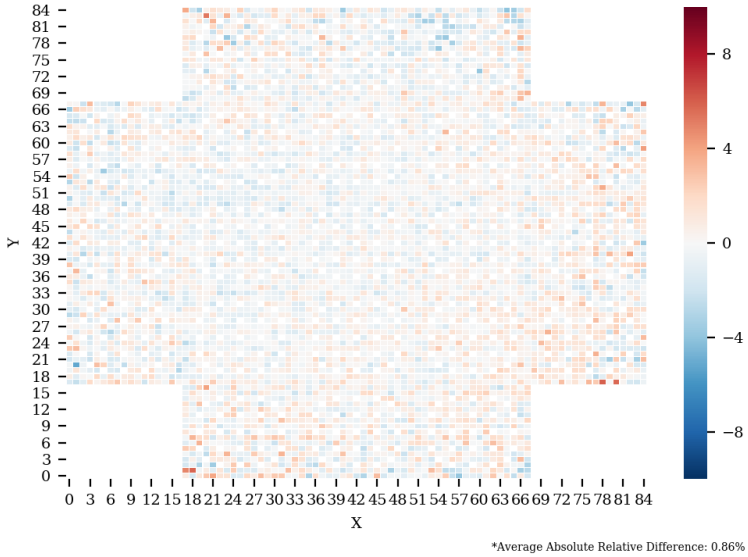


(f) Trial 6 vs. Trial 9

Figure 4.6 Relative difference of fission neutron distribution between Trials 1 - 8 and Trial 9 (cont.)



(g) Trial 7 vs. Trial 9



(h) Trial 8 vs. Trial 9

Figure 4.6 Relative difference of fission neutron distribution between Trials 1 - 8 and Trial 9 (cont.)

4.2.3 NPS

NPS must be tested by looking at Shannon entropy, k_{eff} uncertainty, and fission source distribution convergence. Table 4.3 describes the different values tested for NPS, holding NSK and NAC constant. Figure 4.7 shows the Shannon entropy for each test case over all cycles. Figure 4.8 shows the k_{eff} uncertainty for each trial over time. Figure 4.9 shows the relative difference of the fission neutron distribution in each pin between Trials 1 through 5, and Trial 6, using the same formulation as Equation 4.1.

One can see that the variance of the entropy is inversely related to the number of particles used, and by Trial 3 this statistical variation is quite low. Examining the k_{eff} uncertainty reveals that Trials 4 and 5 achieve an uncertainty of less than 10 pcm. In terms of fission source distribution convergence, Trials 4 and 5 appear suitably converged. Given this data, the Trial 4 values for NPS, NSK, and NAC will be chosen.

Table 4.3 NPS trial values

	NSK	**NPS**	**NAC**	**Run Time***	**No. Particles Per Region**
Trial 1	200	10^3	500	0.13	0.18
Trial 2	200	$5 \cdot 10^3$	500	0.35	0.90
Trial 3	200	$5 \cdot 10^4$	500	2.93	9.01
Trial 4	200	10^5	500	5.80	18.03
Trial 5	200	$5 \cdot 10^5$	500	24.85	90.18
**Trial 6	200	10^6	500	96.20	180.37

* Run Time has units of wall-clock hours, performed on 8 processors
**Trial 6 is only used for fission source distribution relative difference calculation

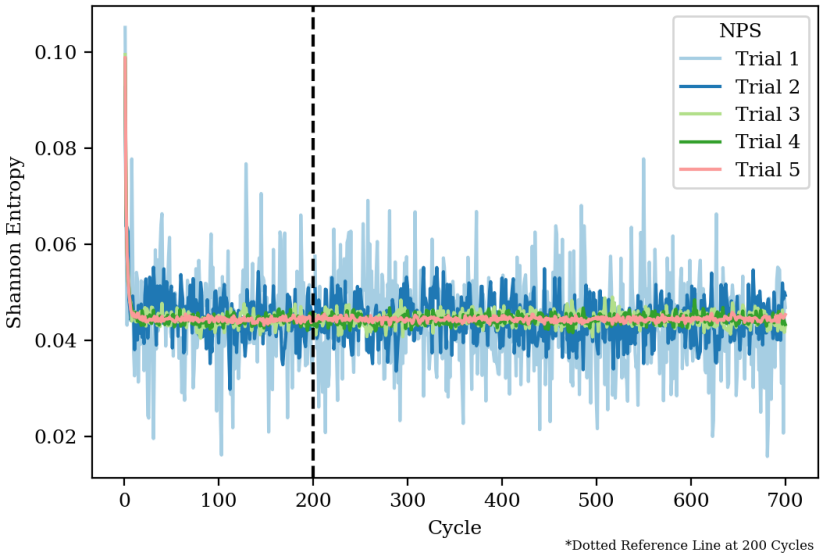


Figure 4.7 Shannon entropy convergence for each trial

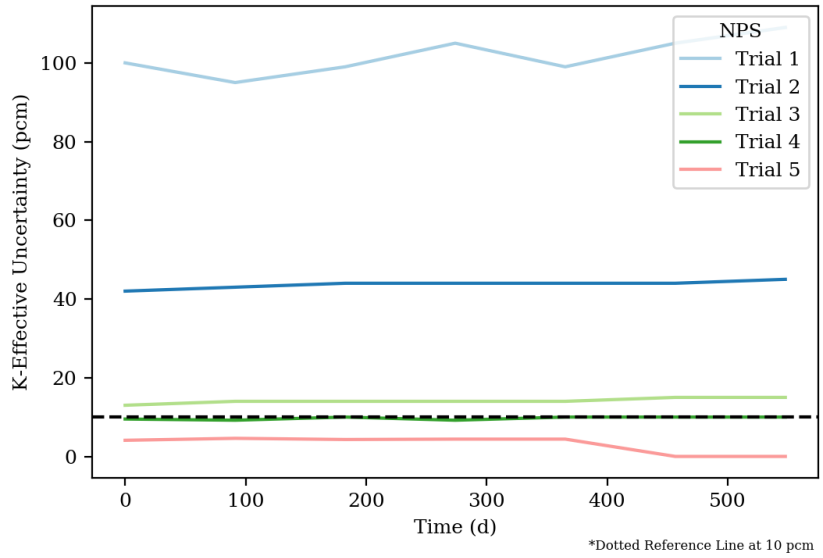
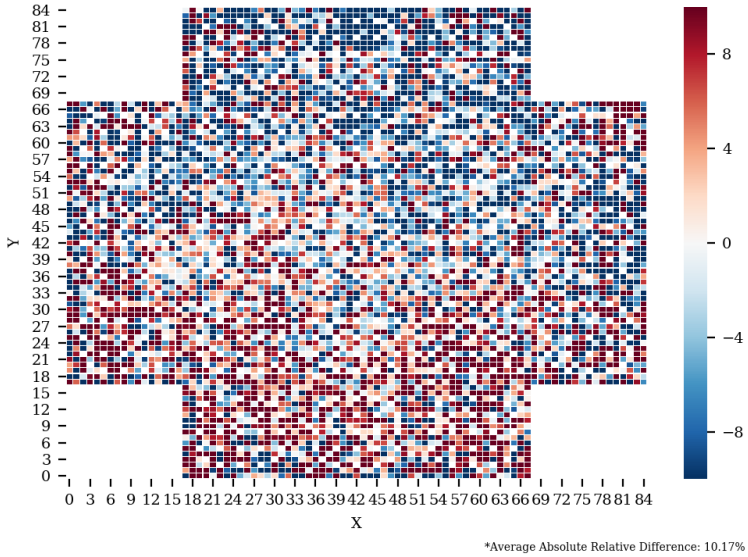
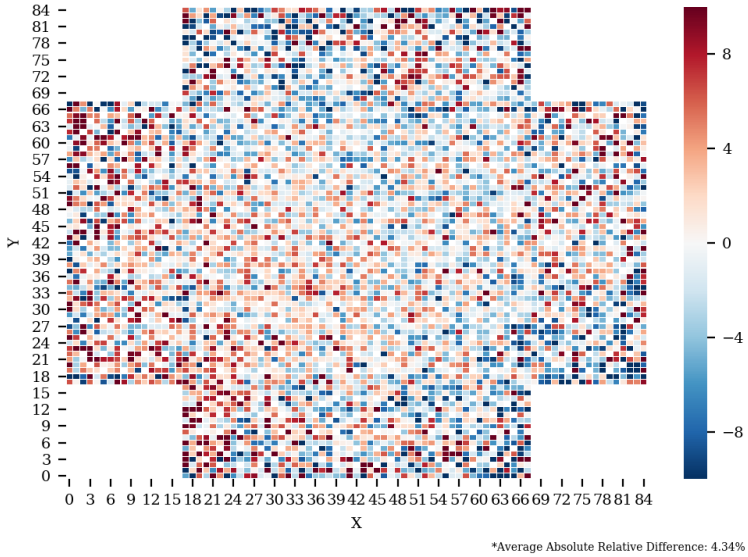


Figure 4.8 k_{eff} uncertainty of each trial over time

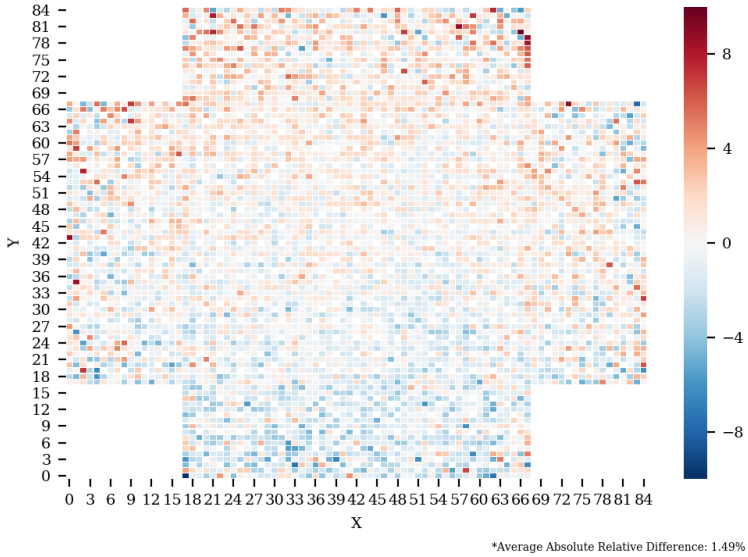


(a) Trial 1 vs. Trial 9

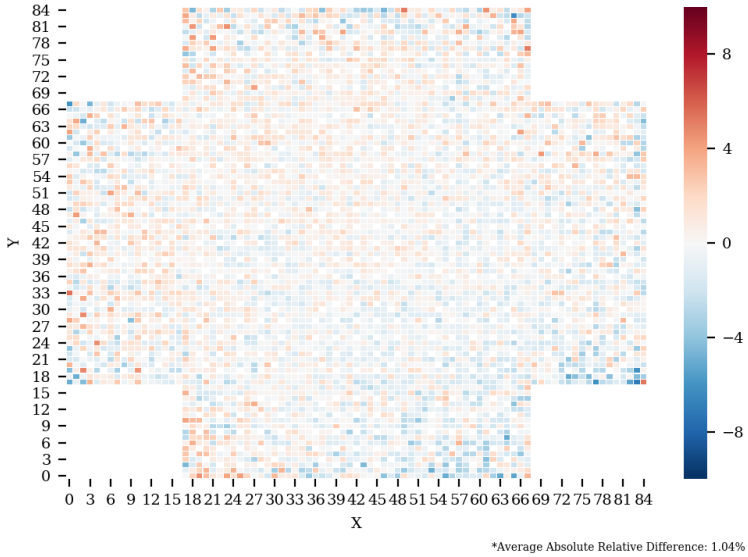


(b) Trial 2 vs. Trial 6

Figure 4.9 Relative difference of fission neutron distribution between Trials 1 - 5 and Trial 6

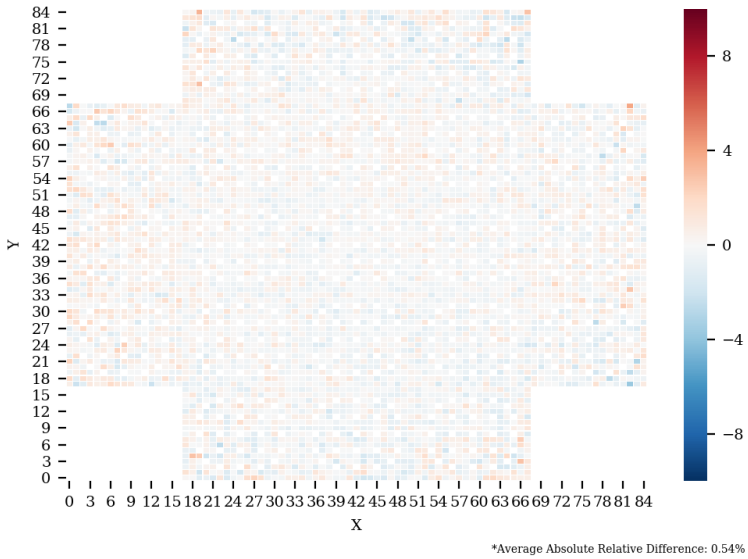


(c) Trial 3 vs. Trial 6



(d) Trial 4 vs. Trial 6

Figure 4.9 Relative difference of fission neutron distribution between Trials 1 - 5 and Trial 6 (cont.)



(e) Trial 5 vs. Trial 6

Figure 4.9 Relative difference of fission neutron distribution between Trials 1 - 5 and Trial 6 (cont.)

Chapter 5

Effect of Time Step Resolution on Isotopic Composition

Fuel burnup calculations are used to determine the time evolution of material compositions in a nuclear reactor core, which is an important part of reactor design and operation. In Monte Carlo based burnup calculations, one must balance the accuracy of the simulation with the computation time needed to perform it. Minimizing the number of burnup time steps used can minimize computation time, however utilizing a low resolution time step scheme can lead to inaccurate results. In this chapter, the effect of time step resolution on assembly-wise material composition as well as computation time are examined.

5.1 Test Description

When performing burnup calculations, one of the most critical parameters to consider is the time step scheme used. In particular, time step resolution can have a measurable effect on the accuracy of a burnup calculation. Time step resolution is defined by the number of depletion steps used in a burnup calculation. A higher time step resolution means that more

burnup steps are being used within a given time interval, or that the Δt time step length is smaller. In order to test the effect of time step resolution on material composition, six time step schemes in Section 5.1.2 and Section 5.1.3 with varying levels of time step resolution within the first 30 days were developed to be used in a burnup calculation using the model described in Chapter 4. The assembly-wise material composition of important isotopes will then be compared in Section 5.2 and run time for each time step scheme in Section 5.3. Isotopic comparisons will be made only at the time steps held in common by all cases (i.e. 10 days, 20 days, 30 days, etc.).

5.1.1 Important Isotopes

The main factor when developing an appropriate time step scheme for fuel burnup calculations is the time evolution of important isotopes. During normal reactor operation, the isotopic make up within each fuel rod changes over time. Isotopes are generated or removed through various transmutation processes such as absorption, decay, and fission. For instance, the primary fissile isotope (initially) in a fuel rod ^{235}U will undergo fission which can produce fission products such as ^{135}Xe. This new material composition will affect things such as reactivity and the evolution of other isotopes; for example, ^{135}Xe is a strong neutron absorber, which will remove neutrons from the reactor core, thereby decreasing the value of k_{eff} over time. Important isotopes are those that have a large impact on the operation of a reactor core. The interaction rate of an isotope is a good indication of the importance of the isotope. Interaction rate can be expressed by

$$RR = \int_V d^3r \int_0^\infty dE \Sigma(\vec{r}, E)\phi(\vec{r}, E) \tag{5.1}$$

where $\phi(\vec{r}, E)$ is the flux, and the integrand is integrated over all energies within a given volume.

Previous work using this model [1] studied the reaction rate of important isotopes over time.

Figure 5.1 shows the reaction rate for fission products, and Figure 5.2 shows the reaction rate for fissile and fissionable isotopes. The default Serpent thermal and fast energy groups, which are divided at 0.625 eV, were used. This is a common delineation point for these energy groups in thermal reactor analysis [40].

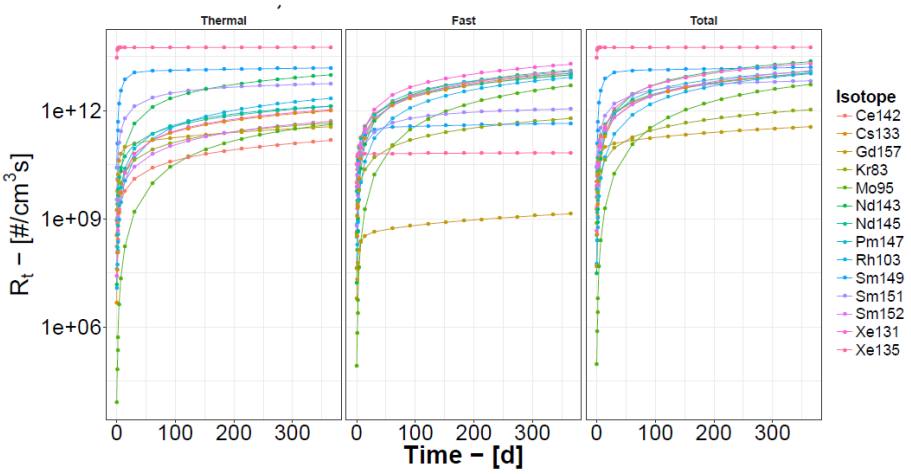


Figure 5.1 Reaction rates for fission products at thermal (below 0.625 eV), fast (above 0.625 eV) and total energy ranges [1]

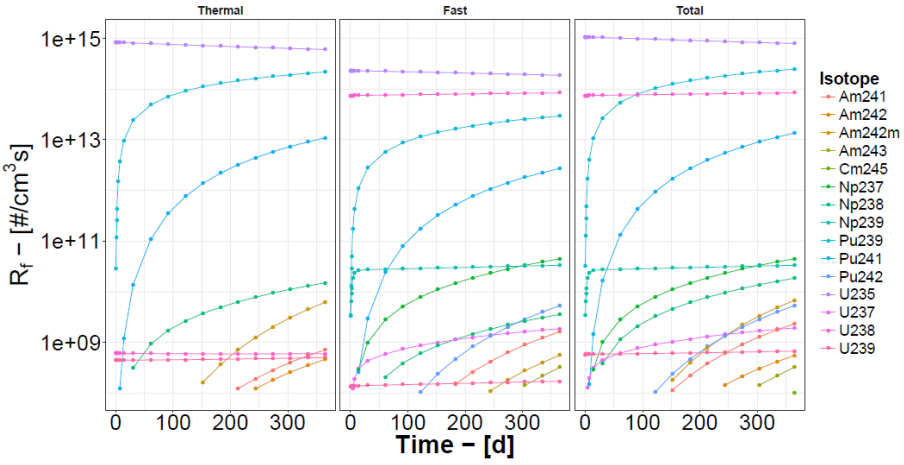


Figure 5.2 Reaction rates for fissile and fissionable isotopes at thermal (below 0.625 eV), fast (above 0.625 eV) and total energy ranges [1]

This work shows that the strongest absorbing fission products were ^{135}Xe, ^{131}Xe, ^{149}Sm, and ^{143}Nd. The highest reaction rates among fissile and fissionable isotopes were from ^{235}U,

^{239}Pu, ^{238}U, and ^{241}Pu. These 8 isotopes, in particular ^{135}Xe, ^{149}Sm, and ^{239}Pu, will be examined as these are the most problematic isotopes to capture in terms of their evolution given their importance.

5.1.2 Constant Time Step Schemes

This test varied the time step resolution for the first 30 days, then kept time steps constant for all cases out to 365 days. Five different cases using a constant Δt were chosen, based on previous studies using this model with different time steps. These five cases are presented in Table 5.1. Due to the complexity of Monte Carlo burnup methodologies, most code systems do not provide material composition uncertainty values. Therefore, Case 5 will act as the 'reference' calculation case, as it utilizes 48 burnup steps within the first 30 days, and has the smallest Δt of the constant step size cases.

Table 5.1 Constant Time Step Schemes

Case	**Δt (days)**	**No. of Steps**
1	10	3
2	5	6
3	2.5	12
4	1.25	24
5	0.625	48

5.1.3 Variable Time Step Scheme

In order to accurately capture important isotope evolution in the core, a 'Variable step' scheme (also called Case 6) using a combination of Δt time steps in the first 30 days ranging from 0.25 days to 10 days was tested as well. This scheme was found heuristically through experimentation as time step scheme development is highly problem specific. However, there

are some general guiding principles to follow when creating a time step scheme. For example, one must attempt to capture the evolution of important isotopes with a time step scheme particularly in non-linear areas of growth or decline in composition, while taking care not to reach a point of diminishing returns with respect to computation time. Figure 5.3 shows ^{135}Xe concentration as a function of time for a constant and variable time interval compared with the reference calculation. This figure illustrates the method of picking steps that characterize isotopic evolution. The constant Δt scheme places far too many points in a region where the ^{135}Xe atom density is roughly linear and has few steps where ^{135}Xe is rapidly evolving; in contrast, the variable Δt scheme places more points in areas where there is rapid change, and less where ^{135}Xe has reached equilibrium. Despite requiring approximately the same amount of computation time, the variable step scheme will produce accurate results.

While there is no definitive algorithm for picking a time step scheme, there is a general procedure one can use to develop a new scheme:

1. Create an initial time step scheme and perform a burnup calculation. Initial time step schemes should generally have more steps early on (in the first 30 days) and fewer in later stages. Some important isotopes such as ^{135}Xe rapidly reach an equilibrium (within 4 days in this model), and therefore must be modeled accordingly.

2. Identify important isotopes. Important isotopes are those with the highest reaction rates in the model.

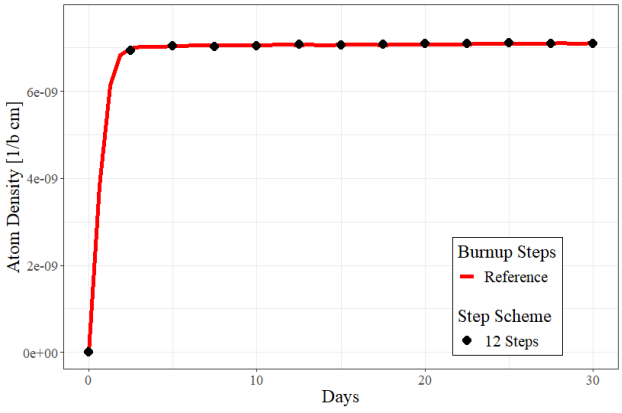
3. Analyze and characterize the evolution of important isotopes over time to find areas of growth, and areas of equilibrium. Specifically, areas of exponential growth (particularly over small time periods) should be taken note of. The example of ^{135}Xe, which rapidly grows then turns into a flat distribution within 4 days, illustrates this.

4. Add time step points in areas of non-linear change in isotopic composition for important isotopes.

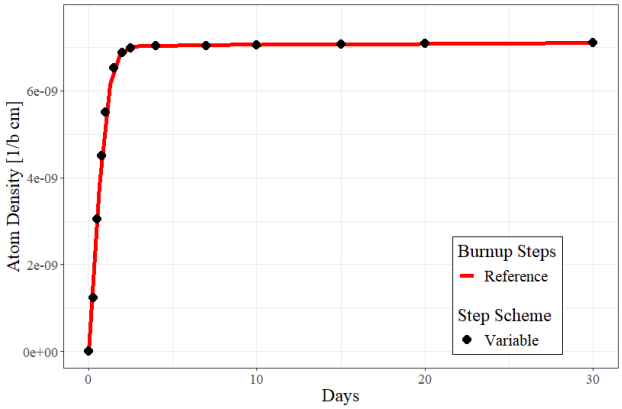
5. Decrease time step points in areas of linear change in isotopic composition for important

isotopes. Once an isotope is in equilbrium, changes in concentration will likely be small and therefore time steps do not need to be used as often.

6. Repeat the process of adding or subtracting time step points in important areas in order to achieve optimal accuracy while minimizing computation time.



(a) 12 constant Δt scheme (Case 3) vs. reference



(b) Variable step scheme (Case 6) vs. reference

Figure 5.3 Constant Δt and variable Δt step schemes compared to the reference atom density evolution for ^{135}Xe

Table 5.2 details the variable step scheme, and Figure 5.4 illustrates each case's time step scheme for the first 30 days.

Table 5.2 Variable Time Step Scheme (Case 6)

Step	Cum. Days	Step	Cum. Days
1	0.25	8	4.0
2	0.5	9	7.0
3	0.75	10	10.0
4	1.0	11	15.0
5	1.5	12	20.0
6	2.0	13	30.0
7	2.5		

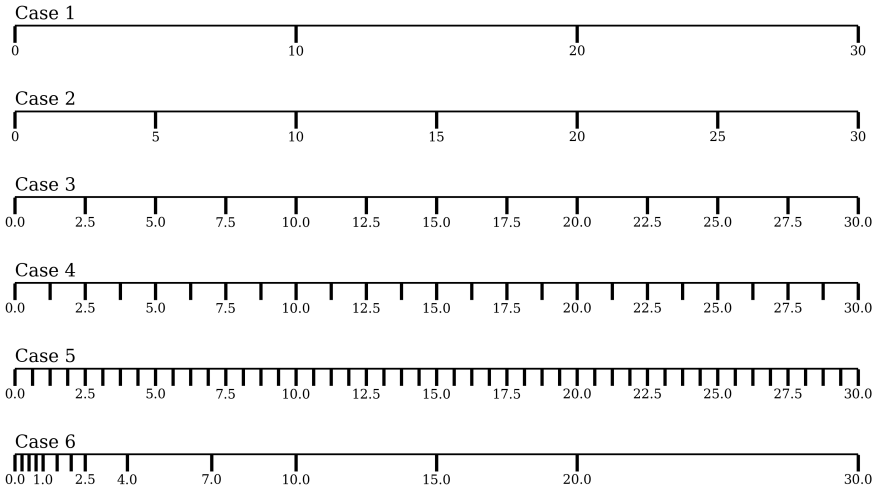


Figure 5.4 Time Step Scheme Cases

5.2 Isotopic Composition

The results of this section will be analyzed by calculating the percent relative difference of the atom density between Case 5, the reference, and all of the other cases. Percent relative

difference is given by

$$\% \text{ Relative Difference}_{case,t} = 100 \times \frac{N_{case,t} - N_{reference,t}}{N_{reference,t}} \tag{5.2}$$

where $N_{case,t}$ is the atom density of a given isotope for a case at time step t, and $N_{reference,t}$ is the reference atom density at time step t.

These relative differences and the time evolution of important isotopes will be calculated for a single corner assembly, labelled '1' in Figure 5.5. This assembly was chosen due to its distance from the center of the reactor core. Flux in a reactor drops as one moves away from the center, which leads to issues with convergence and depletion calculation accuracy. Thus, a corner assembly (being the most problematic assembly) will be an appropriate limit case to test the efficacy of different step schemes throughout the reactor core.

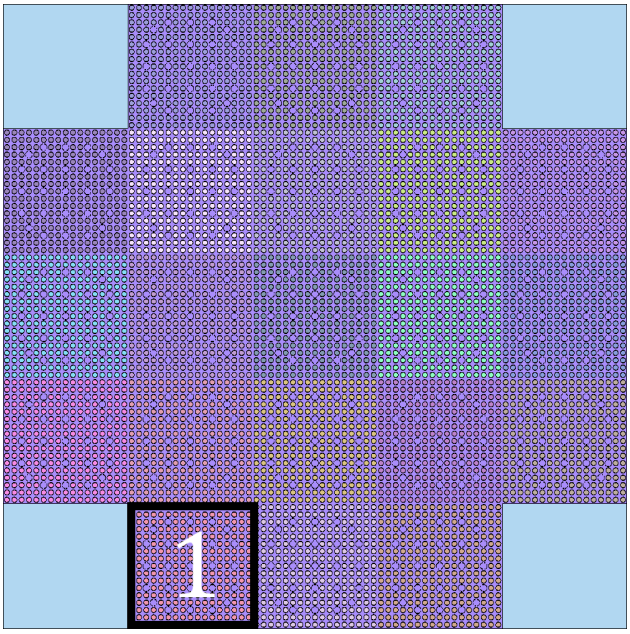


Figure 5.5 Corner assembly '1' used for testing isotopic composition

Figures 5.6a through 5.13 show isotopic evolution and the relative difference comparisons between each case. The gray vertical box in each relative difference plot indicates the first 30 days, where each case is different, and the blue horizontal box displays the minimum and

maximum percent relative difference of Case 4, which is a case with 24 burnup steps in the first 30 days. Due to a lack of material composition uncertainty values, this blue box was presented in order to demarcate the range of relative difference values that Case 4 exhibited, in order to provide a point of reference for Case 6. The solid black line connects the points of Case 6 in the relative difference plots.

Despite having 11 fewer burnup steps, the relative difference values of Case 6 fell outside of the 'range' of Case 4 only twice in the isotopes evaluated. For most isotopes, Cases 4 and 6 have nearly zero relative difference within the first 30 days, whereas Cases 1 through 3 show issues within this timeframe, particularly with ^{149}Sm, ^{131}Xe, ^{239}Pu, and ^{241}Pu where the lower resolution time step schemes appear to underestimate the values. Most of the important isotopes appear adequately captured for all steps by Cases 4 and 6, with the exception of ^{241}Pu which was difficult to capture by all time step schemes. This is likely due to the consistent exponential behavior of this isotope evolution, which does not exhibit a period of equilibrium or linear behavior. This issue can be remedied with greater step resolution in later steps, between 100 and 300 days where the isotope shows large exponential growth.

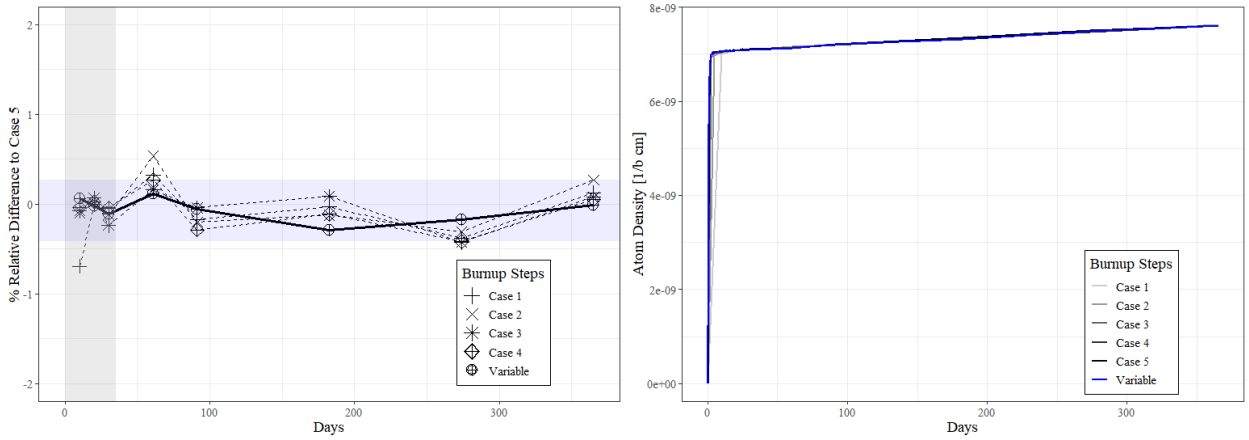
All of the results shown fall within ± 2% relative difference, which one may find acceptable if only looking to evaluate isotopes at given burnup step values. However, significant issues arise when attempting to interpolate material composition values between burnup steps using a low resolution time step scheme. The *b*RAPID methodology uses one such interpolation scheme, and so it is imperative when developing a time step scheme for *b*RAPID database development that the scheme adequately describes the isotopic evolution curves of important isotopes.

To help quantify the degree to which the isotopic atom density varies from the reference for each case, the average absolute relative difference, or AARD, for each case and isotope will

be calculated. This average is calculated using the expression

$$\text{AARD} = \frac{1}{T}\sum_{t}^{T} |\%\ \text{Rel. Diff}_{case,t}| \tag{5.3}$$

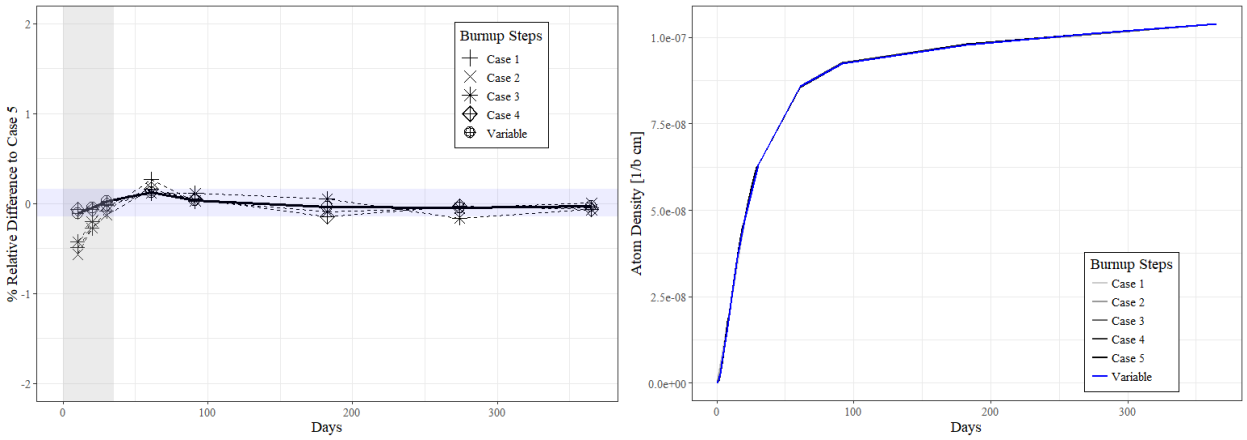
where T is the total number of time steps used in the relative difference calculation, t is each time step, and $|\text{Rel. Diff}_{case,t}|$ is the absolute value of the relative difference for a given isotope of *case* at time step t calculated with Equation 5.2. The AARD is similar to a normal average, only the values averaged are absolute to deal with negative relative difference values. The result of these calculations are shown in Figure 5.14. One can see a decreasing trend among all isotopes from Case 1 to Case 6, indicating that an increase in time step resolution increases accuracy. However, Case 6 (the variable time step case) is seen to outperform Case 4, which has more time steps– showcasing the improvement due to this methodology. AARD values are explicitly shown in Table 5.3.



(a) ^{135}Xe relative difference for each case

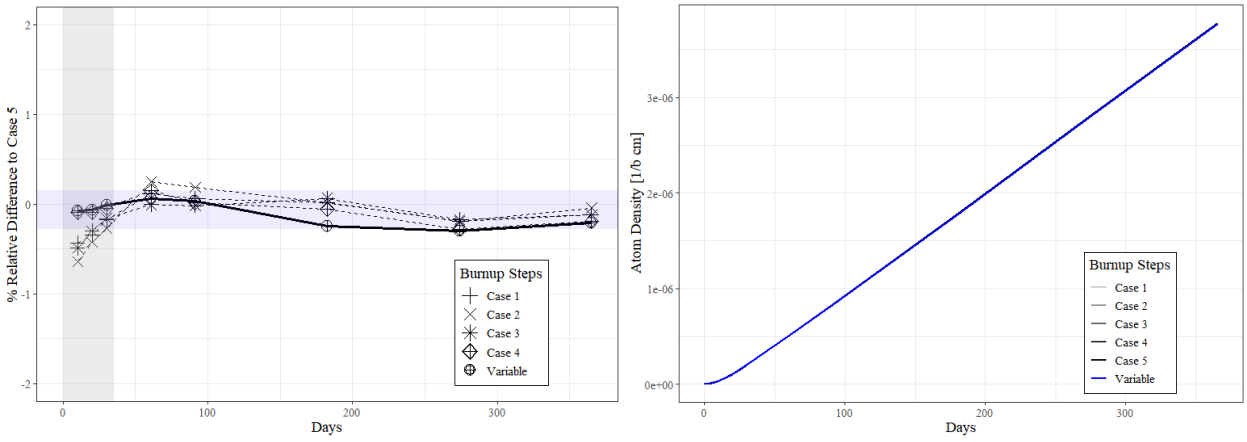
(b) ^{135}Xe concentration time evolution for each case in units of atom density

Figure 5.6 ^{135}Xe comparisons for each case in corner assembly



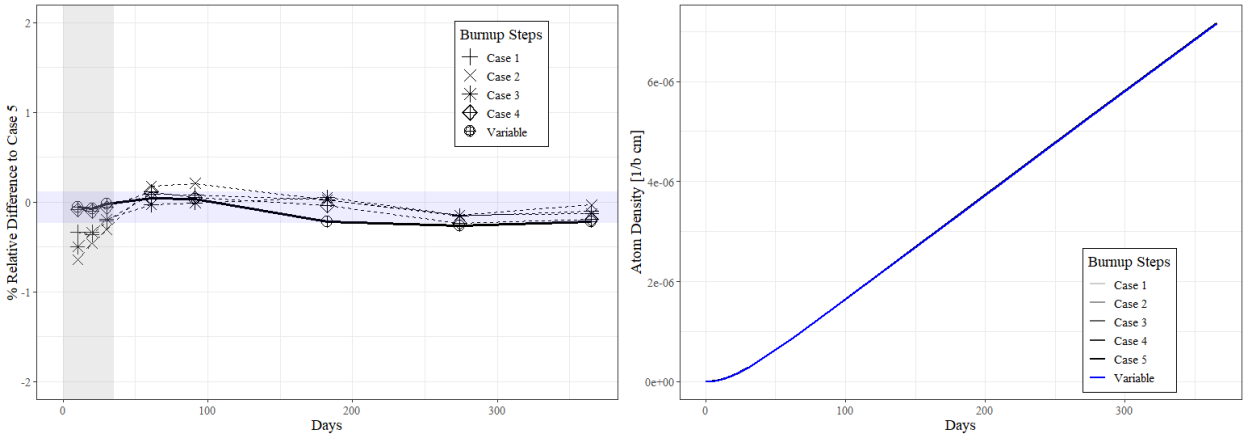
(a) ^{149}Sm relative difference for each case **(b)** ^{149}Sm concentration time evolution for each case in units of atom density

Figure 5.7 ^{149}Sm comparisons for each case in corner assembly



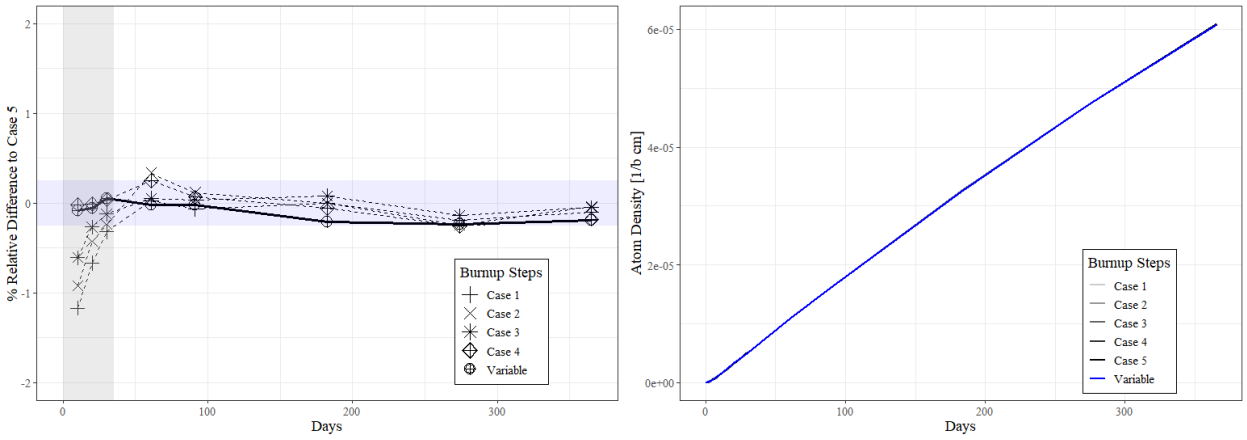
(a) ^{131}Xe relative difference for each case **(b)** ^{131}Xe concentration time evolution for each case in units of atom density

Figure 5.8 ^{131}Xe comparisons for each case in corner assembly



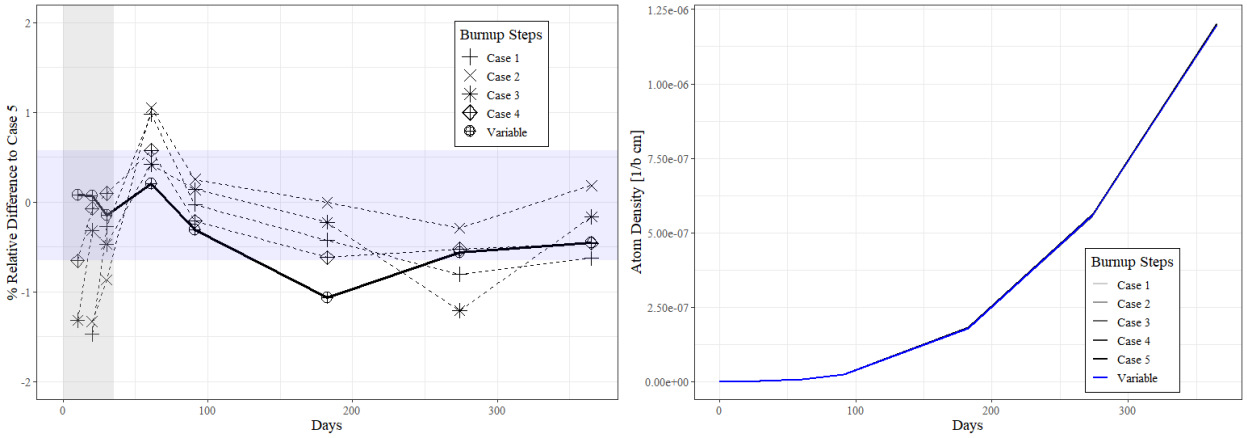
(a) ^{143}Nd relative difference for each case **(b)** ^{143}Nd concentration time evolution for each case in units of atom density

Figure 5.9 ^{143}Nd comparisons for each case in corner assembly



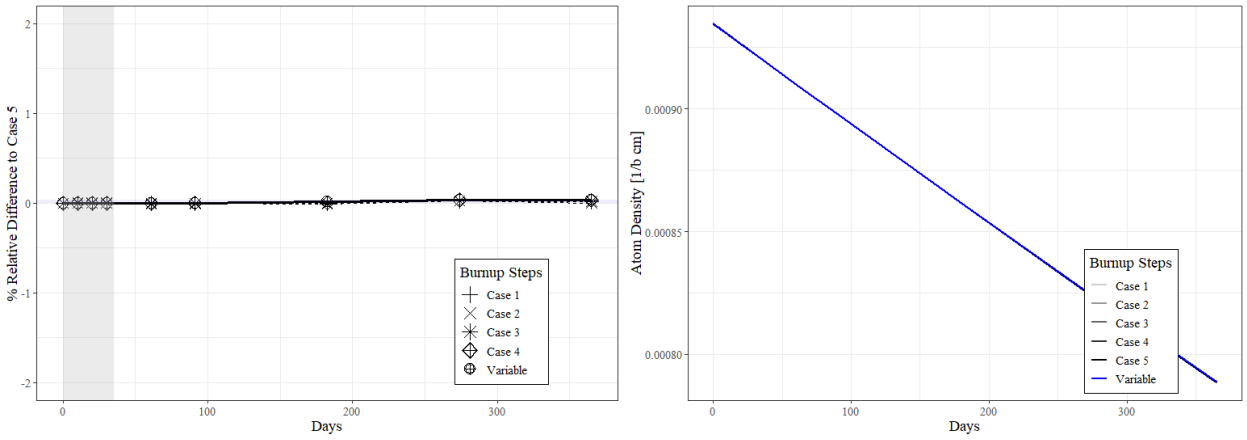
(a) ^{239}Pu relative difference for each case **(b)** ^{239}Pu concentration time evolution for each case in units of atom density

Figure 5.10 ^{239}Pu comparisons for each case in corner assembly



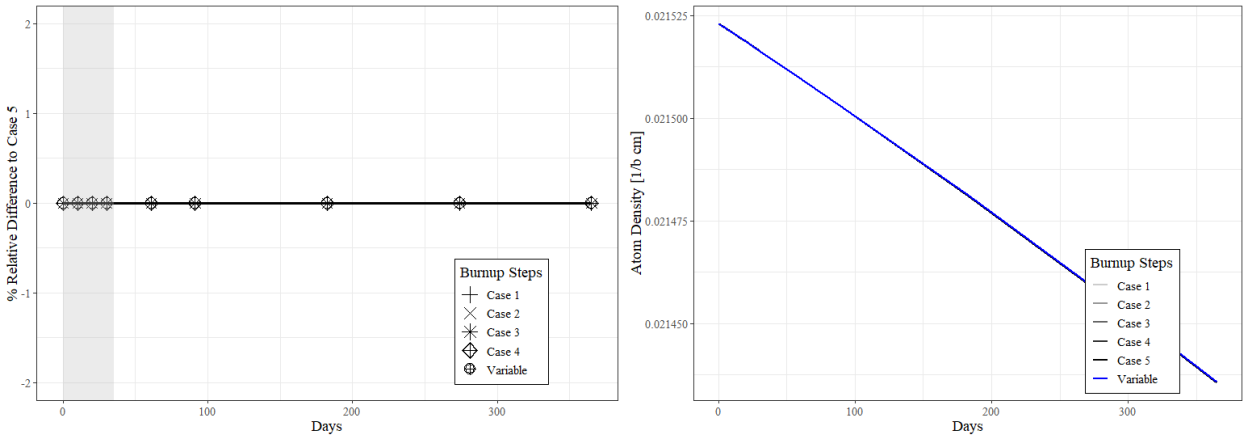
(a) ^{241}Pu relative difference for each case **(b)** ^{241}Pu concentration time evolution for each case in units of atom density

Figure 5.11 ^{241}Pu comparisons for each case in corner assembly



(a) ^{235}U relative difference for each case **(b)** ^{235}U concentration time evolution for each case in units of atom density

Figure 5.12 ^{235}U comparisons for each case in corner assembly



(a) ^{238}U relative difference for each case

(b) ^{238}U concentration time evolution for each case in units of atom density

Figure 5.13 ^{238}U comparisons for each case in corner assembly

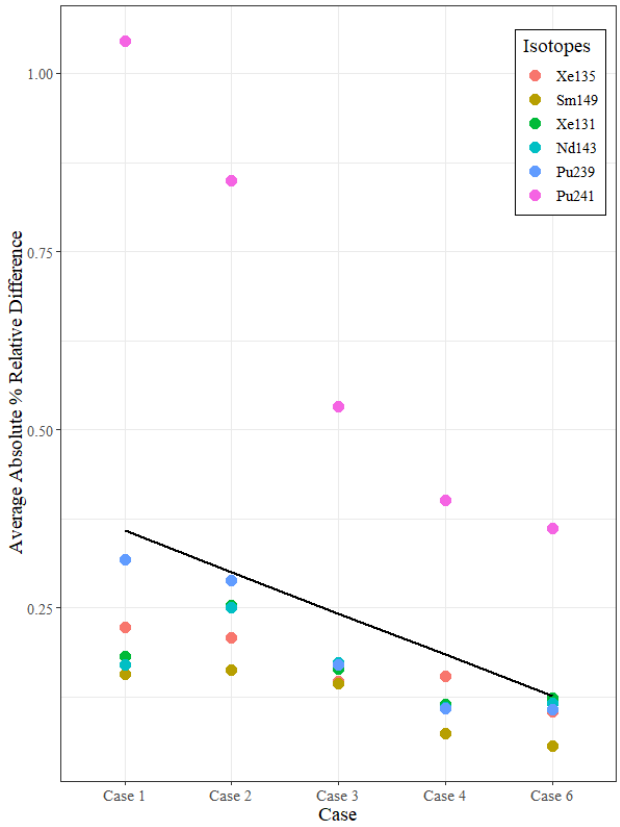


Figure 5.14 Average absolute relative differences for important isotopes, for all cases. A black regression line shows the trend of the data between cases

5.3 Time Step Scheme Calculation Times

Another consideration when developing a time step scheme is calculation time. Ideally, a scheme will be both accurate and fast; however, with increased time step resolution comes increased computation time. As one can see in Table 5.3, the variable time step scheme (Case 6) had a calculation time nearly just over 4 hours less than the comparable Case 4 calculation time. The 'speedup' values are simply a ratio between the calculation time of the reference case (Case 5) and each other Case.

Combining the results from Section 5.2 with these calculation times, it is apparent that a variable time step scheme accomplishes the goal of producing accurate results while minimizing computation time. The variable scheme achieves material composition values generally within ± 1% of the reference case, in less than half of the computation time. AARD values show the variable scheme generally performing better than constant time step schemes (1-4), with only Case 4 showing comparable values for certain isotopes.

Table 5.3 Case calculation times and AARD values

Case	No. of Steps	Run Time *	Speedup	AARD ^{135}Xe	^{149}Sm	^{131}Xe	^{143}Nd	^{239}Pu	^{241}Pu
1	3	7.48	5.2	0.223	0.157	0.182	0.170	0.318	1.045
2	6	8.46	4.6	0.207	0.163	0.253	0.250	0.289	0.850
3	12	12.75	3.1	0.146	0.144	0.165	0.173	0.170	0.532
4	24	22.06	1.8	0.154	0.074	0.115	0.110	0.108	0.401
5	48	39.13	1.0	–	–	–	–	–	–
6	13	17.45	2.2	0.104	0.056	0.123	0.116	0.108	0.361

* Run Time has units of wall-clock hours, performed on 8 processors

The effectiveness of each case, with respect to their run times, may be evaluated by introducing a Figure of Merit (FOM) for each isotope. The FOM takes into account the time required to complete a burnup calculation using a case, and the accuracy of said case as

measured by AARD. The expression is given by:

$$\text{FOM} = \frac{\text{Speedup}}{\text{AARD}} \tag{5.4}$$

One can see that a higher speedup and lower AARD increases the FOM– thus, a high FOM is desired. FOM's are calculated for each case and isotope in Table 5.4. These values indicate that Case 6 (variable time steps) is more efficient at reducing relative difference in the run time it uses than the comparable Case 4. The table also shows that, if one is not too concerned with highly accurate results, Case 1 may prove to be most efficient due to its very low computation time.

Table 5.4 FOM of each case for important isotopes

	FOM					
Case	^{135}Xe	^{149}Sm	^{131}Xe	^{143}Nd	^{239}Pu	^{241}Pu
1	23.3	33.1	28.5	30.5	16.3	4.9
2	22.2	28.2	18.1	18.4	15.9	5.4
3	21.2	21.5	18.7	17.9	18.2	5.8
4	11.6	24.3	15.6	16.3	16.6	4.4
6	21.1	39.2	17.8	18.9	20.3	6.0

This study indicates that time step schemes should be developed specifically with isotopic evolution in mind, and molding time step resolution to important areas of isotopic growth is more efficient than simply increasing time step resolution across the board. Further studies into this area could include increasing the scope of the study; the final burnup value at 365 days was only 9.13 GWd/tHM, whereas isotopic evolution continues to occur well past this burnup value in regular reactor operation.

Chapter 6

Development of a Time-Dependent Boundary Correction Algorithm

RAPID database coefficients are normally generated in the center of a reactor environment and then applied to every assembly in a RAPID model– however, models often have assemblies that are surrounded by other materials, such as water, creating boundary effects in the fission source distribution. This dissonance can lead to inaccurate results for k_{eff} , fission source distribution, and other quantities. In order to correct this, the RAPID user manual [35] lays out a procedure for creating what is known as a "boundary correction". This procedure is useful for fresh fuel RAPID calculations, however previously there was no boundary correction procedure or algorithm for RAPID calculations that change in material composition over time, as is the case with *b*RAPID calculations. This Chapter presents a new algorithm developed for *b*RAPID that allows users to utilize time-dependent boundary corrections for burnup calculations, and then compares the results of *b*RAPID calculations using time-dependent boundary corrections with the original time-independent boundary correction method.

6.1 Boundary Correction Methodologies

The purpose of a RAPID boundary correction is to correct for the aforementioned differences in physics between a central assembly region and a boundary assembly region. FM coefficients, as described in Chapter 3, are generally generated in the central area of a reactor model and then used to describe the entire reactor model by translating the coefficients using geometric symmetries. Several key assumptions are made during this process, most of which are valid except for at the boundary regions of the reactor where radial flux behavior differs significantly between a central assembly and a boundary assembly. These differences in physics become more apparent in smaller reactor models, such as the 5x5 'mini-core' model.

In order to achieve accurate results, a methodology for creating boundary corrections was developed as described in Section 6.1.1. However, this methodology was developed for use in a static, fresh fuel RAPID calculation. The development of the *b*RAPID burnup algorithm created a need for time-dependent boundary corrections, as assembly-wise material compositions change during reactor operation leading to a non-uniform distribution. Section 6.1.2 describes the time-dependent boundary correction algorithm for *b*RAPID and its basic inputs.

6.1.1 Time-Independent Boundary Correction Methodology

The fundamental concept behind a boundary correction is to create a ratio that describes the difference in behavior caused by creating FM coefficients using a central assembly environment rather than a boundary assembly environment. To this end, the standard time-independent boundary correction method, here written as $bnd(x, y)$, provides a simple solution. There are three steps used to create a time-independent boundary correction as laid out in the RAPID user manual [35]:

1. Run the standard model of the system (Figure 6.1a) with a uniform fixed source in the

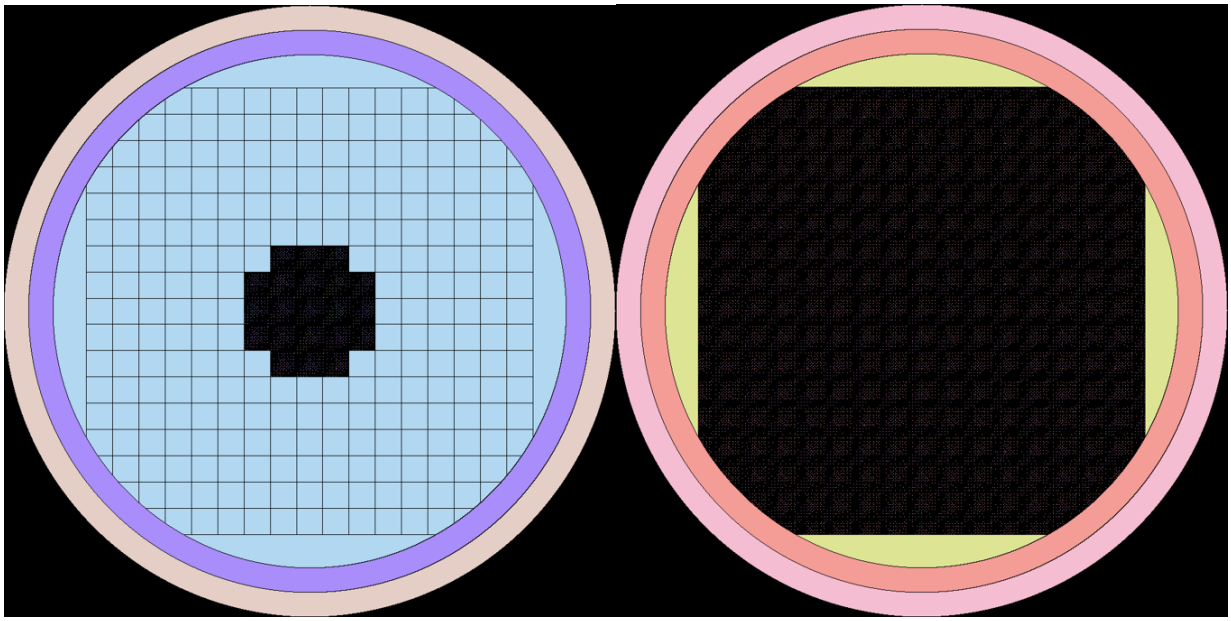
fuel region, with fission turned off. Tally the fission neutron production in each pin to create a distribution, $f(x, y)$

2. Create a new model (Figure 6.1b) with an infinite array of assemblies, and run using the same uniform fixed source in the original fuel region, with fission turned off. Tally fission neutron production, to create the infinite distribution, $\tilde{f}(x, y)$.

3. Calculate the FM coefficient boundary correction factors (Equation 6.1):

$$bnd(x, y) = \frac{f(x, y)}{\tilde{f}(x, y)} \tag{6.1}$$

Every assembly in this procedure has the same material composition.



(a) Regular model **(b)** Infinite model

Figure 6.1 Regular and Infinite model geometries used in $bnd(x, y)$ creation

Once generated, $bnd(x, y)$ can be applied to the FM coefficients of a RAPID model, which can be done by specifying the `bndxy.dat` file in the \db folder of a RAPID database. Equation 6.2 describes this:

$$a_{i,j} = bnd(x_i, y_i)\tilde{a}_{i,j} \tag{6.2}$$

where $a_{i,j}$ is the corrected coefficient, and $\tilde{a}_{i,j}$ is the uncorrected coefficient.

Table 6.1 compares the k_{eff} values of a RAPID run for the 5x5 model with and without a boundary correction as compared with the reference case results.

Table 6.1 Calculated fresh fuel eigenvalue comparison between Serpent reference and *b*RAPID (with and without boundary correction applied) and relative differences in pcm

Serpent Reference* k_{eff}	***b*RAPID without bnd(x,y)** k_{eff}	**Rel. Diff. (pcm)**	***b*RAPID with bnd(x,y)** k_{eff}	**Rel. Diff. (pcm)**
1.13263	1.12905	-316	1.13126	-121

* Serpent $\sigma = \pm$ 10 pcm

These results show a significant improvement (∼200 pcm) in the k_{eff} value when using the boundary correction methodology due to the accurate representation of boundary physics.

6.1.2 Time-Dependent Boundary Correction Methodology

Similar to the $bnd(x, y)$ method, the time-dependent boundary correction method labeled here $bnd(x, y, t)$ is used to correct FM coefficients by using a ratio of a regular model and an infinite model. The key difference in the $bnd(x, y, t)$ methodology is that instead of using a uniform material distribution, $bnd(x, y, t)$ utilizes the assembly-wise material distribution for each burnup step in order to reflect the changing isotopics of the system over time. Instead of a single boundary correction, the $bnd(x, y, t)$ methodology involves creating a boundary correction for each desired time step that can be applied automatically during *b*RAPID calculations.

The methodology for developing a time-independent correction is similar to that of the $bnd(x, y)$ correction but holds some key differences. The general $bnd(x, y, t)$ procedure is as follows:

1. Perform a set of burnup calculations to obtain assembly-wise material composition for different time steps for the standard model. Figure 6.2a labels assemblies that are parsed for the 5x5 material distribution

2. Create a standard model (Figure 6.1a) for time Δt with the appropriate material distribution obtained from the previous step

3. Run the standard model of the system, using a uniform fixed source in the fuel region and with fission turned off. Tally the fission neutron production in each pin to create a distribution, $f(x, y, t)$

4. Create a new infinite model (Figure 6.1b) by surrounding the standard model from step 2 with assemblies composed of step t 'corner' and 'side' material compositions. Figure 6.2b illustrates the material compositions used in the infinite 5x5 model. Corner assemblies (defined by having two sides in contact with water) should be surrounded by material composition parsed from a corner assembly, such as assembly 21. Side assemblies should be adjacent to materials parsed from a side assembly, like assembly 9

5. Run infinite model using the same uniform fixed source in the original fuel region, with fission turned off. Tally fission neutron production, to create the infinite distribution, $\tilde{f}(x, y, t)$.

6. Calculate the FM coefficient boundary correction factors for each step (Equation 6.3):

$$bnd(x, y, t) = \frac{f(x, y, t)}{\tilde{f}(x, y, t)} \tag{6.3}$$

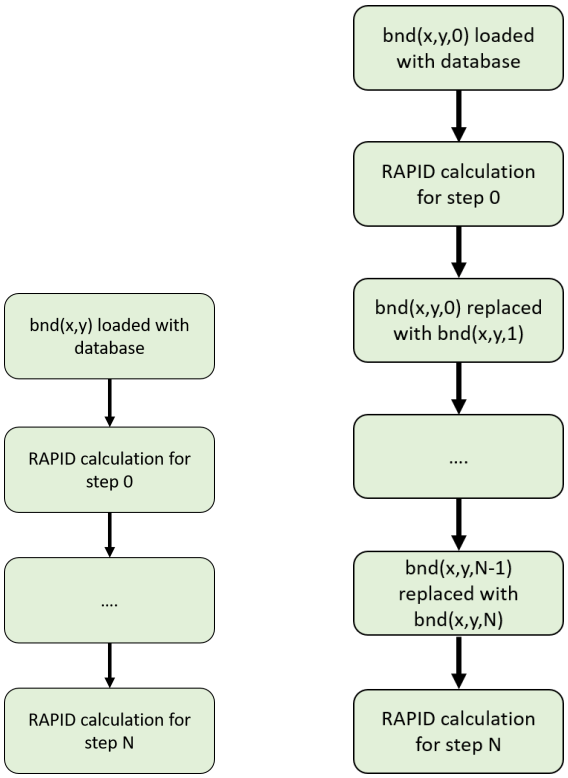
(a) Regular model assembly map **(b)** Infinite model assembly map

Figure 6.2 Regular and Infinite assembly material composition maps used in $bnd(x, y, t)$ creation

At each time step, the same calculation from Equation 6.2 is performed. It is important to note that the choice of Assembly 21 in the infinite model, which is used to surround corner assemblies, is arbitrary insofar as any corner assembly material composition can be used to fill the infinite lattice (such as 1,3,19, etc.). Due to model symmetries, the corner assemblies have very similar material compositions– thus little difference would be made by exchanging one assembly for another. However, in the future, an average corner assembly type composition or a combination of assemblies could be used to fill the lattice. Also note that the calculation for step 1 (described above) is separate from the burnup pre-calculation stage of RAPID database creation. Boundary correction factors, in general, are calculated separately from RAPID databases. In the future, these two burnup calculations may be integrated into a single process.

Figure 6.3 shows the difference between using $bnd(x, y)$ and $bnd(x, y, t)$ in a bRAPID calculation. One can see that the original method simply uses the 0^{th} step boundary correction

for all time steps– whereas the $bnd(x, y, t)$ method switches out the boundary correction at each time step specified.



(a) $bnd(x, y)$ **(b)** $bnd(x, y, t)$

Figure 6.3 Flowchart illustrating the operational difference between $bnd(x, y)$ and $bnd(x, y, t)$ in bRAPID calculations

In order to use time-dependent boundary corrections in bRAPID, the appropriate time-dependent boundary correction files for each step should be placed in the \db, and should be named `bnd2xy_sN.dat` where N is the step number associated with that boundary correction. Figure 6.4 shows a sample of what a database with $bnd(x, y, t)$ boundary correction files looks like.

For example; if one would like to use a boundary correction for steps 0, 1, 2, and 5, one would:

1. Create the boundary correction files

2. Label them `bnd2xy.dat`, `bnd2xy_s1.dat`, `bnd2xy_s2.dat`, `bnd2xy_s5.dat`

3. Place the boundary correction files in the \db folder

and then run *b*RAPID normally.

One important consideration when creating time-dependent boundary correction files is the number of particles to use in the uniform fixed source calculation. Some early testing has shown that, for the 5x5 model presented in this thesis, using 800 million particles produces boundary corrections with less statistical variation than boundary corrections using less particles (say, 100 million). The appropriate number of particles to use in this fixed source calculation should be the subject of sensitivity studies conducted prior to using the $bnd(x, y, t)$ methodology. Uniform fixed-source calculation times vary widely based on the number of particles used– anywhere from 20-30 minutes to hours (at high particle numbers). For the reference calculations performed here at a high number of particles, fixed-source calculation times each required approximately 5 hours on 8 processors. If there is little step-to-step change in material composition in a model, it will likely be unnecessary to use a different boundary correction factor for each time step. This should also be a subject of sensitivity studies for a given model.

Figure 6.4 $bnd(x, y, t)$ files in bRAPID database folder, labeled bnd2xy_sN.dat where N are the step numbers specified by the t_bnd line in burn.inp

6.2 Comparison Between Time-Dependent and Time-Independent Boundary Corrections

Thus far, the original $bnd(x, y)$ methodology and the time-dependent boundary correction $bnd(x, y, t)$ methodology have been introduced, as well as their operation in RAPID. In this section, the effect of applying the $bnd(x, y, t)$ methodology to bRAPID for the 5x5 'mini-core' model will be explored. As shown in Table 6.1, when comparing k_{eff} , the relative differences between Serpent and bRAPID generally fall on the order of 100's of pcm, or 0.1%. Needless to say, these differences are very small, and thus care should be taken when comparing results in order to ensure that any changes seen are due to the methodology rather than random chance.

The k_{eff} eigenvalue and pin-wise fission source distribution of Serpent, bRAPID using

$bnd(x, y)$, and bRAPID using $bnd(x, y, t)$ will be compared to ensure a holistic view of the effect of each methodology. The burnup steps used for this calculation are shown in Table 6.2. The bRAPID database used for this run consisted of the following parameters:

- 1 assembly type
- 5 power densities (0.1, 0.2, 0.3, 0.4, 0.5)
- 19 burnup steps (same as Table 6.2)

The Serpent reference calculation was performed using the parameters chosen in Chapter 4:

- NPS = 100,000
- NAC = 500
- NSK = 200

Table 6.2 Time step scheme used for bRAPID testing

Step	**Cum. Days**	**Step**	**Cum. Days**	**Step**	**Cum. Days**
1	0.5	8	30.438	15	243.5
2	1.0	9	60.875	16	273.938
3	1.5	10	91.313	17	304.376
4	2.0	11	121.751	18	334.813
5	4.0	12	152.188	19	365.25
6	7.0	13	182.625		
7	14.0	14	213.063		

Figure 6.5 shows the boundary correction factors used for the $bnd(x, y)$ tests, and Figure 6.6 shows the boundary correction factors for $bnd(x, y, t)$ for steps 1 through 19. One can see an increase in the boundary correction factors over time in Figure 6.6, particularly along

the perimeter. Figure 6.7 shows the average boundary correction factor for both the time-dependent and time-independent cases.

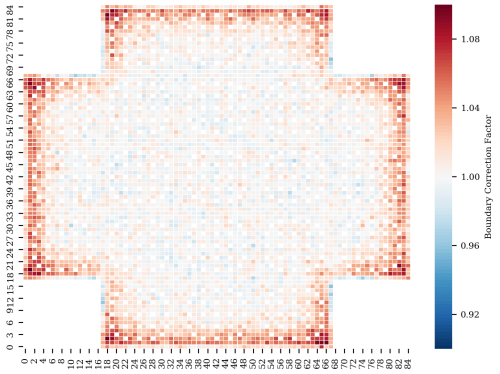
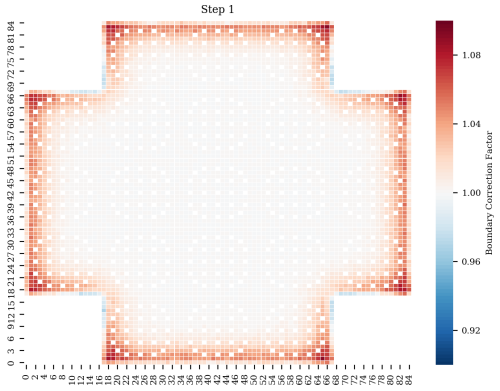
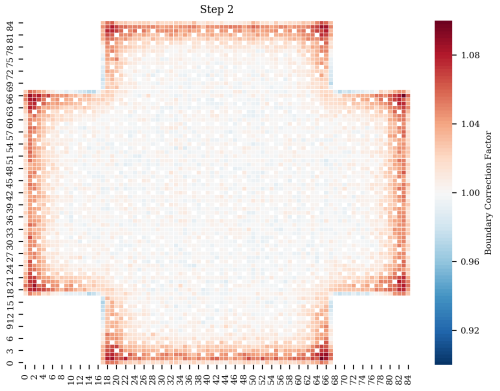


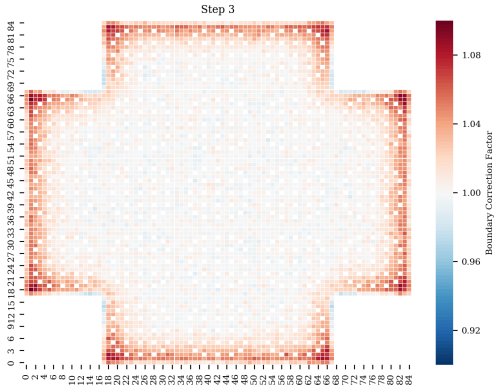
Figure 6.5 $bnd(x, y)$ boundary correction factor



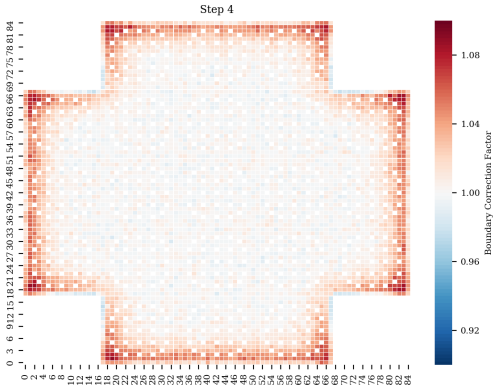
(a) $bnd(x, y, t)$ step 1



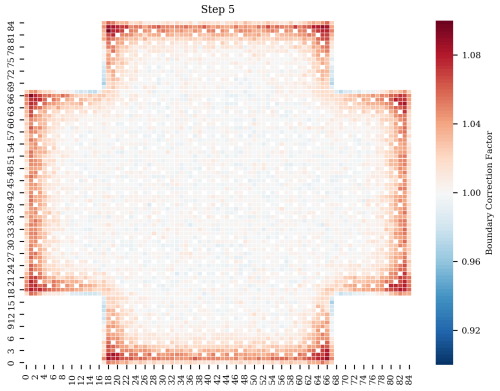
(b) $bnd(x, y, t)$ step 2



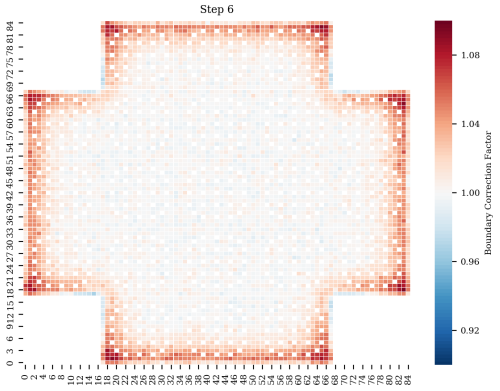
(c) $bnd(x, y, t)$ step 3



(d) $bnd(x, y, t)$ step 4

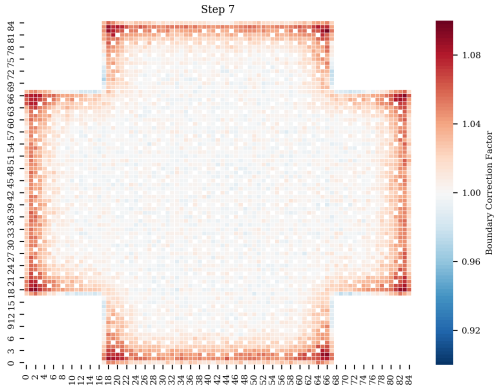


(e) $bnd(x, y, t)$ step 5

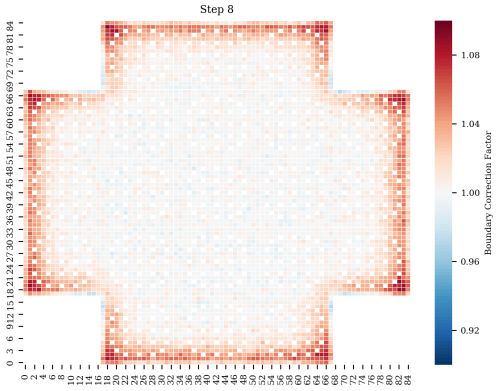


(f) $bnd(x, y, t)$ step 6

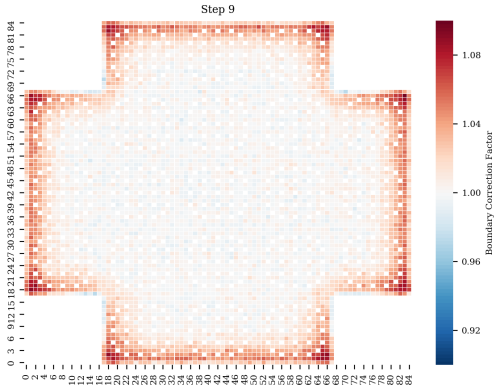
Figure 6.6 $bnd(x, y, t)$ boundary correction factors



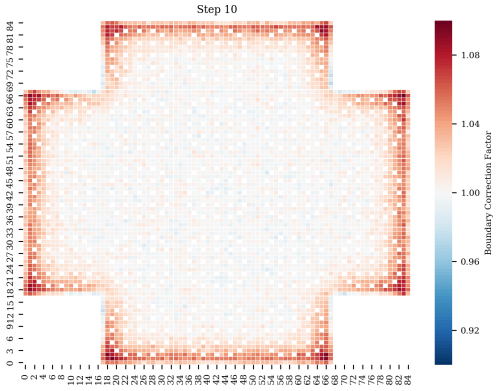
(g) $bnd(x, y, t)$ step 7



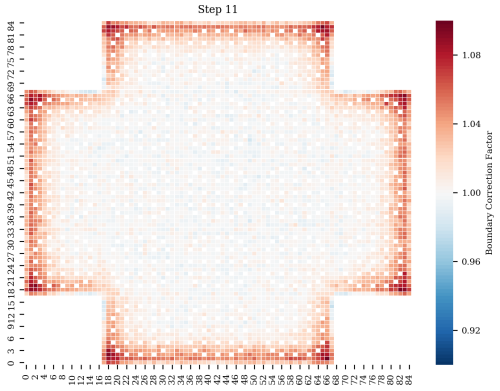
(h) $bnd(x, y, t)$ step 8



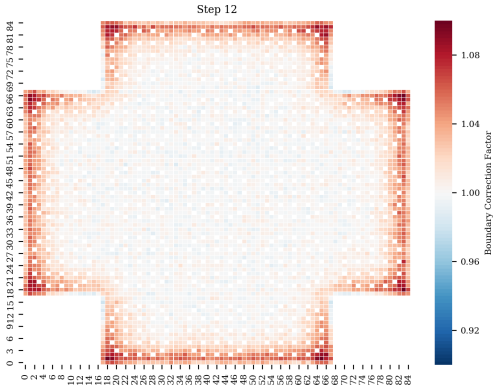
(i) $bnd(x, y, t)$ step 9



(j) $bnd(x, y, t)$ step 10

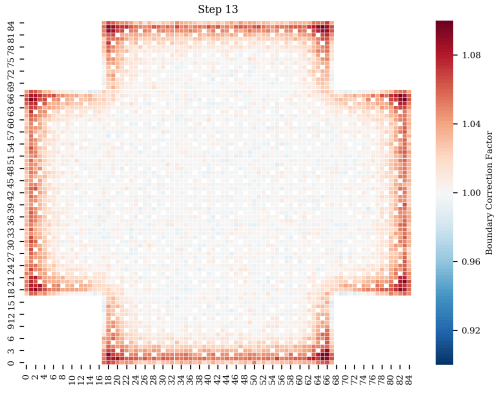


(k) $bnd(x, y, t)$ step 11

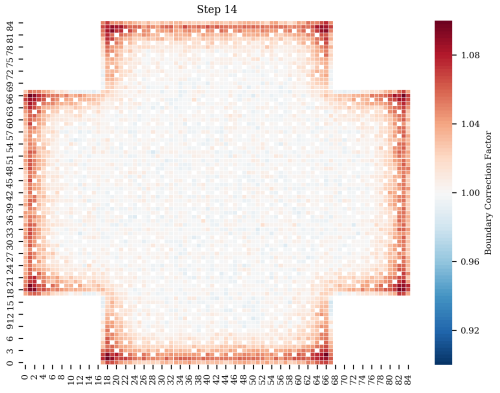


(l) $bnd(x, y, t)$ step 12

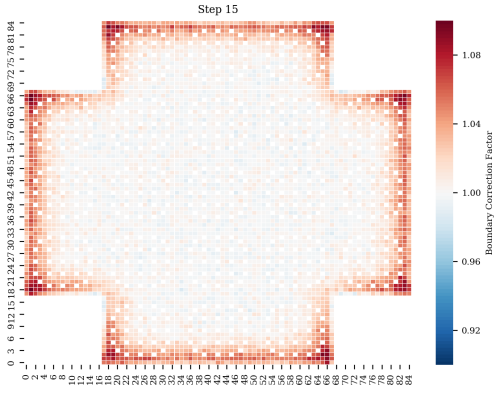
Figure 6.6 $bnd(x, y, t)$ boundary correction factors (cont.)



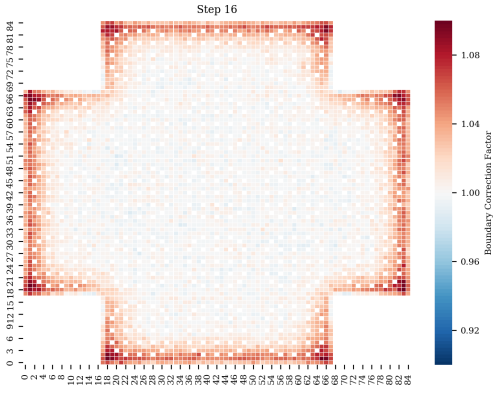
(m) $bnd(x, y, t)$ step 13



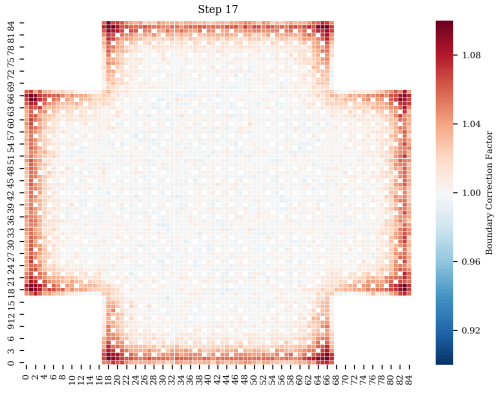
(n) $bnd(x, y, t)$ step 14



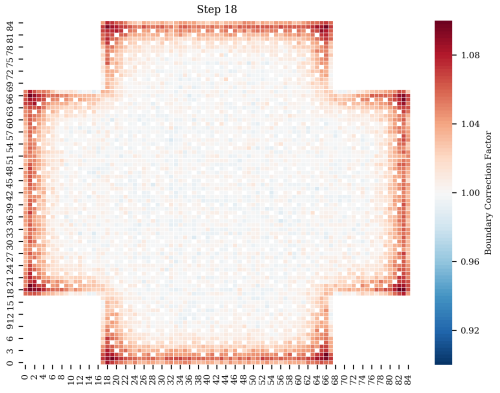
(o) $bnd(x, y, t)$ step 15



(p) $bnd(x, y, t)$ step 16

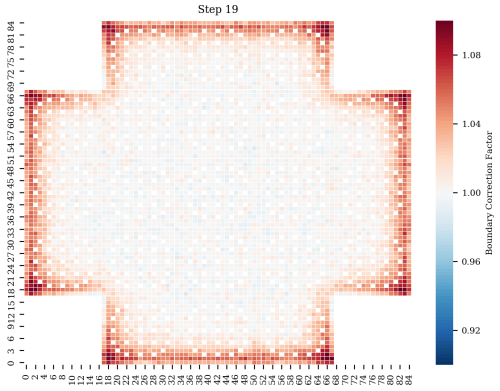


(q) $bnd(x, y, t)$ step 17



(r) $bnd(x, y, t)$ step 18

Figure 6.6 $bnd(x, y, t)$ boundary correction factors (cont.)



(s) $bnd(x, y, t)$ step 19

Figure 6.6 $bnd(x, y, t)$ boundary correction factors (cont.)

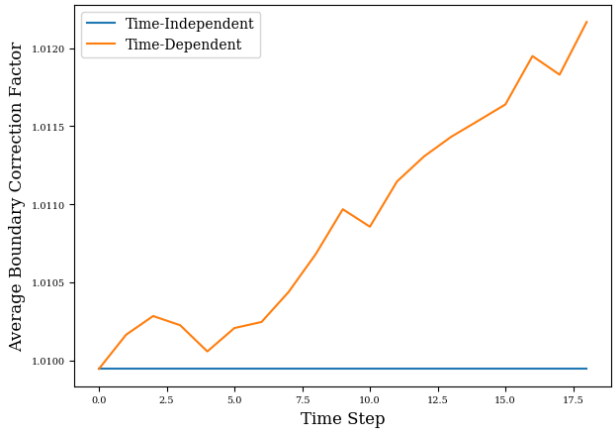


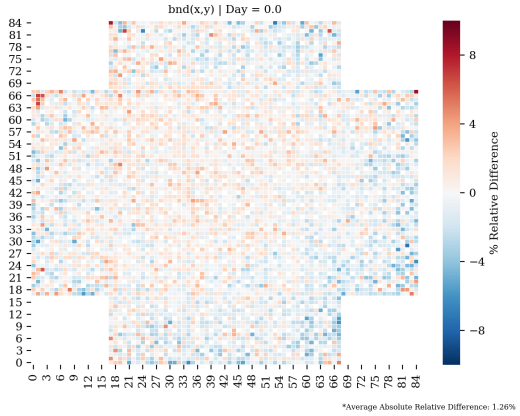
Figure 6.7 Average boundary correction factors for time-independent and time-dependent cases

6.2.1 Fission Source Distribution Comparison

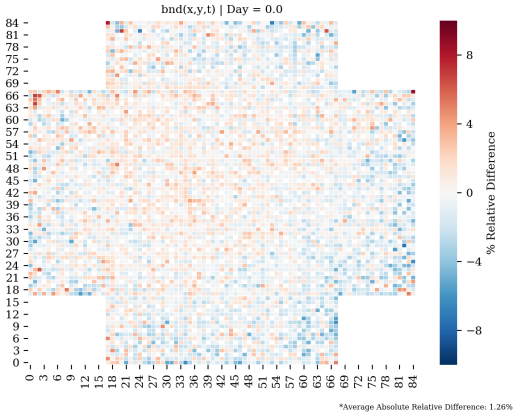
Figure 6.8 shows the fission source distribution relative difference plots for $bnd(x, y)$ and $bnd(x, y, t)$. Here, differences in the two methods can be seen developing as early as day 60, or step 9. The $bnd(x, y)$ figures begin showing increased relative differences along the perimeter regions, and to a lesser degree central regions, compared to $bnd(x, y, t)$. Figure 6.9 displays these differences clearly for assembly 19 (see Figure 6.2a), a corner assembly

bordered by water on the left and top sides. One can observe the relative difference of the border pins on the left and top fade significantly, from 4%-8% in some areas to 1%-2%. Improvements are also seen in the interior of the assembly to a lesser extent as well.

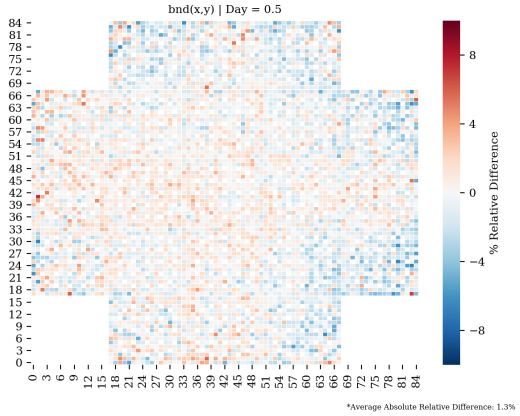
The effect of the $bnd(x, y, t)$ correction is to raise the fission source distribution in the perimeter regions, and decrease the fission source distribution in the center, thus providing a more accurate fission source distribution. These two effects (raising the perimeter, lowering the center) appear to roughly cancel each other out, which will be discussed in the next section. However, correcting the fission source distribution at each step does have value; downstream calculations such as assembly-wise burnup and pin-wise burnup are dependent on fission source distribution. In later chapters, the effect this method has on pin-wise burnup and isotopic composition will be investigated.



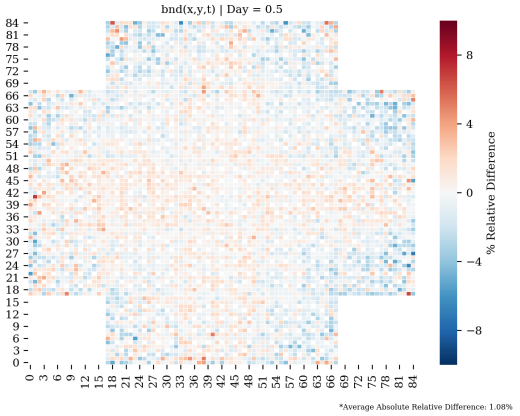
(a) $bnd(x, y)$ Day 0



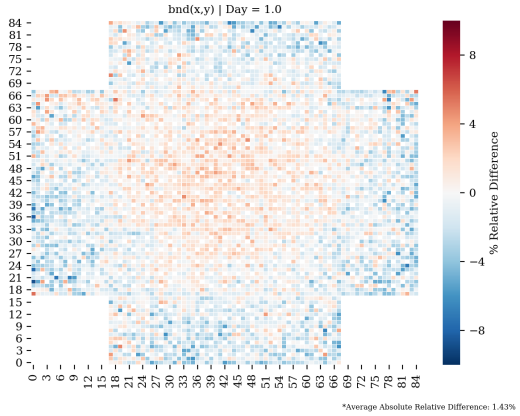
(b) $bnd(x, y, t)$ Day 0



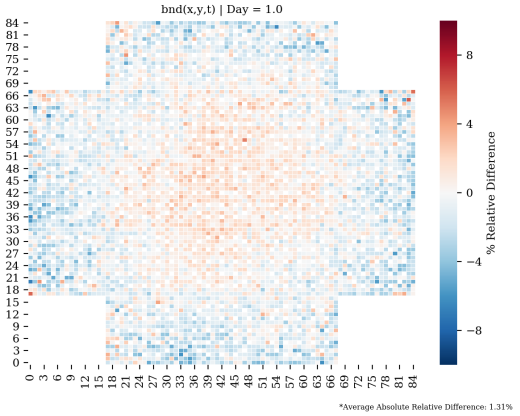
(c) $bnd(x, y)$ Day 0.5



(d) $bnd(x, y, t)$ Day 0.5

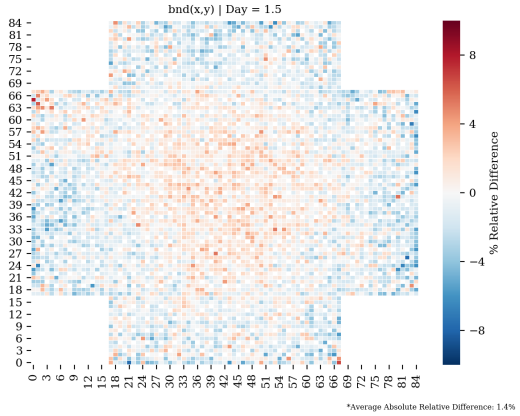


(e) $bnd(x, y)$ Day 1.0

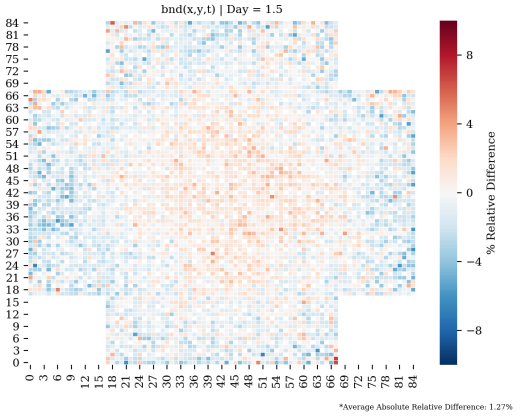


(f) $bnd(x, y, t)$ Day 1.0

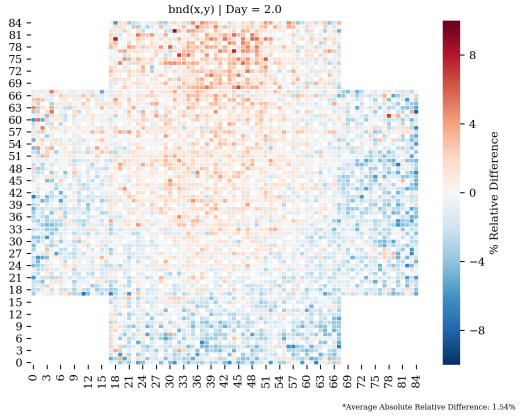
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$



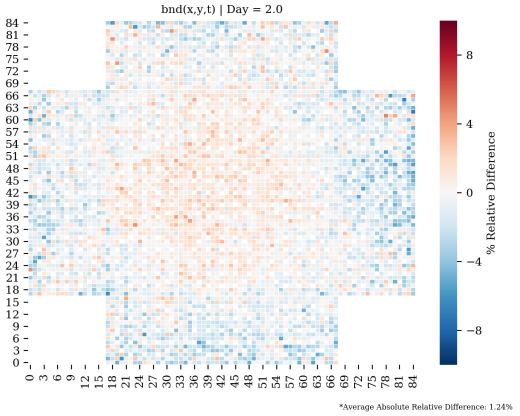
(g) $bnd(x, y)$ Day 1.5



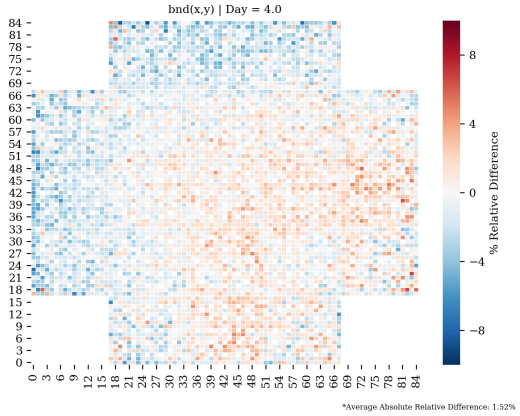
(h) $bnd(x, y, t)$ Day 1.5



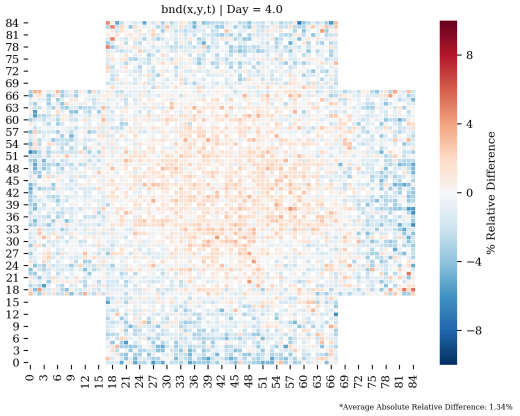
(i) $bnd(x, y)$ Day 2.0



(j) $bnd(x, y, t)$ Day 2.0

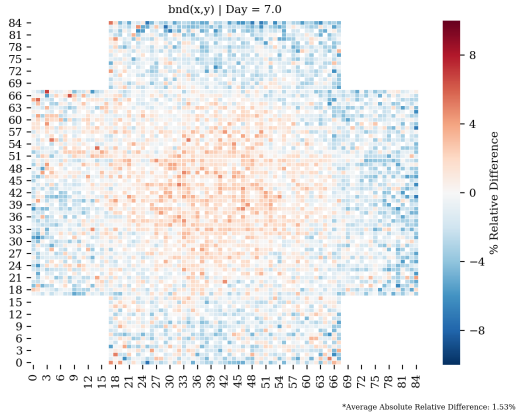


(k) $bnd(x, y)$ Day 4.0

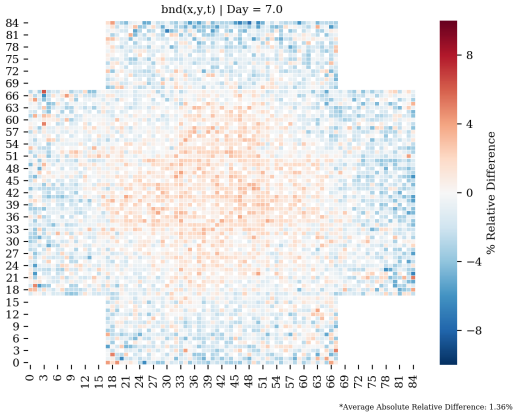


(l) $bnd(x, y, t)$ Day 4.0

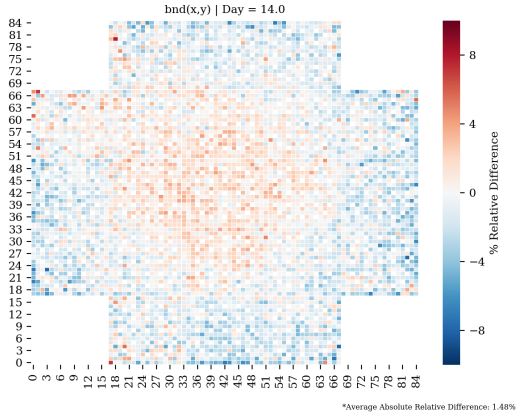
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



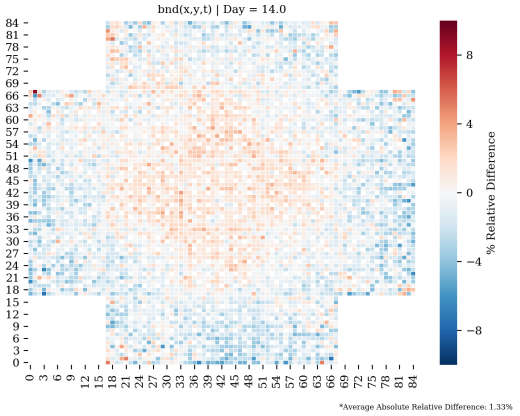
(m) $bnd(x, y)$ Day 7.0



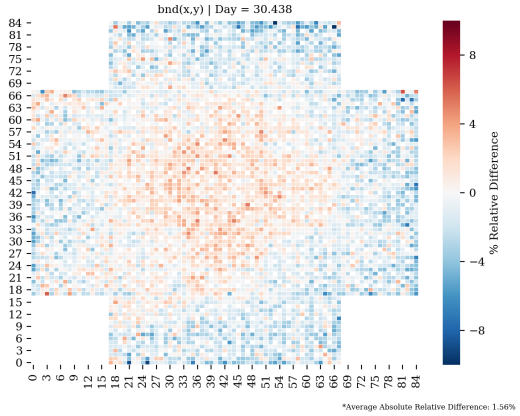
(n) $bnd(x, y, t)$ Day 7.0



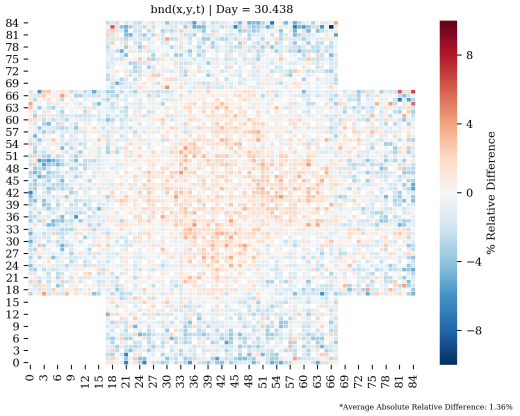
(o) $bnd(x, y)$ Day 14.0



(p) $bnd(x, y, t)$ Day 14.0

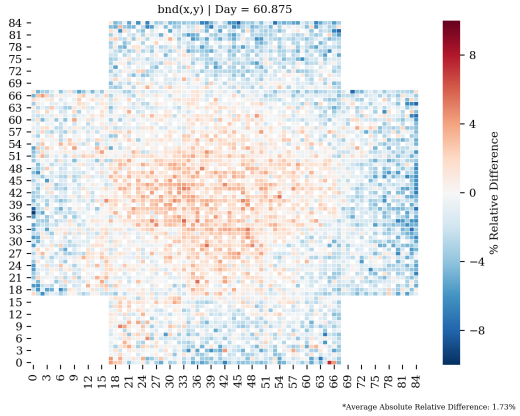


(q) $bnd(x, y)$ Day 30.438

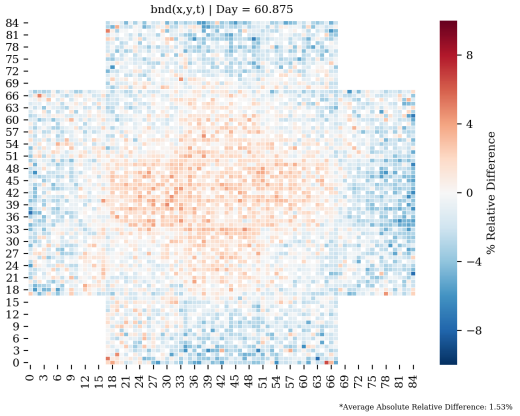


(r) $bnd(x, y, t)$ Day 30.438

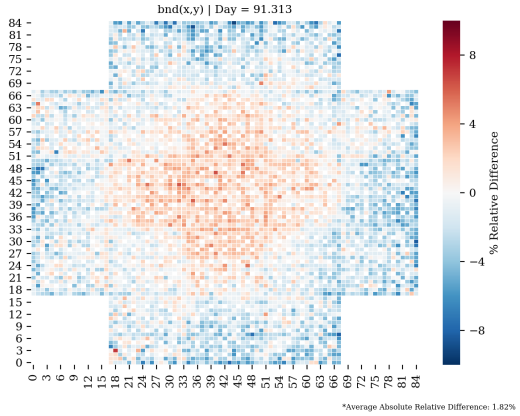
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



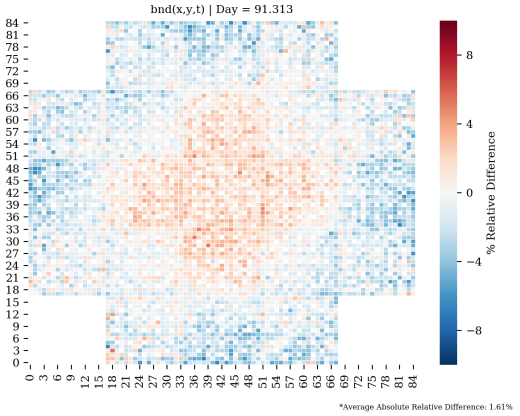
(s) $bnd(x, y)$ Day 60.875



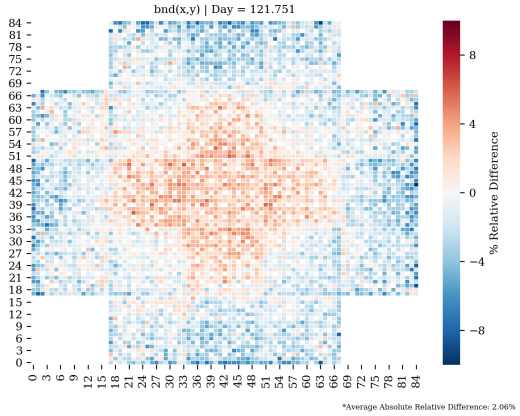
(t) $bnd(x, y, t)$ Day 60.875



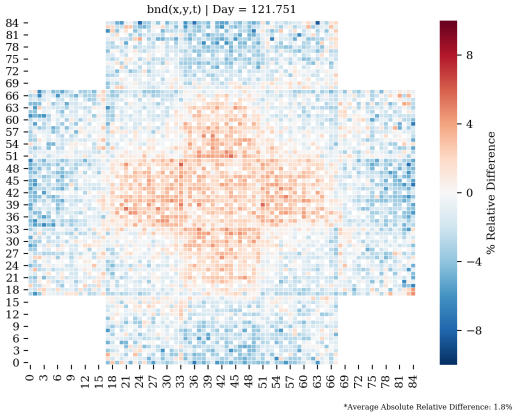
(u) $bnd(x, y)$ Day 91.313



(v) $bnd(x, y, t)$ Day 91.313

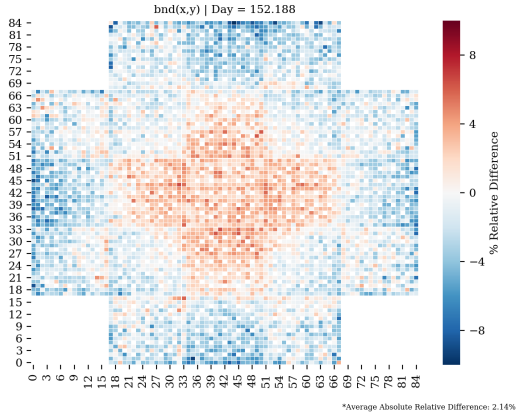


(w) $bnd(x, y)$ Day 121.751

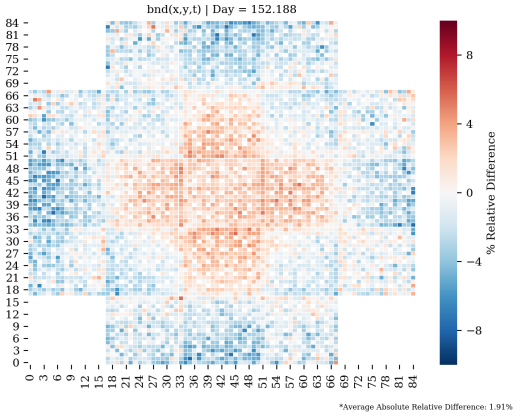


(x) $bnd(x, y, t)$ Day 121.751

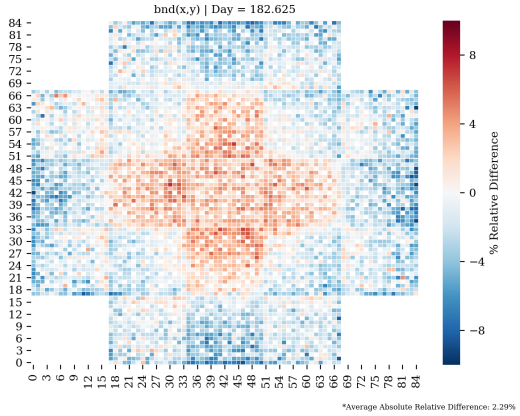
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



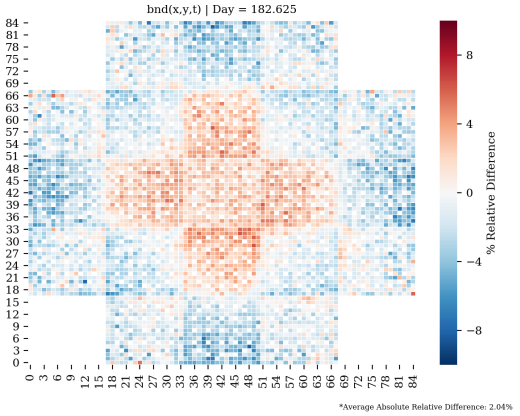
(y) $bnd(x, y)$ Day 152.188



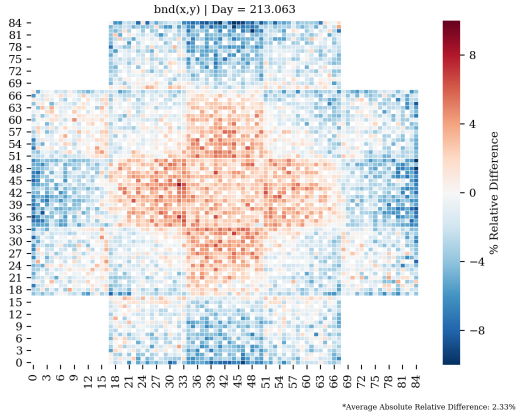
(z) $bnd(x, y, t)$ Day 152.188



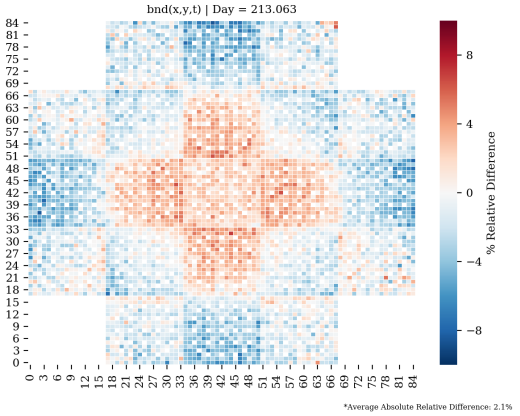
(aa) $bnd(x, y)$ Day 182.625



(ab) $bnd(x, y, t)$ Day 182.625

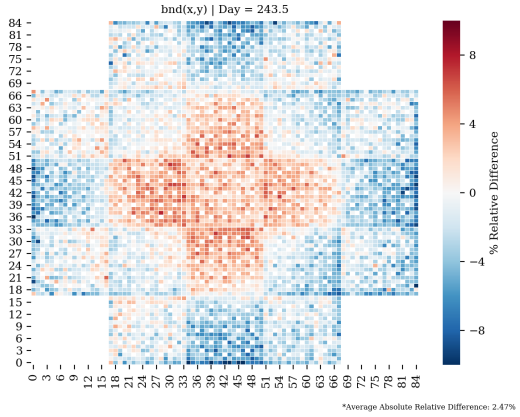


(ac) $bnd(x, y)$ Day 213.063

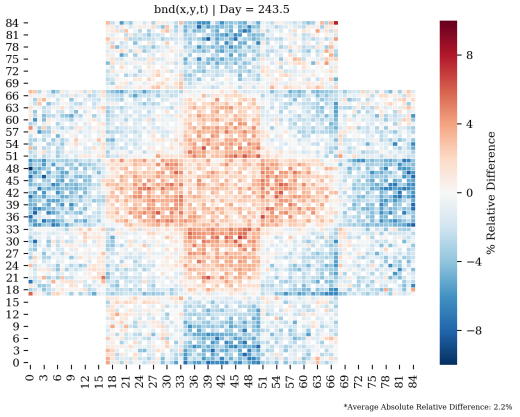


(ad) $bnd(x, y, t)$ Day 213.063

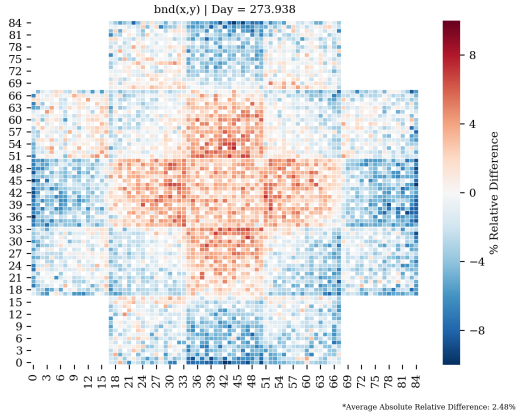
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



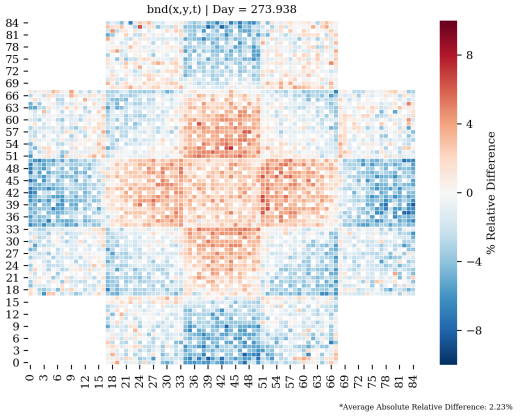
(ae) $bnd(x, y)$ Day 243.5



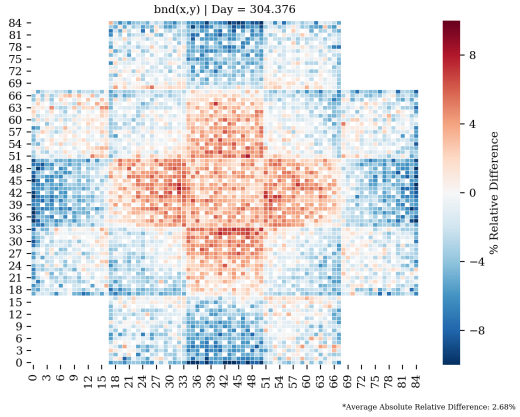
(af) $bnd(x, y, t)$ Day 243.5



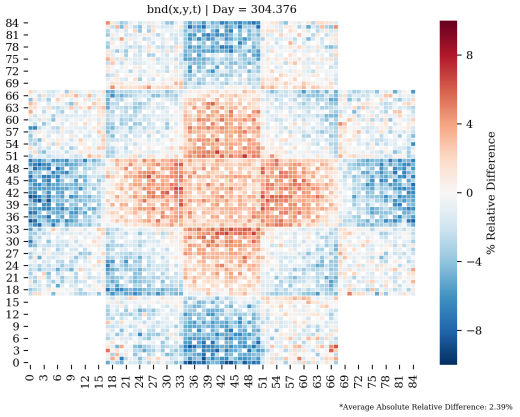
(ag) $bnd(x, y)$ Day 273.938



(ah) $bnd(x, y, t)$ Day 273.938

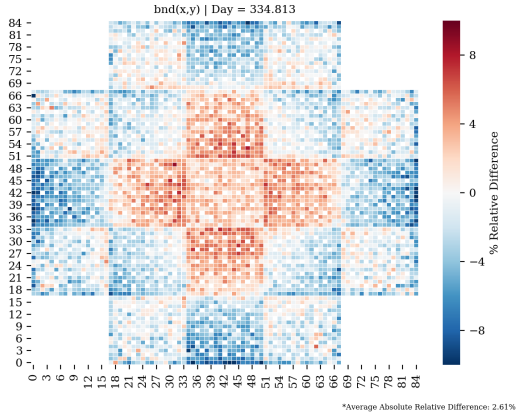


(ai) $bnd(x, y)$ Day 304.376

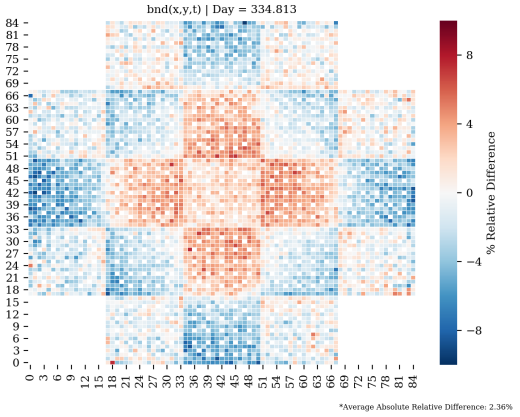


(aj) $bnd(x, y, t)$ Day 304.376

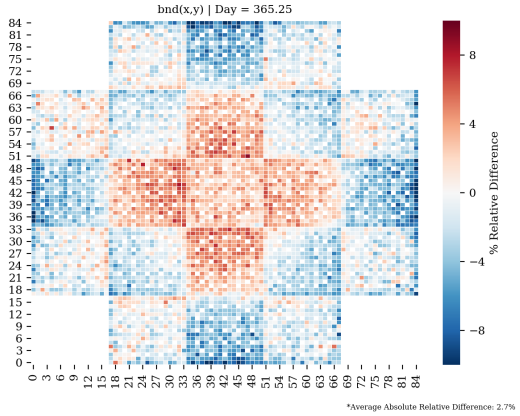
Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



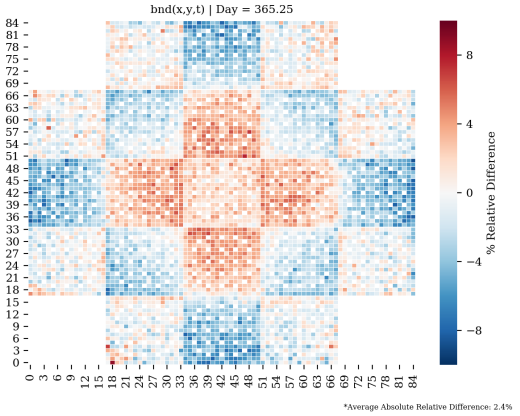
(ak) $bnd(x, y)$ Day 334.813



(al) $bnd(x, y, t)$ Day 334.813

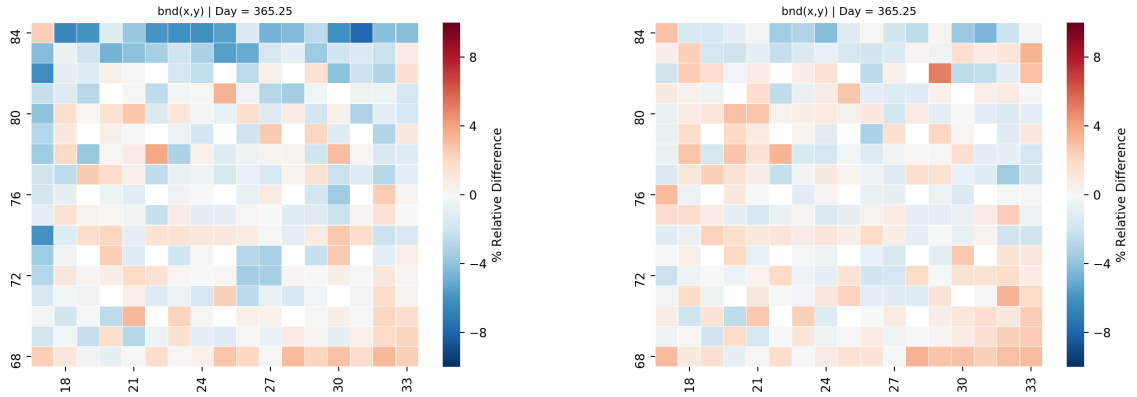


(am) $bnd(x, y)$ Day 365.25



(an) $bnd(x, y, t)$ Day 365.25

Figure 6.8 Fission source distribution relative differences for $bnd(x, y)$ and $bnd(x, y, t)$ (cont.)



(a) $bnd(x, y)$ Day 365.25 **(b)** $bnd(x, y, t)$ Day 365.25

Figure 6.9 Assembly 19 at day 365.25 shows differences between the $bnd(x, y)$ and $bnd(x, y, t)$ methodologies

The effect can be observed on a pin-row by pin-row basis as well. Figures 6.10 through 6.13 display the values for a single row of pins at a time by selecting a single "Y" index and parsing the fission source distribution for all "X" indices in that row. These figures show the fission source relative differences for row 0, 17, 67, and 84, in order to show the boundary correction effects on pins bordering water. A general trend of improvement in these border rows can be seen when the time-dependent correction is applied compared to the time-independent correction. Rows on the edge of the reactor see greater improvement than rows near the center. Quantitatively, the average relative difference improvement seen in these figures ranges from 0.4% to 2%.

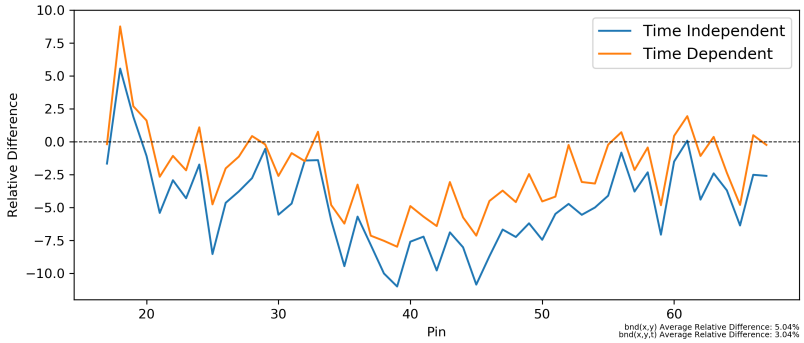


Figure 6.10 $bnd(x, y, t)$ fission source relative difference for row Y = 0

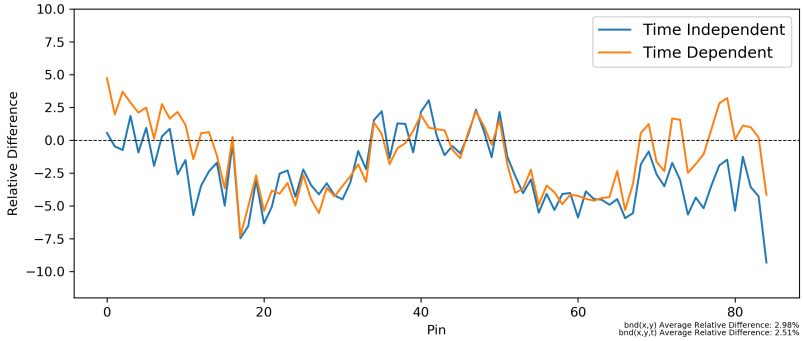


Figure 6.11 $bnd(x, y, t)$ fission source relative difference for row Y = 17

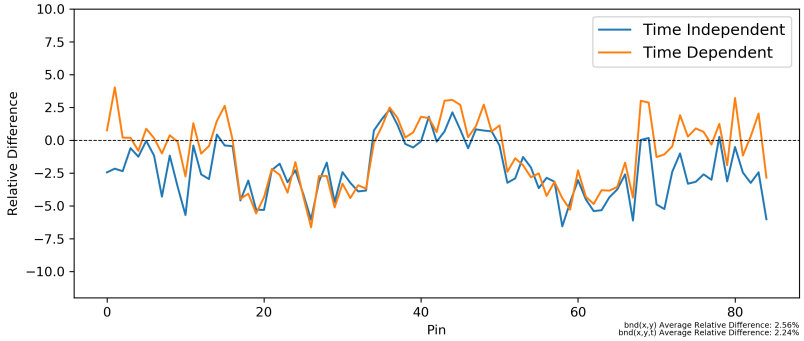


Figure 6.12 $bnd(x, y, t)$ fission source relative difference for row Y = 67

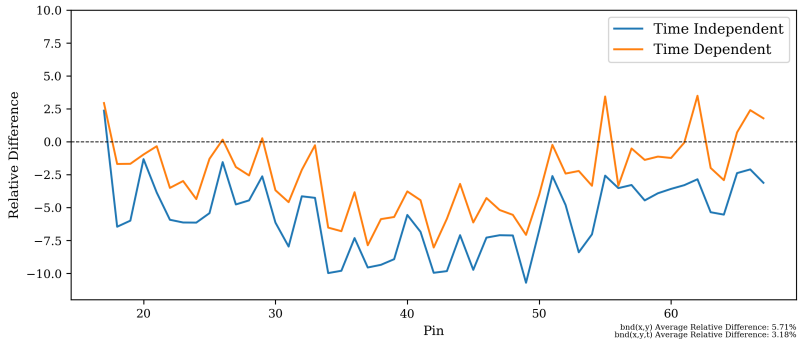


Figure 6.13 $bnd(x, y, t)$ fission source relative difference for row Y = 84

6.2.2 Eigenvalue Comparison

The eigenvalues for the three test runs are presented in Table 6.3. Figure 6.14 displays the same relative difference information as Table 6.3, step by step in order to help visualize the differences. Relative differences between bRAPID and the reference for k_{eff} are calculated using the following formula:

$$Rel.Diff. = 100 * \frac{k_{bRAPID} - k_{ref}}{k_{ref}} \tag{6.4}$$

Little difference between the k_{eff} values of each bRAPID case are seen; some steps using $bnd(x, y)$ are closer to the reference value, while other steps using $bnd(x, y, t)$ are closer to the reference value. These differences gradually increase as time steps progress. This phenomenon is to be expected– eigenvalues in RAPID calculations are directly correlated to the fission source distribution calculated in Section 6.2.1. Thus, just as fission source differences grow over time, so do k_{eff} differences.

As previously mentioned, the improvements in fission source distribution appear to roughly cancel out. k_{eff} is an integral quantity of the fission source distribution throughout the core. As the center of the fission source distribution is suppressed, the periphery is proportionally raised. This leads to the situation seen here– where amelioration of boundary physics

issues may not improve k_{eff} for some models, as the 'raising' effect at the boundary and 'suppressing' effect at the center roughly cancel each other out.

Table 6.3 Calculated eigenvalue comparison between Serpent reference and *b*RAPID ($bnd(x, y)$ and $bnd(x, y, t)$) and relative differences in pcm

Step No.	**Cum. Days**	**Serpent*** k_{eff}	***b*RAPID bnd(x,y)** k_{eff}	**Rel. Diff. (pcm)**	***b*RAPID bnd(x,y,t)** k_{eff}	**Rel. Diff. (pcm)**
0	0.0	1.13263	1.13126	-121	1.13126	-121
1	0.5	1.11814	1.11806	-7	1.11796	-16
2	1.0	1.10902	1.10772	-117	1.10774	-115
3	1.5	1.10565	1.10423	-128	1.10427	-125
4	2.0	1.10451	1.10363	-80	1.1033	-110
5	4.0	1.10273	1.10236	-34	1.1014	-121
6	7.0	1.10101	1.10015	-78	1.10004	-88
7	14.0	1.09762	1.09633	-118	1.09625	-125
8	30.438	1.09212	1.09116	-88	1.09093	-109
9	60.875	1.08466	1.08371	-88	1.08377	-82
10	91.313	1.07736	1.07673	-58	1.07688	-45
11	121.751	1.0701	1.06919	-85	1.06894	-108
12	152.188	1.0627	1.06162	-102	1.06162	-102
13	182.625	1.05538	1.054	-131	1.05404	-127
14	213.063	1.048	1.04651	-142	1.04668	-126
15	243.5	1.04113	1.03895	-209	1.03895	-209
16	273.938	1.03429	1.03163	-257	1.03165	-255
17	304.376	1.02731	1.02449	-275	1.02467	-257
18	334.813	1.0205	1.0177	-274	1.0177	-274
19	365.25	1.01411	1.01068	-338	1.01086	-320

* Serpent $\sigma = \pm$ 10 pcm

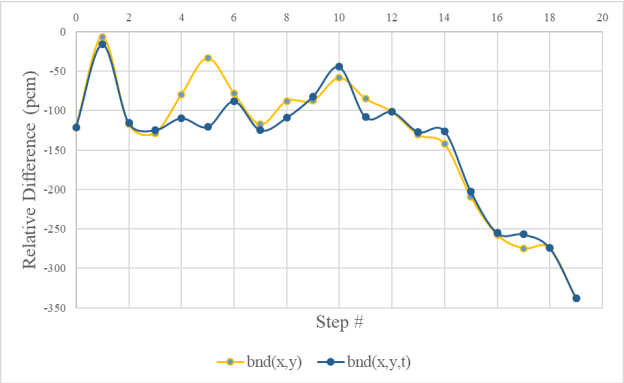


Figure 6.14 $bnd(x, y)$ and $bnd(x, y, t)$ relative difference comparison

6.2.3 Run Time Comparison

Table 6.4 contains run times for the three burnup runs performed in this chapter.

Compared to the Serpent reference calculation time, bRAPID yields a significant speedup– around 84 times faster using $bnd(x, y)$ and 86 times faster using $bnd(x, y, t)$. The variation between these bRAPID two run times is negligible– it is due to the computational load on the head node of the cluster at any given point in time. In practical terms, this means that one can re-run bRAPID in less than 12 minutes on a single core if changes are made to the model, while any changes require 16 hours on 8 processors to re-run using Serpent. While RAPID does not have parallel processing capability yet, in the future if this is implemented into RAPID, these bRAPID runs could speed up to a few minutes or even seconds.

Usage of the $bnd(x, y, t)$ methodology is useful insofar as it improves the fission source distribution, which will have an impact on down-stream calculations (such as pin-wise burnup), which will be addressed in the next Chapter. While improvements in the fission source distribution of pins may seem small (a few %), this improvement is important, as this distribution is the foundation of many of the analysis capabilities of RAPID. Simplified analyses simply look at the k_{eff} relative difference between models– but any detailed analysis requires that the 'shape' of your distributions are accurate. Due to the increase in accuracy with no in-

crease in *b*RAPID run time, use of the time-dependent boundary correction is recommended to achieve the most accurate *b*RAPID calculations.

Table 6.4 Calculation run time for Serpent and *b*RAPID runs. All times are presented in wall-clock time

	No. Processors	**Run Time**	**Speedup**
Serpent	8	16.86 hr	-
***b*RAPID with bnd(x,y)**	1	11.95 min	84.6
***b*RAPID with bnd(x,y,t)**	1	11.75 min	86.0

Chapter 7

Development of a 2D Pin-wise Burnup Algorithm for RAPID

*b*RAPID was designed to perform accurate k_{eff} and 2D/3D assembly-wise burnup calculations for core design, safeguards, and criticality safety analyses in real time [1]. It accomplishes this in part by utilizing the assembly-wise fission source distribution calculated by RAPID at each time step of a burnup calculation. While *b*RAPID does generate full 3D pin-wise fission source data, it does not use this data to calculate pin-wise burnup or material compositions. In this chapter, a new algorithm for *b*RAPID that calculates pin-wise burnup and material compositions in real time for 2D cases is introduced. Pin-wise burnup results are compared against a 5x5 Serpent reference model, and improvements made to this algorithm by using the time-dependent boundary correction algorithm are presented. For burnup and material composition comparisons shown in this Chapter, the time-dependent boundary correction algorithm was used.

7.1 Pin-wise Burnup Algorithm Methodology

The pin-wise burnup algorithm builds on top of the existing *b*RAPID structure and uses a methodology similar to the assembly-wise burnup algorithm. *b*RAPID first runs an initial RAPID run with inputs specified by the user, which solves the fission matrix equations discussed in previous chapters. This RAPID run will produce a number of output files, including the fission source distribution (F_i^n), which is used by *b*RAPID to define the power distribution input of the next step:

$$p_i^{n+1} = (\bar{P}^{n+1} N_b) f_i^n \tag{7.1}$$

where p_i^{n+1} is the power density distribution at index i for step $n+1$, N_b is the number of burnable regions, and $\bar{P}^{n+1}$ is the user-defined average power density.

While the user-defined core average power density remains the same for both the assembly-wise and the pin-wise burnup calculations, the number of burnable regions N_b changes from the number of fuel *assemblies* to the number of fuel *pins*. The f_i^n also changes from an assembly-wise quantity to pin-wise, described by:

$$f_i^n = \frac{F_i^n}{\sum_i F_i^n} \tag{7.2}$$

The general idea here is to use the 'shape' of the fission neutron density to produce a pin-wise power density distribution that can be used to calculate burnup and material compositions.

Figure 7.1 is a flowchart of *b*RAPID. The light blue boxes represent the changes made to the *b*RAPID algorithm for pin-wise burnup. One can see that the general operation of *b*RAPID remains unchanged, however, the changes made require additional calculations, and create additional outputs as a result. The primary output for the pin-wise burnup algorithm is `mats_pinwise.out`, which uses the same formatting scheme as the traditional `mats.out` file

containing assembly-wise burnup data.

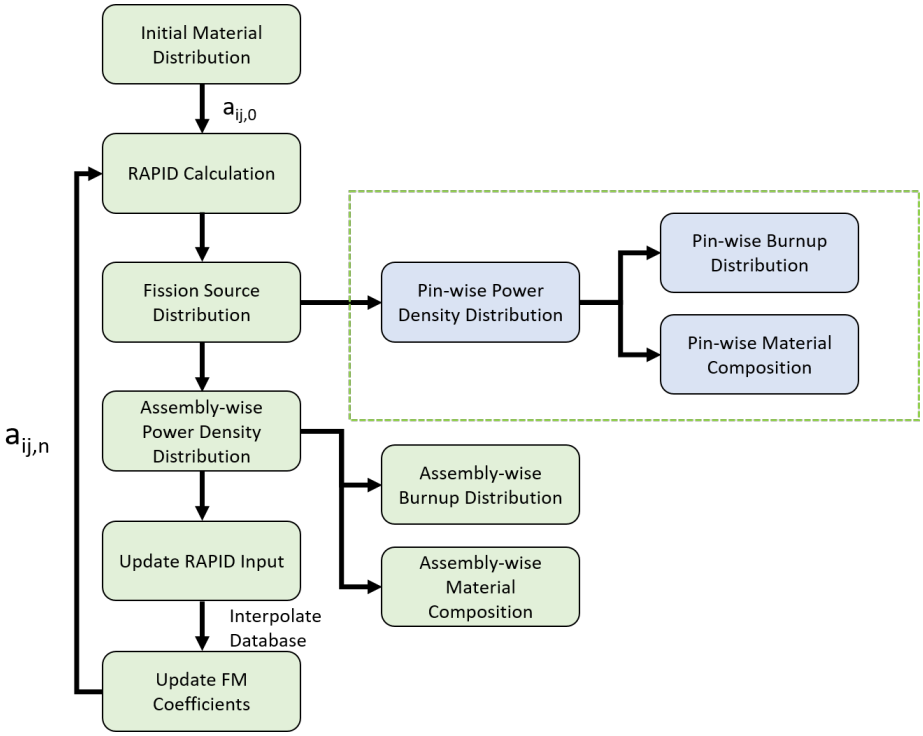


Figure 7.1 Flowchart of the additions made to the *b*RAPID algorithm to allow for pin-wise burnup calculations

The pin-wise algorithm is a 'downstream' calculation; the underlying RAPID calculation is unchanged and still performed using axially dependent assembly-wise FM coefficients. An even more detailed and accurate RAPID pin-wise burnup algorithm could be developed with significant changes to the existing RAPID code, however these changes are beyond the scope of this thesis. The pin-wise burnup approach presented here makes efficient use of the data already generated by RAPID to retrieve even more information about the system.

One major limitation when performing pin-wise material composition calculations is the memory and storage usage required. For example, the 2D 5x5 'mini-core' Serpent reference calculation, when tracking every nuclide and assigning every pin for burnup, produced a 15 GB depletion output file. This can be lowered by extracting fewer nuclides (currently less than 200 important isotopes; see Appendix B.1 for list) and by using model symmetries. The RAPID pin-wise material output file was 330 MB by comparison. Still, one can imagine

the large memory and storage requirements if one were to increase the system size to a 3D 'full-core' model. When performing a pin-wise simulation with this algorithm, regional pin-wise information can be extracted at any time needed by storing the power density distribution. Further, extracting only those isotopes of interest to fuel burnup would reduce storage requirements as well.

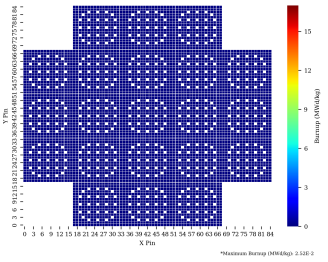
7.2 Pin-wise Burnup Comparison

In this section, results of the pin-wise burnup algorithm are compared with the Serpent reference model by analyzing pin-wise distributions and relative differences. The same time step scheme described in Table 6.2, the same database configuration described in Chapter 6, and the same Monte Carlo parameters chosen in Chapter 4 are used; however, the Serpent reference model has been modified to specify each fuel pin as a burnable region. In addition, the time-dependent boundary correction algorithm from Chapter 6 will be used in this section in order to provide more accurate results.

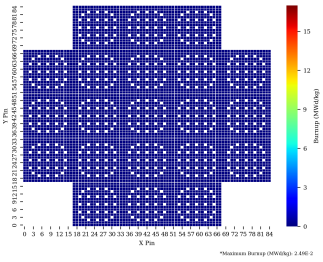
Figure 7.2 shows the pin-wise burnup for both *b*RAPID and Serpent for all time steps, as well as the relative differences between those two distributions. The relative difference between these two distributions begins at a low level, averaging less than 2% until around day 60. At this point, a pattern of *b*RAPID overestimation in the center, and underestimation in the outer region emerges. As time goes on, these differences become more pronounced until the final burnup step is reached. The relative difference values at this point are almost all within $\pm$ 10%, with an average relative difference of 3.73%. The maximum burnup value for Serpent is 16.5 MWd/kg, and for *b*RAPID it is 17.4 MWd/kg.

The behavior of the burnup distribution relative differences is similar to that of the fission source distribution relative differences seen in Chapter 6, as expected, since the pin-wise burnup is derived from the fission source distribution at each step. In general, RAPID displays a sharper fission source and burnup drop-off compared to Serpent which has a

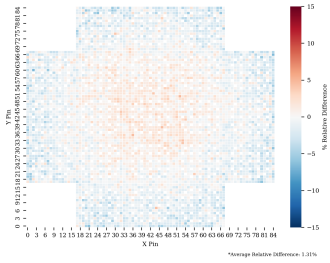
flatter distribution. Figure 7.3 illustrates this clearly; in the center of the core the RAPID burnup value is higher than Serpent, but as one moves towards the edge of the core, the RAPID value crosses the Serpent value and ends up slightly underestimating the burnup near the boundary.



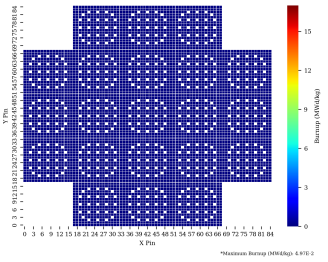
(a) *b*RAPID Day 0.5



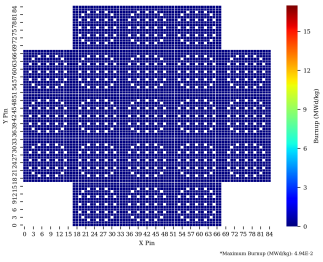
(b) Serpent Day 0.5



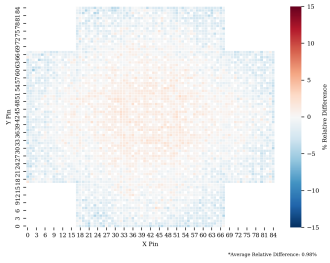
(c) Relative Difference Day 0.5



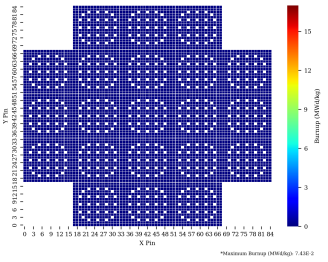
(d) *b*RAPID Day 1.0



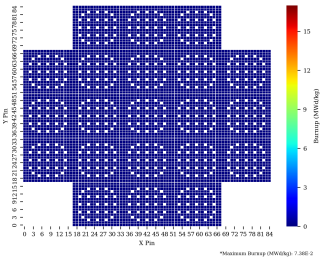
(e) Serpent Day 1.0



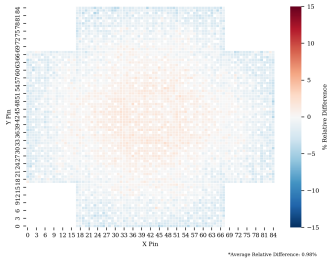
(f) Relative Difference Day 1.0



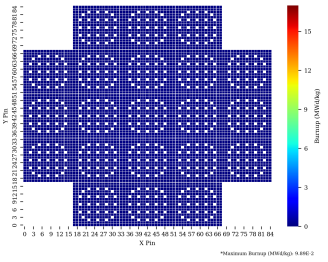
(g) *b*RAPID Day 1.5



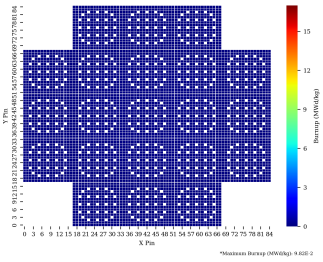
(h) Serpent Day 1.5



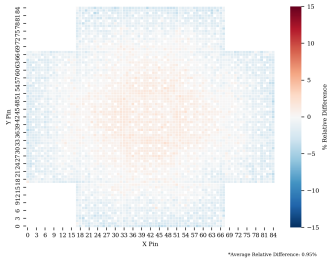
(i) Relative Difference Day 1.5



(j) *b*RAPID Day 2.0

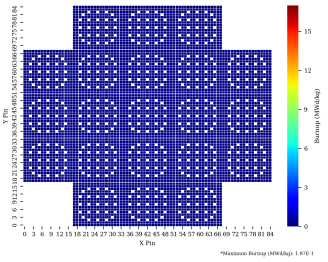


(k) Serpent Day 2.0

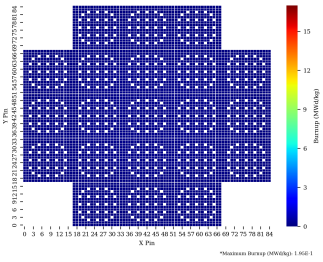


(l) Relative Difference Day 2.0

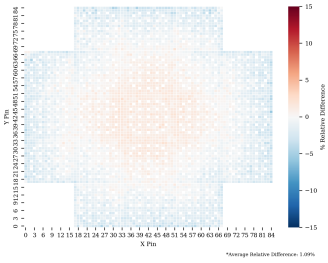
Figure 7.2 *b*RAPID and Serpent pin-wise burnup distribution and relative differences



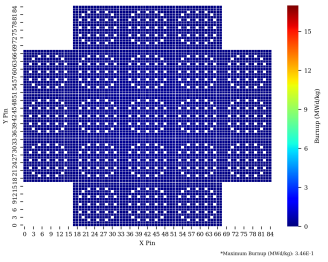
(m) *b*RAPID Day 4.0



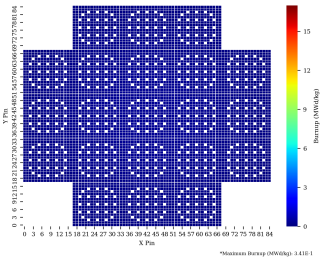
(n) Serpent Day 4.0



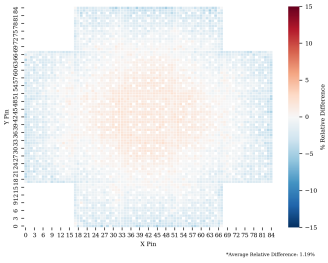
(o) Relative Difference Day 4.0



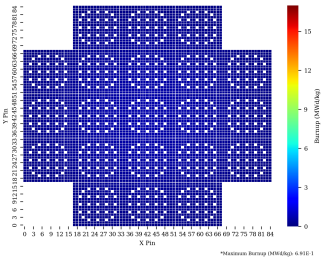
(p) *b*RAPID Day 7.0



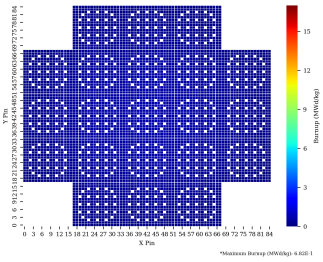
(q) Serpent Day 7.0



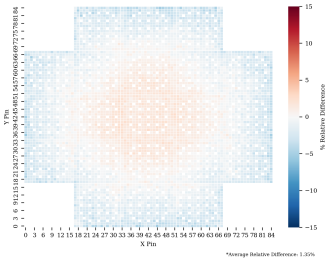
(r) Relative Difference Day 7.0



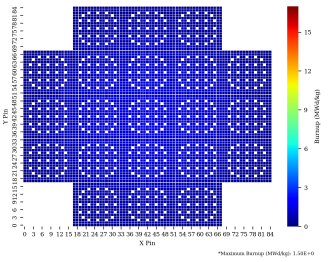
(s) *b*RAPID Day 14.0



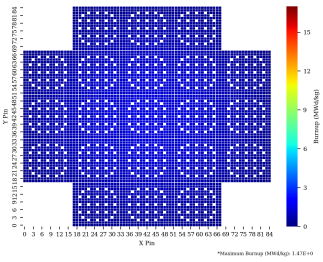
(t) Serpent Day 14.0



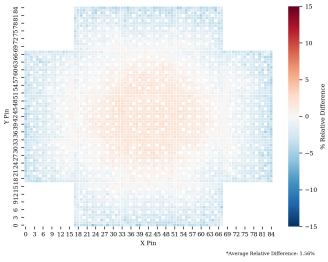
(u) Relative Difference Day 14.0



(v) *b*RAPID Day 30.438

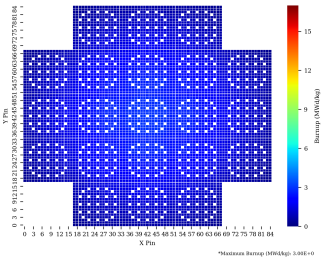


(w) Serpent Day 30.438

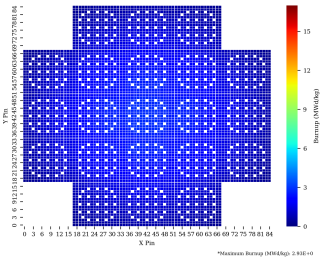


(x) Relative Difference Day 30.438

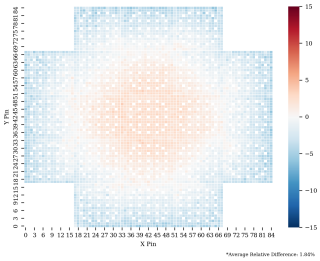
Figure 7.2 *b*RAPID and Serpent pin-wise burnup distribution and relative differences (cont.)



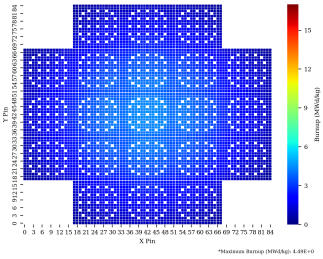
(y) *b*RAPID Day 60.875



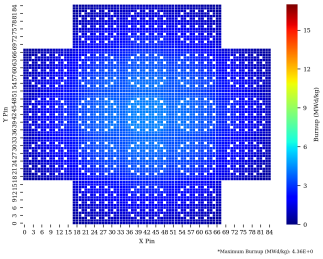
(z) Serpent Day 60.875



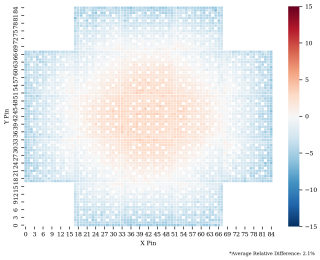
(aa) Relative Difference Day 60.875



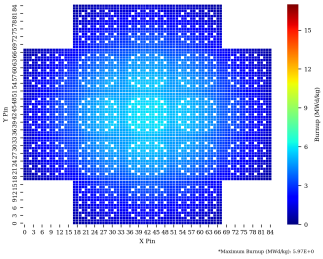
(ab) *b*RAPID Day 91.313



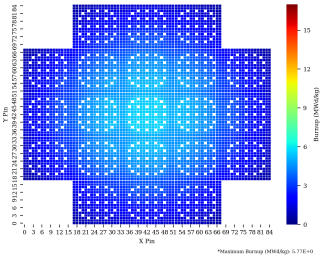
(ac) Serpent Day 91.313



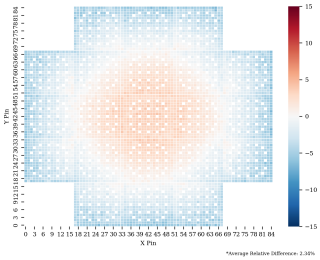
(ad) Relative Difference Day 91.313



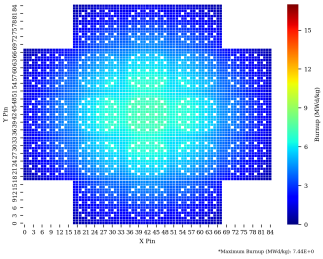
(ae) *b*RAPID Day 121.751



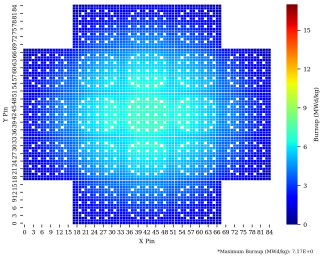
(af) Serpent Day 121.751



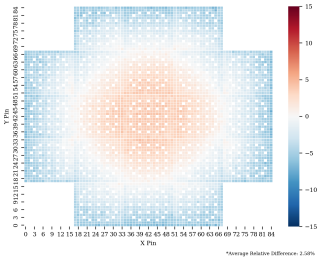
(ag) Relative Difference Day 121.751



(ah) *b*RAPID Day 152.188

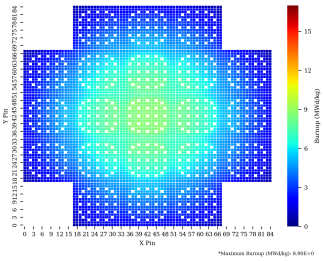


(ai) Serpent Day 152.188

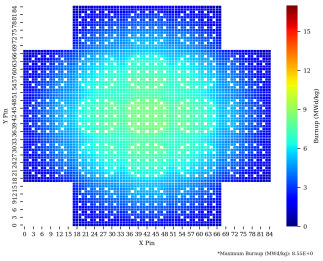


(aj) Relative Difference Day 152.188

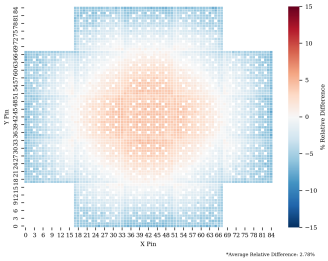
Figure 7.2 *b*RAPID and Serpent pin-wise burnup distribution and relative differences (cont.)



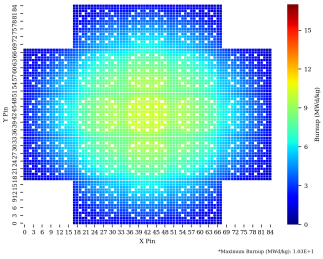
(ak) *b*RAPID Day 182.625



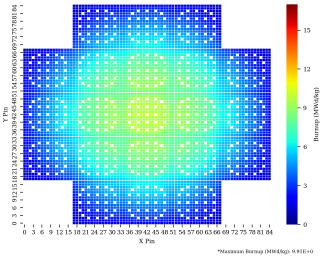
(al) Serpent Day 182.625



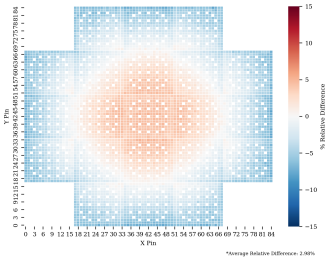
(am) Relative Difference Day 182.625



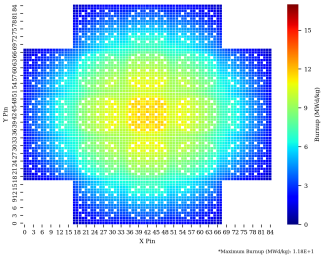
(an) *b*RAPID Day 213.063



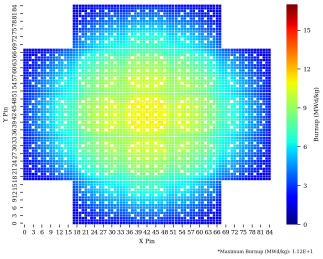
(ao) Serpent Day 213.063



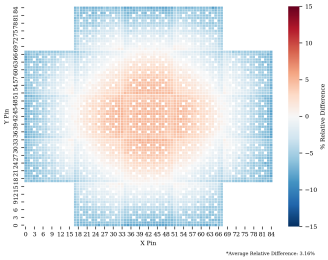
(ap) Relative Difference Day 213.063



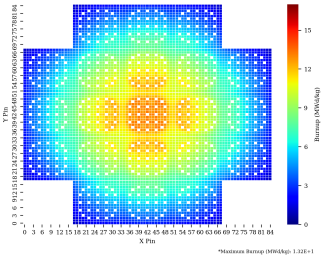
(aq) *b*RAPID Day 243.5



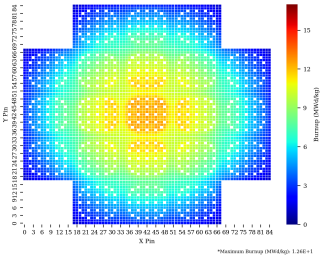
(ar) Serpent Day 243.5



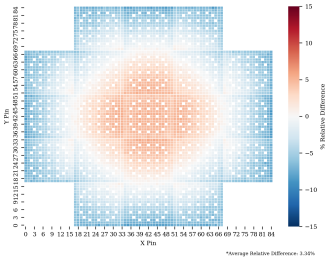
(as) Relative Difference Day 243.5



(at) *b*RAPID Day 273.938

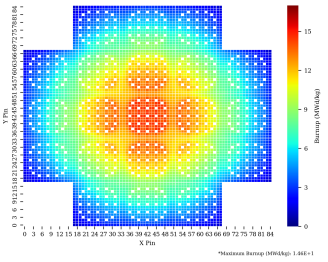


(au) Serpent Day 273.938

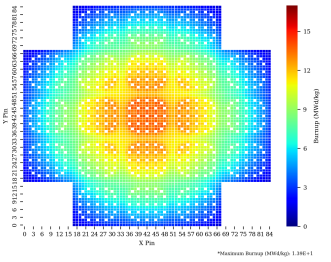


(av) Relative Difference Day 273.938

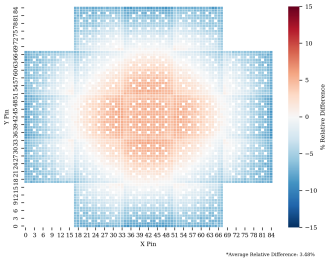
Figure 7.2 *b*RAPID and Serpent pin-wise burnup distribution and relative differences (cont.)



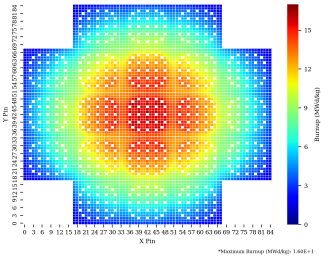
(aw) *b*RAPID Day 304.376



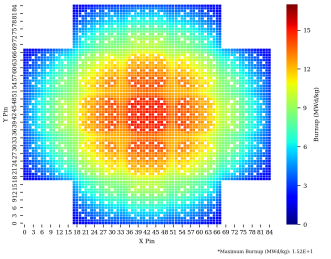
(ax) Serpent Day 304.376



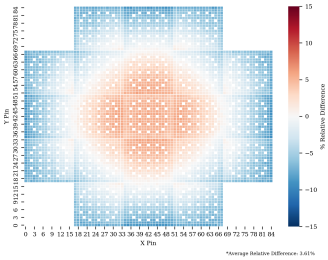
(ay) Relative Difference Day 304.376



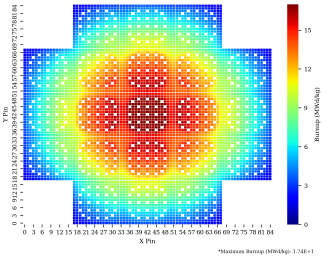
(az) *b*RAPID Day 334.813



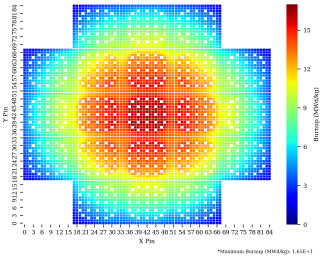
(ba) Serpent Day 334.813



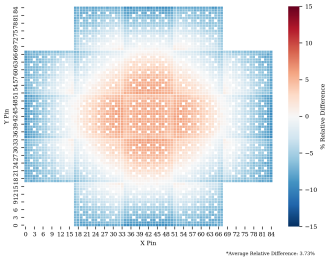
(bb) Relative Difference Day 334.813



(bc) *b*RAPID Day 365.25



(bd) Serpent Day 365.25



(be) Relative Difference Day 365.25

Figure 7.2 *b*RAPID and Serpent pin-wise burnup distribution and relative differences (cont.)

The pin-wise burnup routine can be seen as an extension of the assembly-wise burnup routine; the underlying calculation for both processes is the same. The pin-wise routine effectively 'unpacks' a greater amount of information from the same RAPID fission source calculation. This is readily apparent when the two calculation outputs are compared side by side. Figure 7.4 shows this comparison for a single burnup step. One can see that the overall distribution appears the same, but the pin-wise routine produces a distribution with greater granularity.

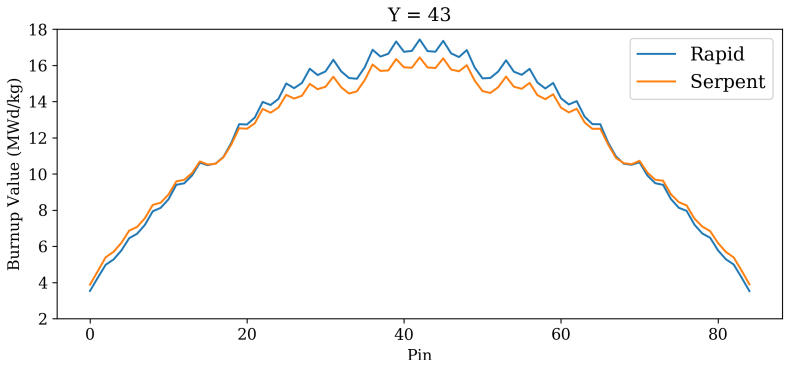
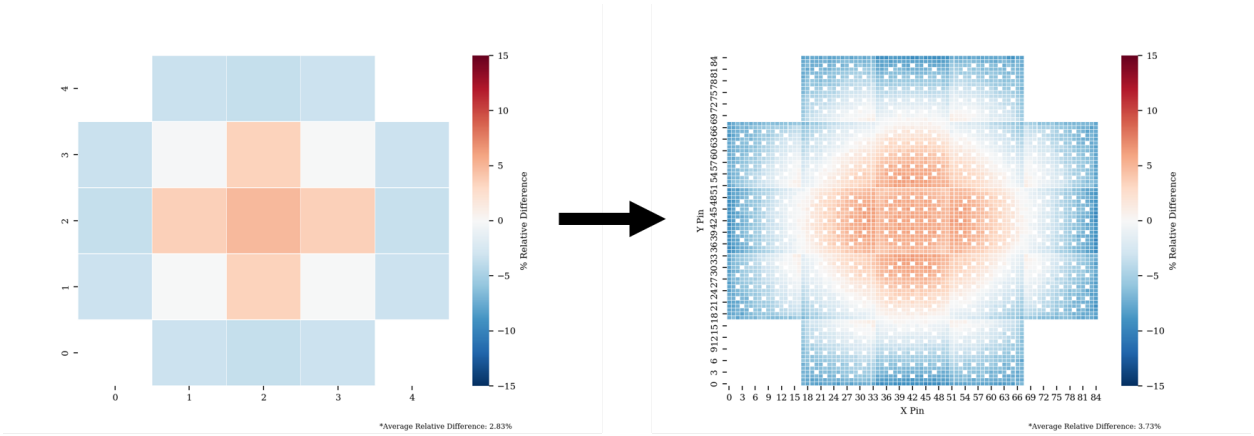


Figure 7.3 Burnup distribution for a single row of pins (Y = 43) at final burnup step for *b*RAPID and Serpent



Assembly-wise Burnup Pin-wise Burnup

Figure 7.4 Assembly-wise and Pin-wise burnup calculations compared side by side. The figures above are relative difference plots compared to Serpent for a single burnup step

7.3 Pin-wise Material Composition

As time passes during reactor operation, the material composition of the fuel rods change due to processes such as decay, fission, capture and others. The purpose of burnup simulations is not just to measure the amount of energy released by a fuel region, but also to determine how material compositions change over time, and particularly to determine the spacial distribution of those material compositions. Previously, *b*RAPID was capable of calculating

assembly-wise material compositions over time. By converting the pin-wise fission source distribution to a pin-wise power density distribution at each step, the algorithm was able to calculate pin-wise burnup over time. Now, the pin-wise power density is used to produce pin-wise material compositions by employing the two-factor interpolation scheme described in Chapter 3.

Recall that *b*RAPID databases are created as a function of power density and irradiation time. At each time step, the power density (p_x) is calculated for a given irradiation time (t_x) for each pin in a core. Now, apply the same logic used to interpolate assembly-wise material compositions to pins:

1. Find p_{high} and p_{low} in the database such that $p_{low} < p_x < p_{high}$
2. Find t_{high} and t_{low} in the database such that $t_{low} < t_x < t_{high}$
3. Calculate weighting coefficients w_i based on Equations 3.20 – 3.23
4. Use weighting coefficients to calculate m_x, the isotopic composition associated with p_x and t_x

Where m_x is the atom density of each isotope specified given in units of $b^{-1}cm^{-1}$.

Here the isotopic composition and relative differences for Xenon 135 are presented. Due to spacing constraints, other important isotope composition and relative difference plots can be found in Appendix B. There are no material composition uncertainties calculated either by Serpent or RAPID– thus evaluation of the relative differences must be performed without knowledge of the variance of the underlying quantity.

In Figure 7.5 one can see that ^{135}Xe quickly reaches an equilibrium condition within 4 days, and stays relatively constant thereafter. One can see that, prior to equilibrium, *b*RAPID actually overestimates the isotopic composition in the peripheral regions, as seen in Figure 7.6, but then begins to underestimate once equilibrium is reached. Despite ^{135}Xe being a difficult isotope to track due to its low concentrations and short half-life, the pin-wise

algorithm captures its composition well in the main portion of the reactor– most pins show a relative difference of ± 5%, and the average relative difference is 5.61% at the final burnup step. Figure 7.7 shows a single row cross section of pins at Y = 43, which illustrate the differences between *b*RAPID and Serpent. Here, one can see the extent to which *b*RAPID overestimates burnup in the central pins and underestimates along the perimeter of the core.

Similar results can be seen for ^{239}Pu in Figure B.1, only on a longer time scale. ^{239}Pu is overestimated at the boundary for approximately 180 days, during the period of growth. ^{239}Pu is underestimated at the boundary after 180 days. Again, most pins show a relative difference of ± 5%, except for at the boundary where there are large relative differences. This pattern holds true for ^{149}Sm as well, as seen by Figure B.2, however, ^{149}Sm appears to have a "flatter" distribution throughout the core.

The ^{131}Xe distribution in Figure B.3 shows interesting behavior. It first starts out overestimated at the boundary, like ^{135}Xe. However, over time the relative differences become smaller throughout the entire core, until finally the distributions are fairly similar. ^{131}Xe displays very low relative differences at this final burnup step, and the distribution appears similar to the overall burnup distribution.

A general trend of underestimation in edge pins is observed. While the absolute value of these relative differences may seem large in the periphery regions– it is important to evaluate the actual impact this underestimation has. Being on the periphery, these pins generally contain a small percentage of the total mass of an isotope in a given system. In other words– the underestimation may be large (> 20%), but the total mass of the isotope underestimated may be small. To investigate this, the "percent-mass of underestimated pins" (PUP) for each isotope can be calculated. PUP is given by the expression:

$$\text{PUP} = 100 \times \frac{m_{under}}{m_{total}} \tag{7.3}$$

where m_{under} is the total mass of a given isotope in underestimated pins, and m_{total} is the total mass of a given isotope in the system. An "underestimated" pin, for this metric,

is any pin that exhibits >-20% underestimation in isotopic composition with respect to the Serpent reference model. These pins were evaluated for the final burnup step (where peripheral underestimation is highest) using the Serpent reference model.

Table 7.1 displays these values for important isotopes which exhibit underestimation behavior in edge pins. One can see that, although the underestimation is high, the amount of isotopic mass that is actually affected by underestimation is quite low. ^{149}Sm has the highest percentage of mass underestimated in the system, but even this only reaches 4.15% of the total system mass of ^{149}Sm.

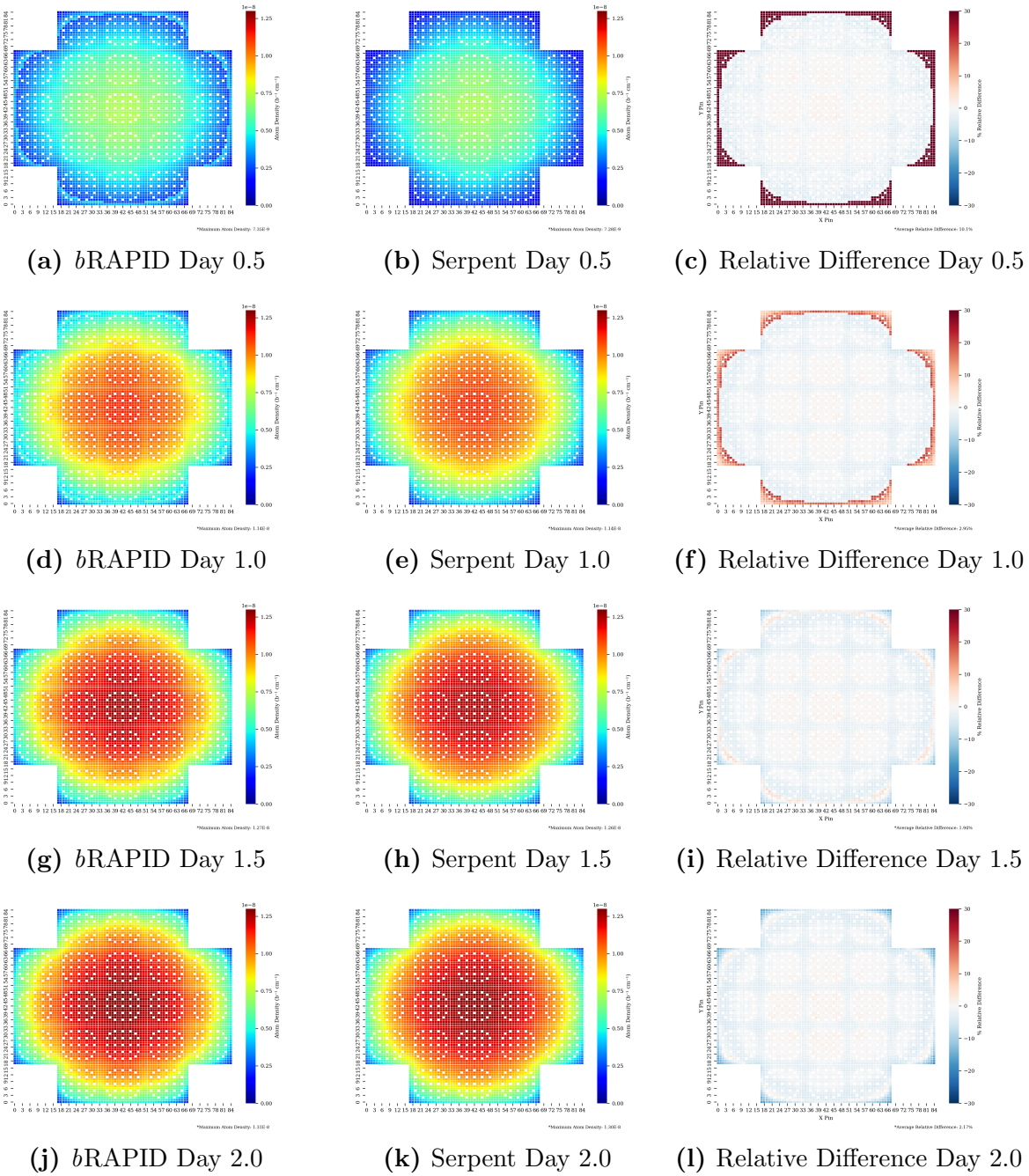
Table 7.1 Percent-mass of Underestimated Pins (PUP) for important isotopes

Isotope	**PUP** (%)
^{135}Xe	1.19
^{149}Sm	4.15
^{131}Xe	0.0
^{143}Nd	0.0
^{239}Pu	0.49

There are two likely causes for the large underestimation on the edge pins: power density underestimation and time step resolution in the database. Pin-wise power density appears to be underestimated at the edge of the reactor, due to the fission source distribution from which it is derived being underestimated. This underestimation is carried through to the interpolation scheme, where the material compositions are collected. However, this underestimation alone likely does not explain the entire difference between the *b*RAPID and Serpent ^{135}Xe compositions, since the relative difference of ^{135}Xe composition is significantly larger than that of burnup or fission source distribution. That leaves the database material composition values as the next most likely cause of the underestimation. If a database has too few time steps in the critical areas of growth for certain isotopes, it can lead to less accurate results. While the time step scheme used for this database is somewhat optimized to capture important isotope evolution especially in early days, more time steps would likely lead to

more accurate results during material interpolation, but would lead to longer computation times.

Additionally, without access to material composition uncertainty values, it is difficult to assess the degree to which differences are due to uncertainties and which are due to methodological differences. Numerics may also be causing issues; larger relative differences are seen in isotopes like ^{135}Xe, which have relatively low atom density (maximum value $\approx$ 1.25e-8 b^{-1} cm^{-1}), and smaller relative differences in isotopes like ^{131}Xe once they have reached a fairly high atom density (maximum value $\approx$ 1e-5 b^{-1} cm^{-1})



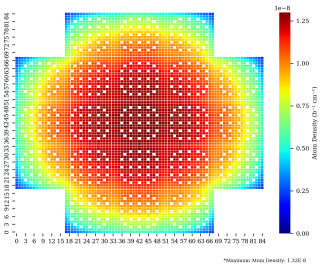
(a) *b*RAPID Day 0.5 **(b)** Serpent Day 0.5 **(c)** Relative Difference Day 0.5

(d) *b*RAPID Day 1.0 **(e)** Serpent Day 1.0 **(f)** Relative Difference Day 1.0

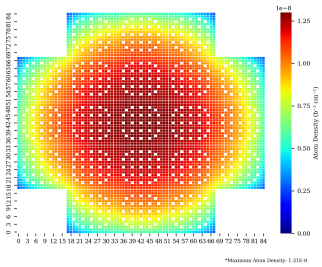
(g) *b*RAPID Day 1.5 **(h)** Serpent Day 1.5 **(i)** Relative Difference Day 1.5

(j) *b*RAPID Day 2.0 **(k)** Serpent Day 2.0 **(l)** Relative Difference Day 2.0

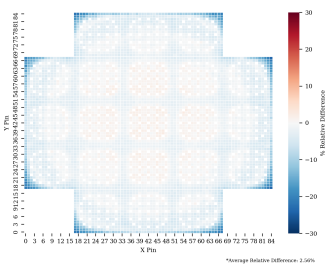
Figure 7.5 *b*RAPID and Serpent pin-wise ^{135}Xe distribution and relative differences



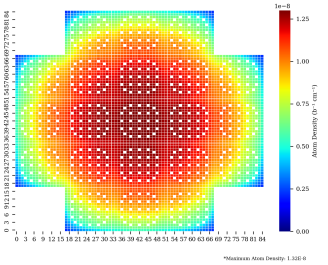
(m) *b*RAPID Day 4.0



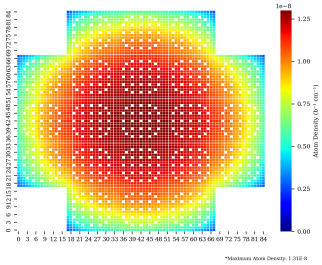
(n) Serpent Day 4.0



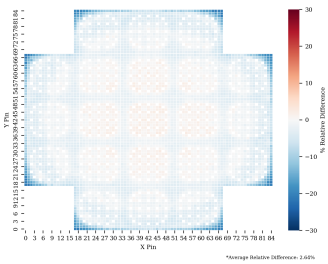
(o) Relative Difference Day 4.0



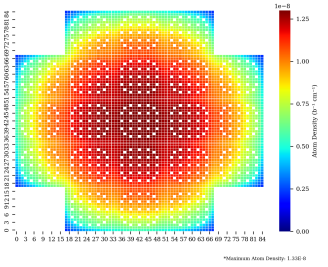
(p) *b*RAPID Day 7.0



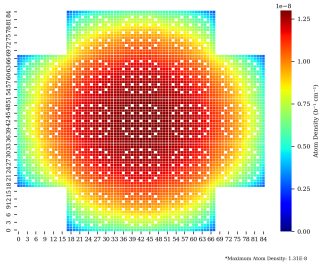
(q) Serpent Day 7.0



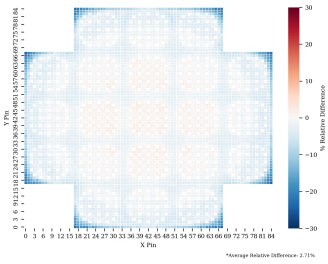
(r) Relative Difference Day 7.0



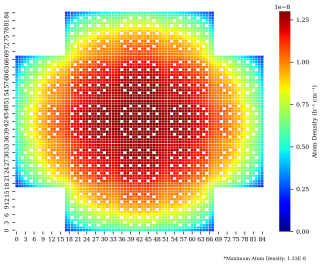
(s) *b*RAPID Day 14.0



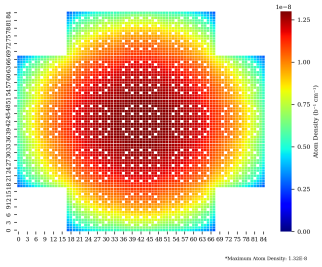
(t) Serpent Day 14.0



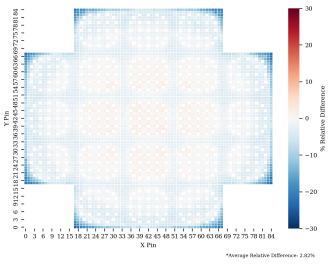
(u) Relative Difference Day 14.0



(v) *b*RAPID Day 30.438

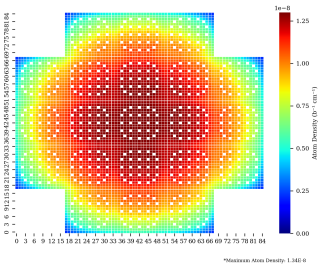


(w) Serpent Day 30.438

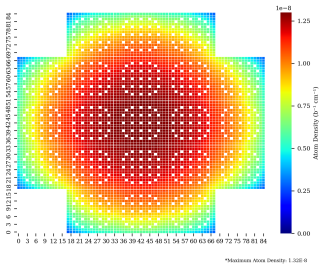


(x) Relative Difference Day 30.438

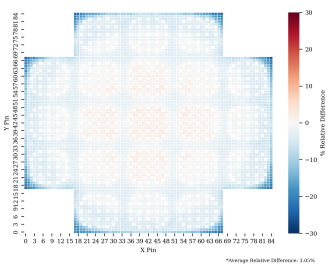
Figure 7.5 *b*RAPID and Serpent pin-wise ^{135}Xe distribution and relative differences (cont.)



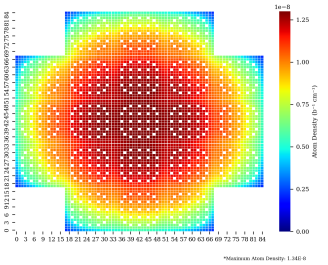
(y) *b*RAPID Day 60.875



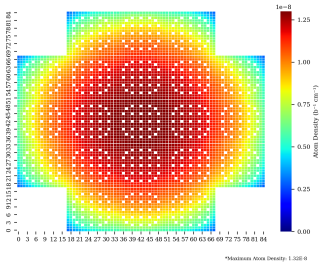
(z) Serpent Day 60.875



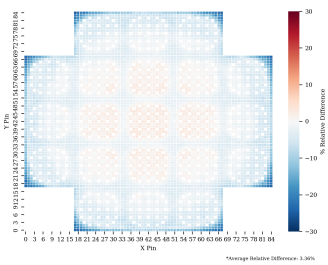
(aa) Relative Difference Day 60.875



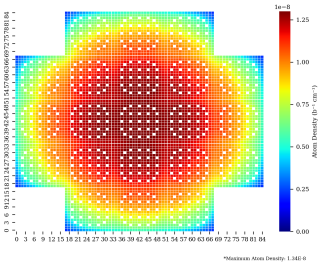
(ab) *b*RAPID Day 91.313



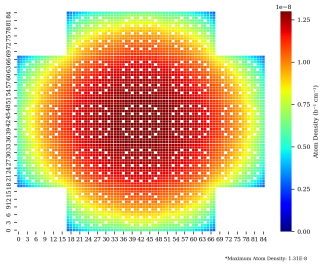
(ac) Serpent Day 91.313



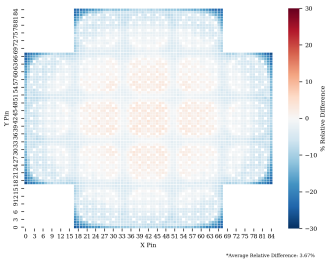
(ad) Relative Difference Day 91.313



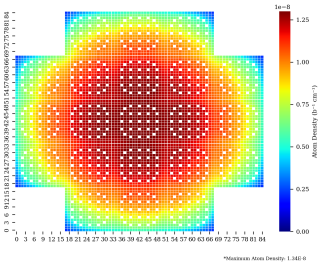
(ae) *b*RAPID Day 121.751



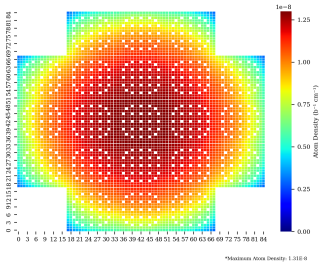
(af) Serpent Day 121.751



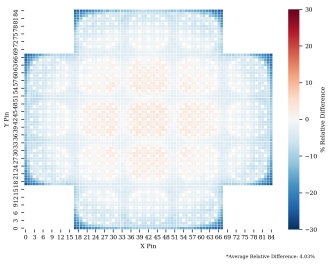
(ag) Relative Difference Day 121.751



(ah) *b*RAPID Day 152.188

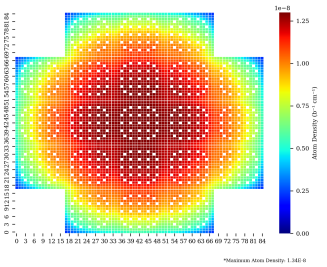


(ai) Serpent Day 152.188

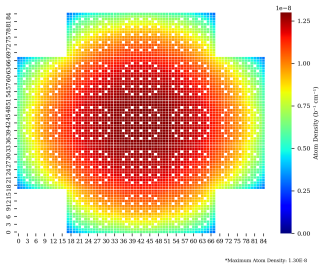


(aj) Relative Difference Day 152.188

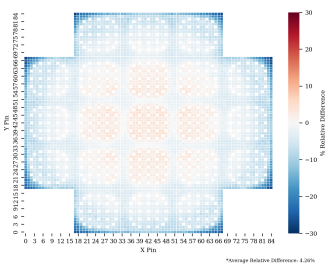
Figure 7.5 *b*RAPID and Serpent pin-wise ^{135}Xe distribution and relative differences (cont.)



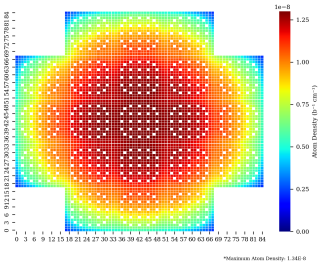
(ak) *b*RAPID Day 182.625



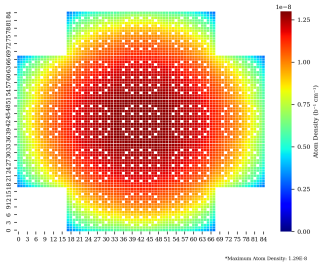
(al) Serpent Day 182.625



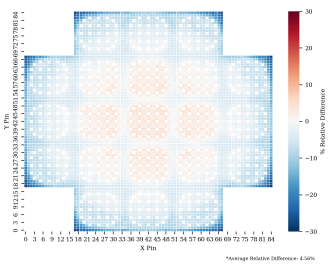
(am) Relative Difference Day 182.625



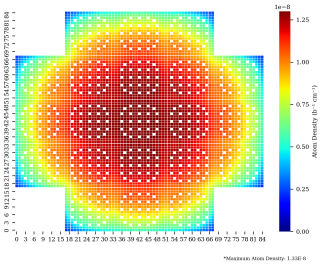
(an) *b*RAPID Day 213.063



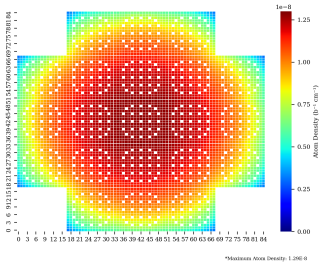
(ao) Serpent Day 213.063



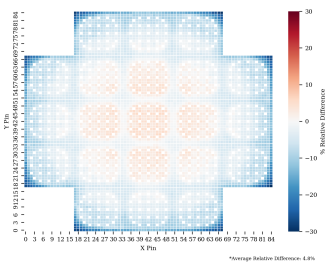
(ap) Relative Difference Day 213.063



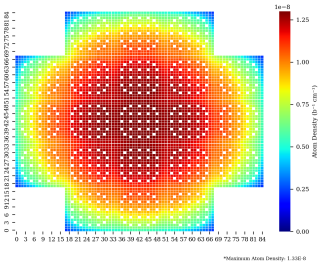
(aq) *b*RAPID Day 243.5



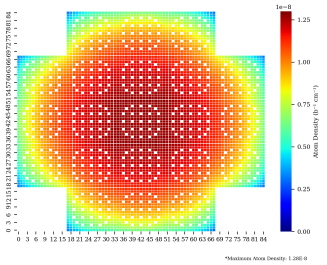
(ar) Serpent Day 243.5



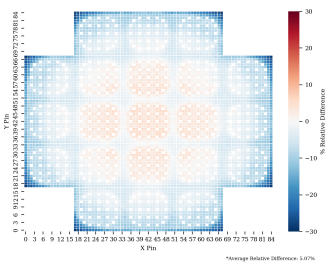
(as) Relative Difference Day 243.5



(at) *b*RAPID Day 273.938

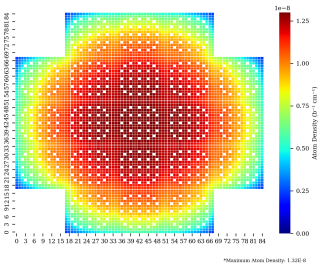


(au) Serpent Day 273.938

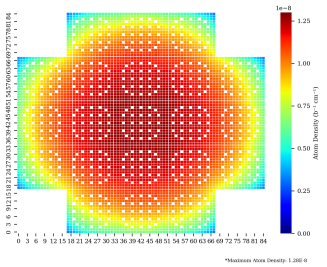


(av) Relative Difference Day 273.938

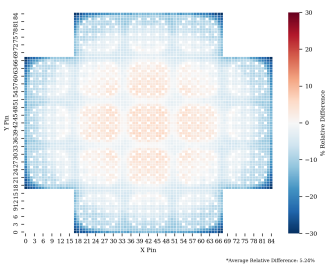
Figure 7.5 *b*RAPID and Serpent pin-wise ^{135}Xe distribution and relative differences (cont.)



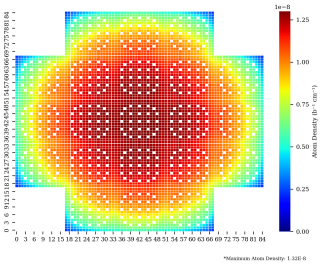
(aw) *b*RAPID Day 304.376



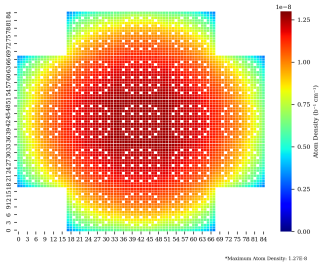
(ax) Serpent Day 304.376



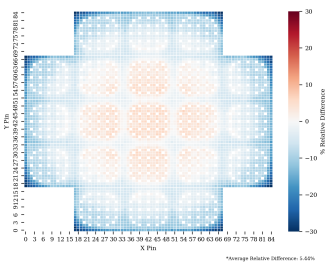
(ay) Relative Difference Day 304.376



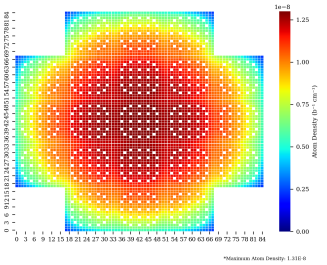
(az) *b*RAPID Day 334.813



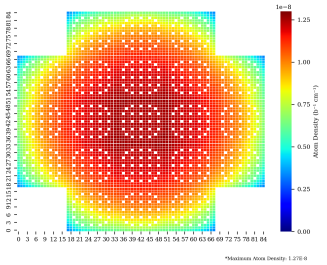
(ba) Serpent Day 334.813



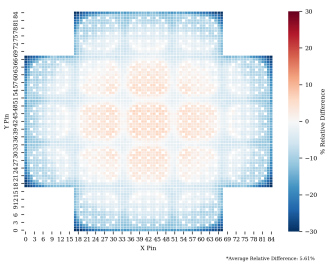
(bb) Relative Difference Day 334.813



(bc) *b*RAPID Day 365.25



(bd) Serpent Day 365.25



(be) Relative Difference Day 365.25

Figure 7.5 *b*RAPID and Serpent pin-wise ^{135}Xe distribution and relative differences (cont.)

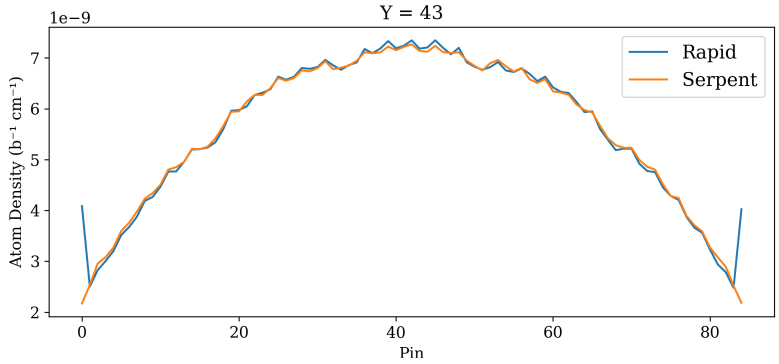


Figure 7.6 ^{135}Xe distribution for a single row of pins (Y = 43) at burnup step 1

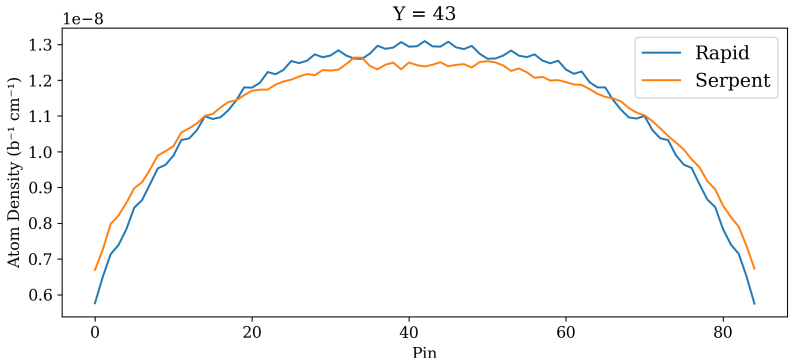


Figure 7.7 ^{135}Xe distribution for a single row of pins (Y = 43) at final burnup step

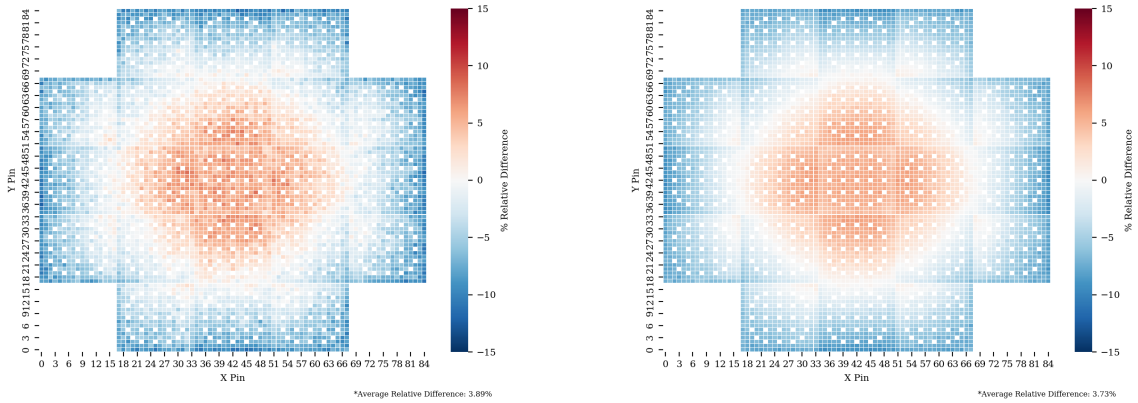
7.4 Effect of Time-Dependent Boundary Condition on Pin-wise Burnup

In Chapter 6, the time-dependent boundary correction methodology, $bnd(x, y, t)$, which uses a more realistic step-specific scheme to correct the fission source distribution was introduced. While the $bnd(x, y, t)$ method can be used independently of pin-wise burnup (the latter has no impact on the former), applying a time-dependent boundary correction does have some impact on pin-wise burnup calculations.

Consider Figure 7.8, which shows the final step burnup distribution for a pin-wise bRAPID run using $bnd(x, y)$ and $bnd(x, y, t)$. There are considerable differences between these two

plots in terms of clarity; the $bnd(x, y)$ burnup shows larger pin-to-pin variations than the $bnd(x, y, t)$, which shows an overall smooth distribution. The $bnd(x, y, t)$ method provides little/marginal improvement on the average burnup relative difference, but does improve the "shape" of the distribution to make it less noisy and more coherent. Figure 7.9 represents these differences in a single row cross section.

These differences are likely due to the physics represented behind each methodology. The $bnd(x, y, t)$ method more accurately captures the physics of the core by utilizing the step-wise burned fuel distribution when developing a boundary correction for that step. It corrects for the effect of center-core coefficient creation by incorporating the boundary effect caused by the isotopics of burned fuel. For instance– build up of absorbing and fissile/fissionable isotopes appears to impact fission neutron production throughout the core (and particularly in edge assemblies) when performing the uniform fixed-source calculation. In contrast, the time-independent $bnd(x, y)$ method simply uses a fresh fuel distribution to account for some boundary effects. The $bnd(x, y, t)$ therefore holds some information about the underlying material composition when correcting the fission source distribution, which shows effects in the pin-wise burnup calculation down the line.



(a) $bnd(x, y)$ **(b)** $bnd(x, y, t)$

Figure 7.8 $bnd(x, y)$ and $bnd(x, y, t)$ final step (365 days) pin-wise burnup distribution

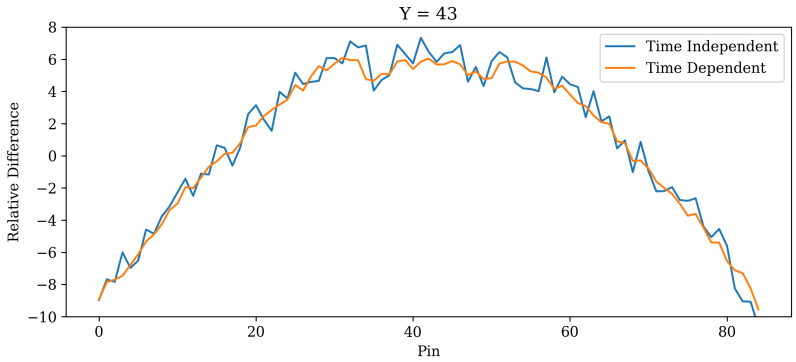


Figure 7.9 Burnup distribution for a single row of pins (Y = 43) at final burnup step for time-independent ($bnd(x, y)$) and time-dependent ($bnd(x, y, t)$) boundary corrections

7.5 Run Time Comparison Between *b*RAPID and Serpent Pin-wise Burnup

Table 7.2 shows the difference in run times between the 5x5 pin-wise Serpent run and the *b*RAPID run. One can see a massive difference in run times between these two calculations, with the *b*RAPID run only requiring 1 processor and approximately 31 minutes to complete, while the Serpent model required 38 hours and 8 processors to complete.

This run time comparison suggests that the *b*RAPID algorithm is well suited to perform pin-wise burnup calculations compared to other burnup codes like Serpent. By 'unpacking' the information held in the pin-wise fission source distribution at each burnup step, more efficient use can be made of existing information, rather than requiring pin-wise fuel region specification, which is more memory and time consuming. This *b*RAPID methodology opens the door to the potential for 3D pin-wise burnup calculations– whereas other methodologies would struggle to perform these calculations in a reasonable time period.

Table 7.2 Calculation run time for Serpent and *b*RAPID runs. All times are presented in wall-clock time

	No. Processors	**Run Time**	**Speedup**
Serpent	8	38.7 hr	-
***b*RAPID**	1	31.0 min	74.7

Chapter 8

Conclusions and Future Work

Conclusions

Sensitivity testing was performed to determine the appropriate Monte Carlo parameters (NPS, NSK, and NAC) to use for the 5x5 'mini-core' model. These tests analyzed the Shannon entropy, fission source distribution, and k_{eff} uncertainty in a burnup calculation. Recommendations from this testing can be used on similar models, but as with all Monte Carlo eigenvalue calculations, the parameters used are highly problem specific and thus generalized conclusions cannot be made.

The effect of time step resolution on Monte Carlo burnup calculations was also studied. By running multiple cases with differing constant Δt time step schemes, and one with a variable step scheme, the direct effect of time step choice on material composition was measured. A general trend of improvement with increased time step resolution was observed. However, the AARD analysis also showed that a variable time step scheme (made up of multiple Δt time step sizes) was able to adequately represent the evolution of important isotopes with less calculation time required. The FOM analysis showed that the variable step scheme was optimized to decrease computation time and increase accuracy– however a low resolution scheme (Case 1) could also be useful if speed is more valuable than accuracy to the user.

While these calculations are model-specific, it is apparent that tailoring time step selection to important isotope evolution can lead to accurate results without wasted computation time.

*b*RAPID was expanded to calculate 2D pin-wise burnup and material composition distributions, and a time dependent boundary correction algorithm was implemented.

The $bnd(x, y, t)$ methodology is a modified version of the previously time-independent boundary correction method used for static RAPID problems. The result of the algorithm was a more accurate pin-wise fission source distribution, and improvements to this distribution in the boundary pins, and an overall improvement in k_{eff} was not observed. This latter observation is likely due to improvements in the fission source distribution cancelling out– as the fission source in boundary pins was brought up, the fission source in central pins was brought down, leading to a net zero difference in k_{eff} . Improvements were seen in the pin-wise burnup algorithm when the $bnd(x, y, t)$ methodology was applied– the result was a less "noisy" burnup distribution, but the overall trend of the distribution stayed the same. The analysis shown here indicates that the $bnd(x, y, t)$ methodology provides a more accurate fission source distribution by representing the boundary physics caused by burned fuel assemblies in a reactor core. This is useful, as detailed analyses require accurate distributions first and foremost. The study of this algorithm also provides greater insight into the impact of burnup in boundary corrections for future reference.

The pin-wise burnup algorithm presented here uses the existing RAPID pin-wise fission source distribution outputs to calculate burnup and interpolate material compositions. Results show agreement with the Serpent reference calculation, with burnup relative difference values reaching approximately 5% to 10% in central pins and -5% to -10% in boundary pins for the final burnup step. Early burnup steps show smaller relative differences, hovering below ±5% in the center/boundary locations. Differences increase over time, likely due to the accumulation of errors in the fission source distribution. For most important isotopes, material composition relative differences were low in the center (below ±5%) but underestimated on the boundaries (20–30% in some cases). However, the total mass of the important

isotopes in the core affected by this underestimation (as measured by PUP) was quite small, around 1-4%. Effectively, the underestimation in edge pins would have little impact on the whole-core isotopic calculation. Underestimation in edge pins points to the need for more accurate databases when performing pin-wise burnup, and more methods for dealing with boundary physics in RAPID. Increased time step resolution (in early steps) could alleviate some of the underestimation. This pin-wise methodology has been shown to not only be accurate (relative to the Serpent reference), but also fast– requiring a fraction of the time of traditional burnup methodologies. By 'unpacking' the information held by the pin-wise fission source distribution at each burnup step, the pin-wise burnup methodology can calculate quantities such as burnup and material composition without the computationally expensive process of specifying each pin as an independent fuel region to be tracked. The efficiency of this methodology opens the door to more expensive computation– such as full-core 3D pin-wise burnup analysis.

Future Work

Future work in the area of Monte Carlo burnup (particularly for *b*RAPID database development) may look towards the prospect of modeling ^{135}Xe isotopic evolution separately from the standard Monte Carlo burnup calculation, then accounting for the missing isotope with another calculation. Much of the computation time in a burnup problem is spent trying to adequately represent ^{135}Xe evolution. By decoupling these calculations for this problematic isotope, it may be possible to decrease the number of burnup steps required in the Monte Carlo burnup pre-calculation stage of database creation. Additionally, more efficient time step scheme selection for databases could be achieved through the creation of an adaptive time step scheme development tool. Such a tool could automate the heuristic process outlined in Chapter 5, and provide the most effective scheme for a given model.

As it currently stands, *b*RAPID still performs calculations based on assembly-wise data due to the input specifications of the RAPID code system. It is possible to change this– however this will require major changes to the RAPID code. In order to provide the most accurate

pin-wise burnup and material composition data possible, these changes could be made, as it would allow *b*RAPID to perform burnup calculations at the pin-level using pin-wise input specification. However, this process would likely require much more memory usage, and the increase in accuracy provided by making this change is unknown. A more direct application of this methodology would be to expand it to 3D pin-wise burnup, which would be relatively simple due to the structure of the algorithm. One difficulty with this would be the reference calculation for comparison– practically no burnup codes can efficiently produce 3D pin-wise burnup and material composition information. If *b*RAPID was expanded to calculate these, finding a suitable reference code may be difficult.

Studies into the fission source distribution differences of neighboring assemblies with different material compositions over time should be performed as well. While a material-to-material correction factor exists in RAPID currently to account for material discontinuities, differences in material composition of neighboring assemblies may still be causing problems for fission source calculations.

Appendices

Appendix A

bRAPID Operation

A.1 Creating a bRAPID Database using pRAPID

The files needed to create a database with pRAPID are:

- shared/`dim.py` – The primary pRAPID input file containing material, cross-section, and bRAPID database information
- shared/`prb_gen.tmp` – Serpent input file used as a template for input generation
- shared/`geom` – Serpent geometry file used for your model (should match your model and `dim.py`)
- shared/`mats` – Material definition input (should match your model and `dim.py`)
- shared/`tals` – Tally file used for FM coefficient calculation
- shared/temps/`<name>` – Environment used in automated input generation
- shared/temps/`<name.inf>` – Environment used in automated input generation
- burnup/shared/`burnup` – Burnup step specification file

- burnup/shared/`decay` – Decay step specification file
- burnup/shared/`inventory` – List of isotopes to track in burnup pre-calculation

The database used for this work has 5 power densities (10E-3, 20E-3, 30E-3, 40E-3, 50E-3 kW/g) and 19 burnup steps specified (0.5, 1.0, 1.5 ... 365.25 days) specified. To create such a database, one would open `dim.py` and find the `pow` line and input:

```
pow = [ 10E-3, 20E-3, 30E-3, 40E-3, 50E-3 ]
```

then go to burnup/shared/`burnup` and input:

```
0.5
1.0
1.5
...
365.25
```

After properly specifying the necessary input files, one will execute the following commands to create a bRAPID database using pRAPID:

1. `./prapid.py -mb 1` : Creates the burnup and decay pre-calculation files and executes a Serpent burnup calculation on cluster
2. `./prapid.py -rb 1` : Processes the burnup pre-calculation output, and submits decay calculations to cluster
3. `./prapid.py -rd 1` : Processes decay outputs
4. `./prapid.py -fd` : Creates FM coefficient calculation directories and populates them

5. `./prapid.py -mt` : Creates fixed source FM calculation files and submits a test calculation to cluster

6. `./prapid.py -ra` : Submits all FM coefficient calculations to cluster

7. `./prapid.py -pa` : Processes FM coefficient calculations and compresses them into database files, stored in the database directory

Table A.1 shows the database structure that would result in executing commands 1 - 7.

Table A.1 Database structure for n power densities and irradiation times (b_n,t_n)

	b1	**b2**	**...**	**bn**
t1	b1t1	b2t2	...	bnt1
t2	b1t2	b2t2	...	bnt2
...	...	...	...	...
tn	b1tn	b2tn	...	bntn

A.2 bRAPID Execution and Outputs

In order to perform a bRAPID calculation, a problem definition must be given to bRAPID. The following files define the calculation parameters for bRAPID to use:

- `burn.inp`: The primary bRAPID input file, specifying the following:
  - `nsteps` : Number of burnup steps
  - `spower` : User specified core average power density
  - `stime` : Time steps in cumulative days
  - `nmass` : Mass of fuel used for power normalization (kg)
  - `nbiso` : Number of isotopes tracked

  - `is3D :` 3D specification (1 = 3D, 0 = 2D)
  - `br_db :` bRAPID database file path

- `<name>_b0.inp:` The initial RAPID input file for burnup step 0. This can either be user-specified or automatically generated by bRAPID (with fresh fuel and zero irradiation time by default)
- `<name>.tmp:` The template RAPID input file, used for generating subsequent RAPID input files during bRAPID execution

Once each file and parameter is specified, bRAPID can by executed by the command `./brapid.py`. First, RAPID will be called to execute an eigenvalue calculation using the `<name>_b0.inp` file described above. The result of this calculation will be an assembly-wise fission source distribution $F_i(x, y, z)$, which is used by bRAPID to generate the parameters for the next burnup step, among other things. This process is repeated until the last burnup step.

Appendix B

Pin-wise Isotopic Composition and Relative Differences for Important Isotopes

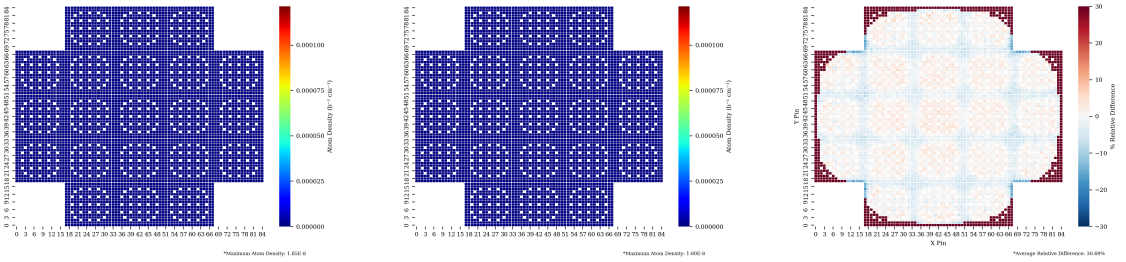
B.1 All Isotopes Tracked

A list of all the isotopes tracked is shown in Table B.1

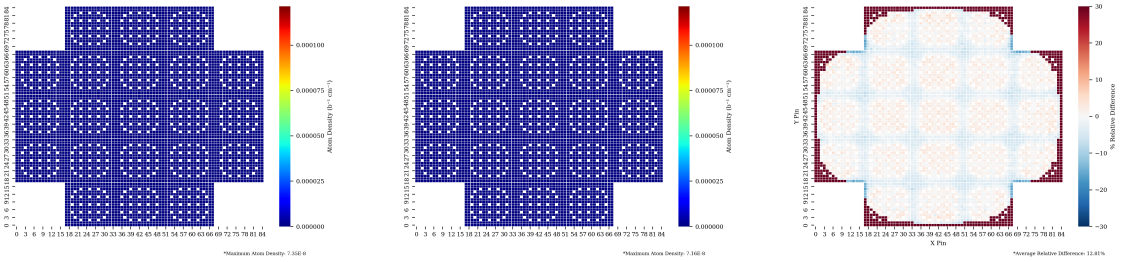
Table B.1 List of all isotopes tracked in the burnup calculations for this thesis

234U	235U	236U	237U	238U	239U	236Np	237Np	238Np	239Np
236Pu	238Pu	239Pu	240Pu	241Pu	242Pu	243Pu	241Am	242Am	243Am
244Am	242mAm	242Cm	243Cm	244Cm	245Cm	246Cm	247Cm	248Cm	249Cm
249Bk	250Bk	249Cf	250Cf	251Cf					
73Ge	74Ge	76Ge	75As	77Se	78Se	79Se	80Se	82Se	81Br
82Kr	83Kr	84Kr	85Kr	86Kr	85Rb	86Rb	87Rb	88Sr	89Sr
90Sr	89Y	90Y	91Y	90Zr	91Zr	92Zr	93Zr	94Zr	95Zr
96Zr	95Nb	95Mo	96Mo	97Mo	98Mo	100Mo	103Rh	103Rh	105Rh
104Pd	105Pd	106Pd	107Pd	108Pd	110Pd	109Ag	111Ag	110Cd	111Cd
112Cd	113Cd	114Cd	116Cd	116Sn	117Sn	118Sn	119Sn	120Sn	122Sn
123Sn	124Sn	125Sn	126Sn	121Sb	123Sb	124Sb	125Sb	126Sb	125Te
126Te	128Te	130Te	127I	129I	130I	135I	130Xe	131Xe	132Xe
133Xe	134Xe	135Xe	136Xe	133Cs	134Cs	135Cs	136Cs	137Cs	134Ba
136Ba	137Ba	138Ba	140Ba	139La	140La	140Ce	141Ce	142Ce	144Ce
141Pr	143Pr	142Nd	143Nd	144Nd	145Nd	146Nd	147Nd	148Nd	150Nd
147Pm	148Pm	149Pm	148mPm	147Sm	148Sm	149Sm	150Sm	151Sm	152Sm
154Sm	152Eu	153Eu	154Eu	155Eu	156Eu	152Gd	154Gd	155Gd	156Gd
157Gd	158Gd	160Gd	159Tb	160Tb	160Dy	161Dy	162Dy	163Dy	164Dy
16O	17O	4He							

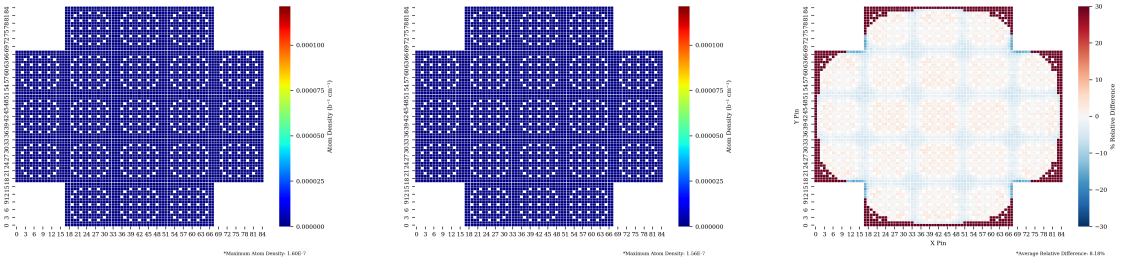
B.2 ^{239}Pu



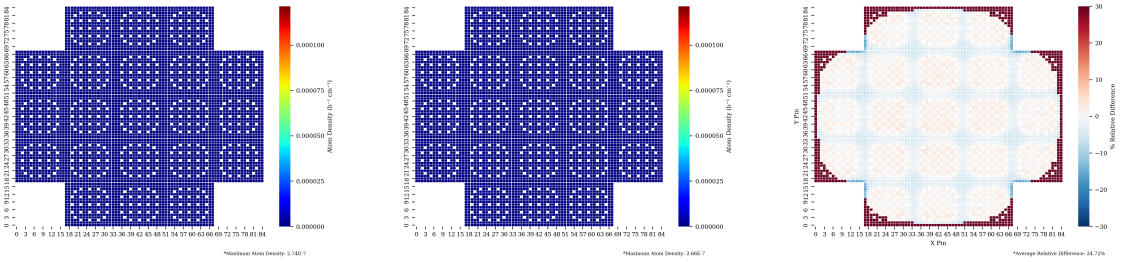
(a) bRAPID Day 0.5 **(b)** Serpent Day 0.5 **(c)** Relative Difference Day 0.5



(d) bRAPID Day 1.0 **(e)** Serpent Day 1.0 **(f)** Relative Difference Day 1.0

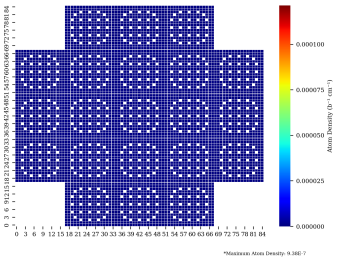


(g) bRAPID Day 1.5 **(h)** Serpent Day 1.5 **(i)** Relative Difference Day 1.5

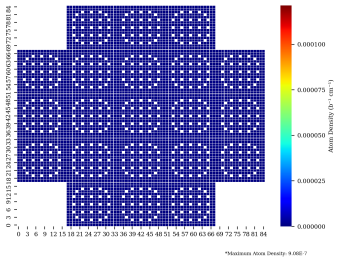


(j) bRAPID Day 2.0 **(k)** Serpent Day 2.0 **(l)** Relative Difference Day 2.0

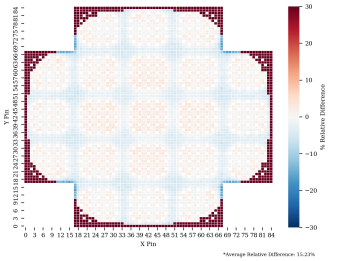
Figure B.1 bRAPID and Serpent pin-wise ^{239}Pu distribution and relative differences



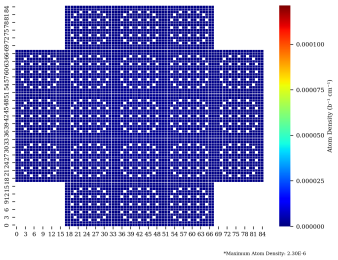
(m) bRAPID Day 4.0



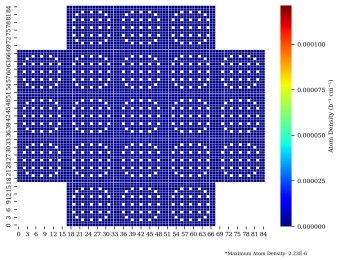
(n) Serpent Day 4.0



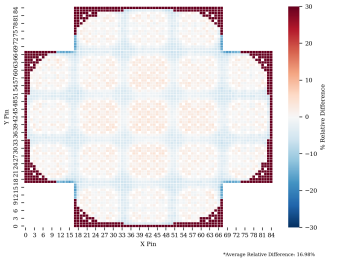
(o) Relative Difference Day 4.0



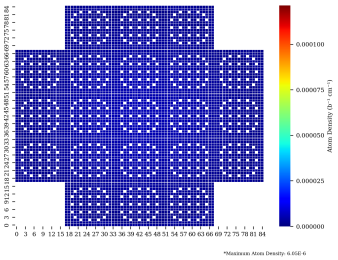
(p) bRAPID Day 7.0



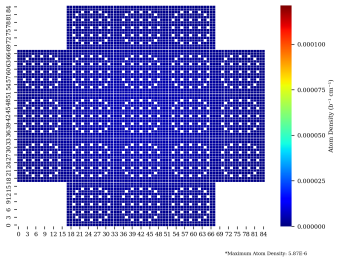
(q) Serpent Day 7.0



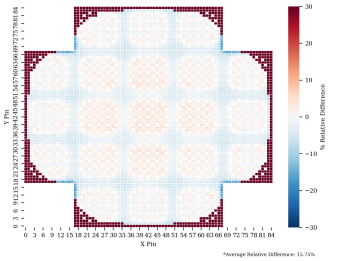
(r) Relative Difference Day 7.0



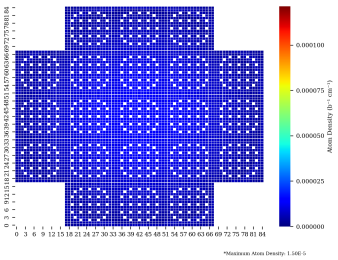
(s) bRAPID Day 14.0



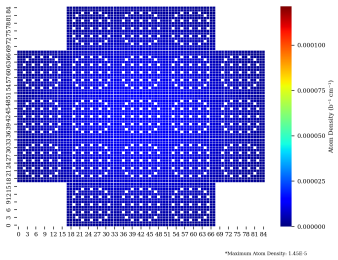
(t) Serpent Day 14.0



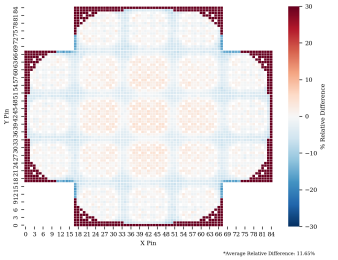
(u) Relative Difference Day 14.0



(v) bRAPID Day 30.438

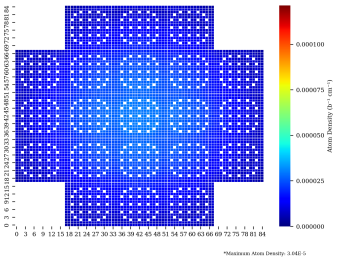


(w) Serpent Day 30.438

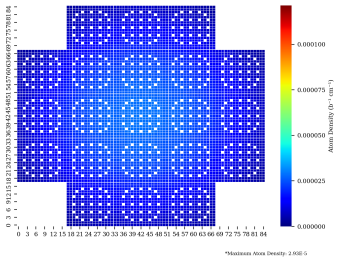


(x) Relative Difference Day 30.438

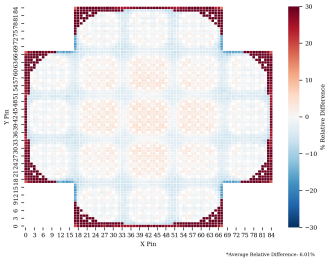
Figure B.1 bRAPID and Serpent pin-wise ^{239}Pu distribution and relative differences (cont.)



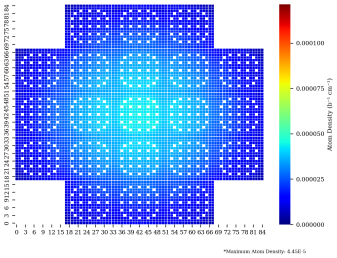
(y) bRAPID Day 60.875



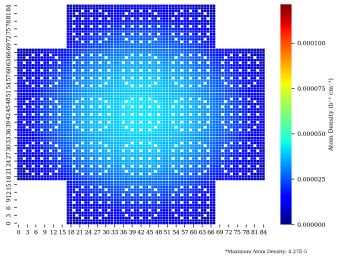
(z) Serpent Day 60.875



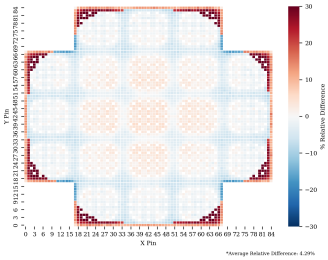
(aa) Relative Difference Day 60.875



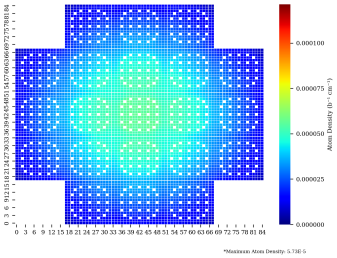
(ab) bRAPID Day 91.313



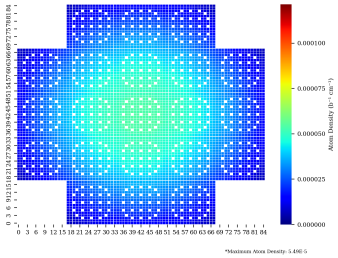
(ac) Serpent Day 91.313



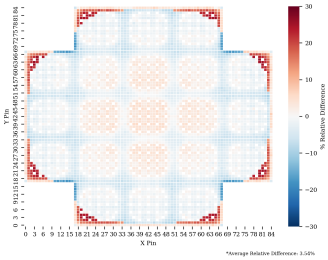
(ad) Relative Difference Day 91.313



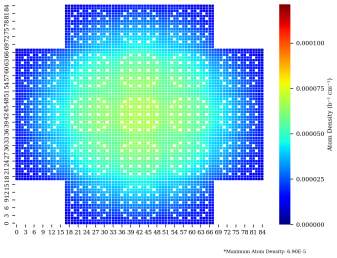
(ae) bRAPID Day 121.751



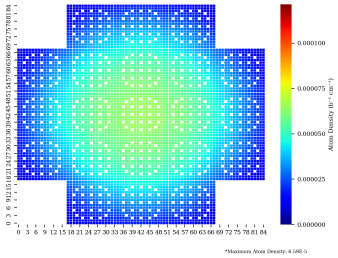
(af) Serpent Day 121.751



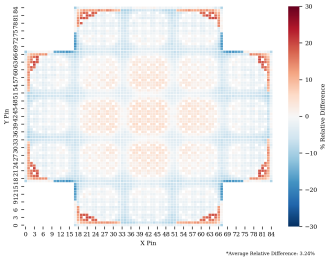
(ag) Relative Difference Day 121.751



(ah) bRAPID Day 152.188

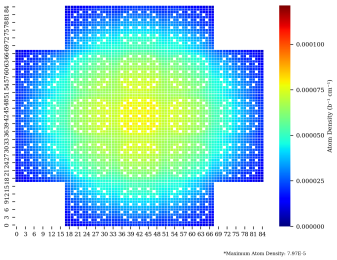


(ai) Serpent Day 152.188

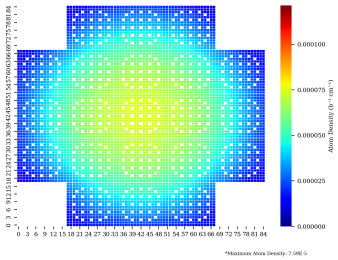


(aj) Relative Difference Day 152.188

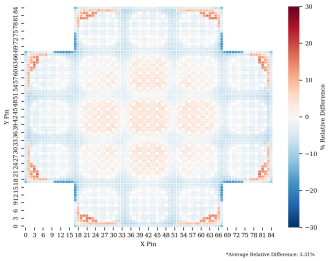
Figure B.1 bRAPID and Serpent pin-wise ^{239}Pu distribution and relative differences (cont.)



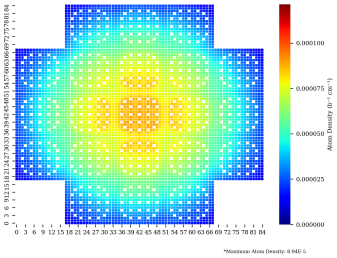
(ak) bRAPID Day 182.625



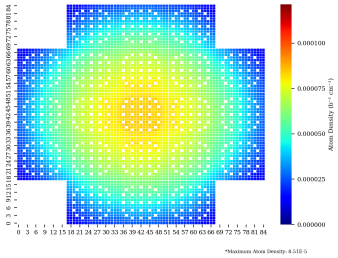
(al) Serpent Day 182.625



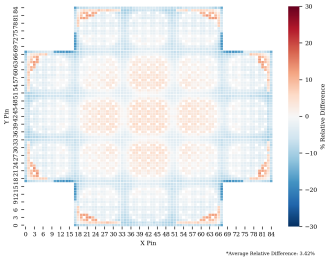
(am) Relative Difference Day 182.625



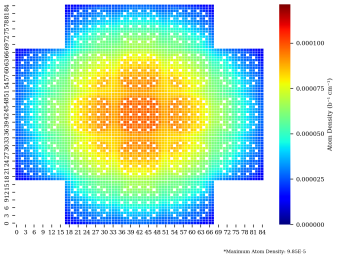
(an) bRAPID Day 213.063



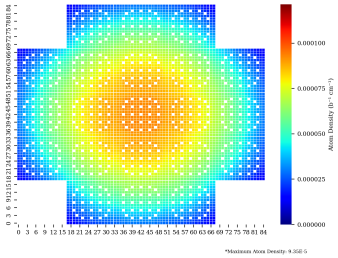
(ao) Serpent Day 213.063



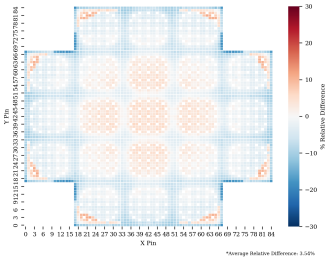
(ap) Relative Difference Day 213.063



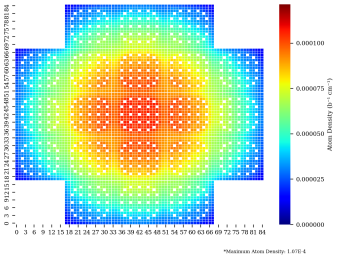
(aq) bRAPID Day 243.5



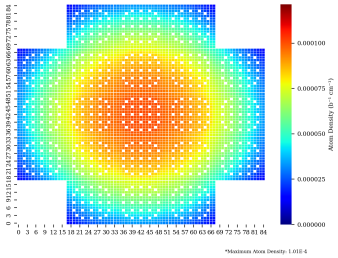
(ar) Serpent Day 243.5



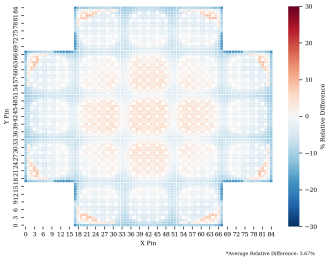
(as) Relative Difference Day 243.5



(at) bRAPID Day 273.938

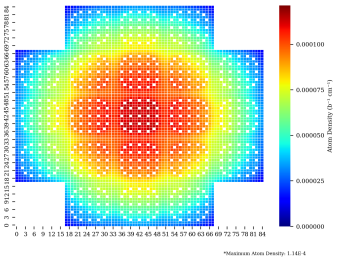


(au) Serpent Day 273.938

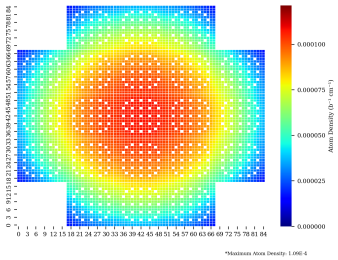


(av) Relative Difference Day 273.938

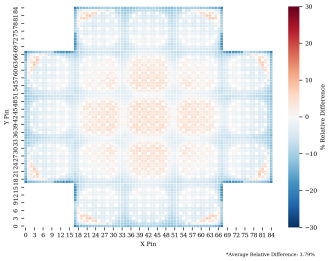
Figure B.1 bRAPID and Serpent pin-wise ^{239}Pu distribution and relative differences (cont.)



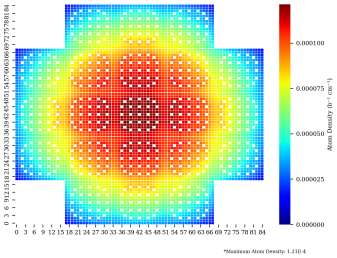
(aw) bRAPID Day 304.376



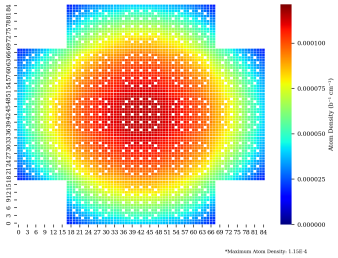
(ax) Serpent Day 304.376



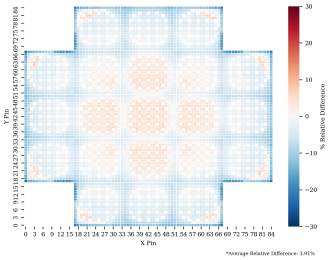
(ay) Relative Difference Day 304.376



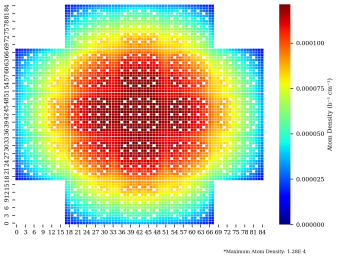
(az) bRAPID Day 334.813



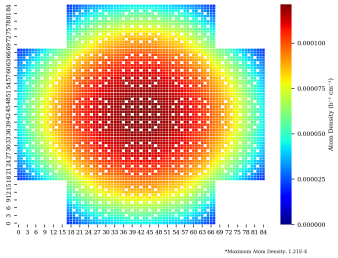
(ba) Serpent Day 334.813



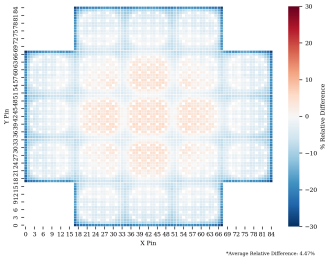
(bb) Relative Difference Day 334.813



(bc) bRAPID Day 365.25



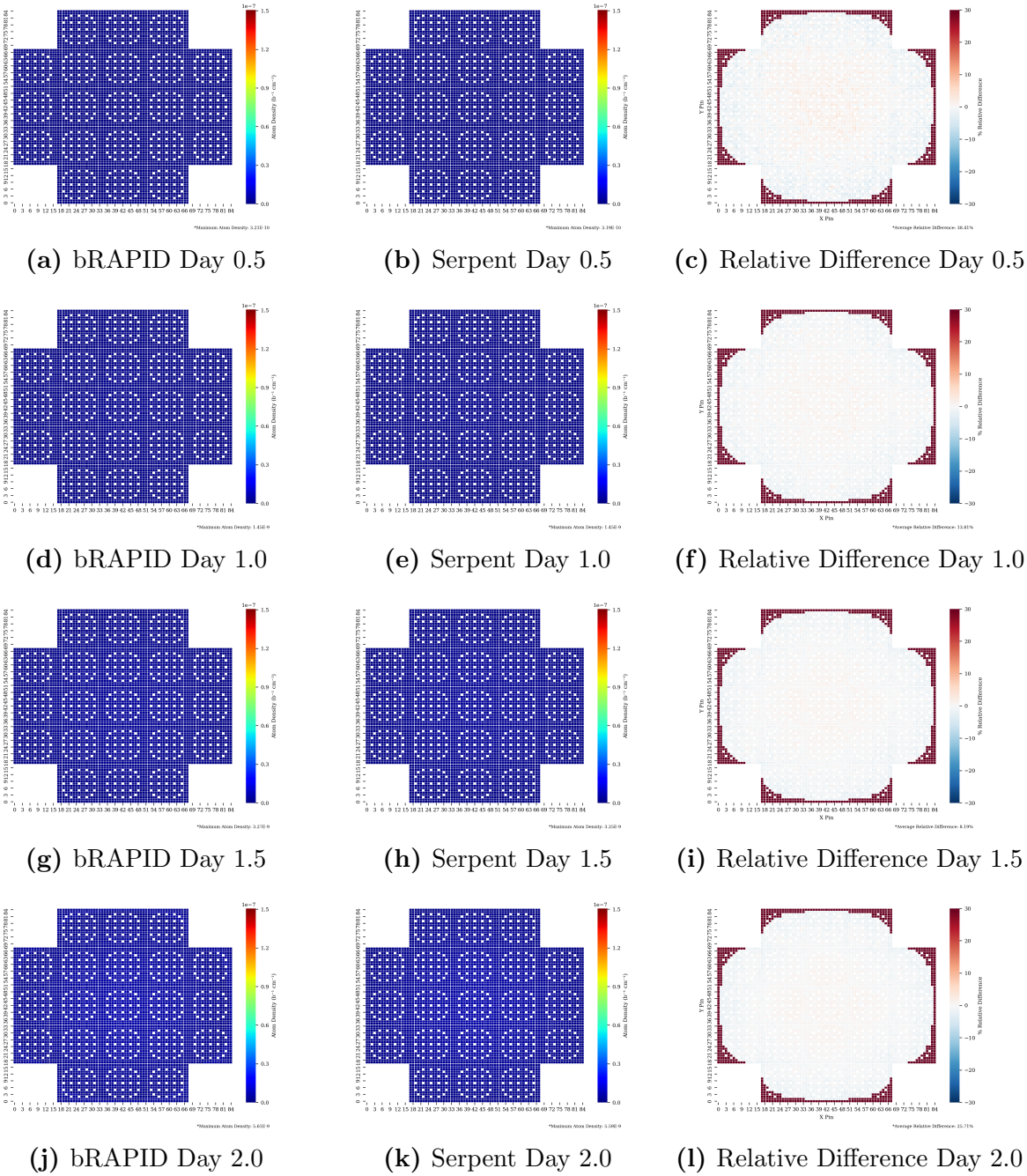
(bd) Serpent Day 365.25



(be) Relative Difference Day 365.25

Figure B.1 bRAPID and Serpent pin-wise ^{239}Pu distribution and relative differences (cont.)

B.3 ^{149}Sm



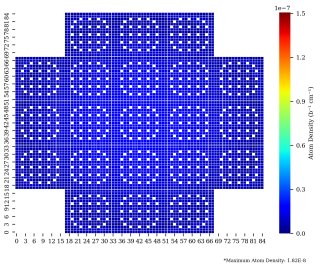
(a) bRAPID Day 0.5 **(b)** Serpent Day 0.5 **(c)** Relative Difference Day 0.5

(d) bRAPID Day 1.0 **(e)** Serpent Day 1.0 **(f)** Relative Difference Day 1.0

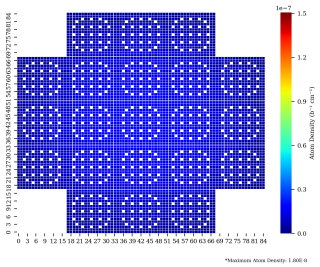
(g) bRAPID Day 1.5 **(h)** Serpent Day 1.5 **(i)** Relative Difference Day 1.5

(j) bRAPID Day 2.0 **(k)** Serpent Day 2.0 **(l)** Relative Difference Day 2.0

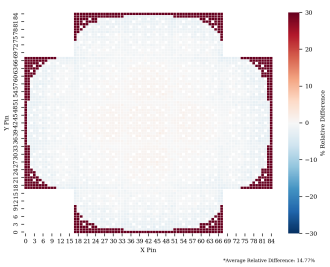
Figure B.2 bRAPID and Serpent pin-wise ^{149}Sm distribution and relative differences



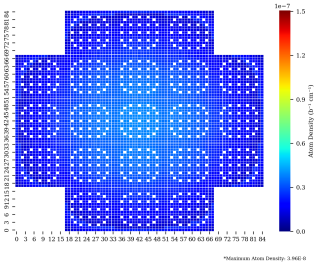
(m) bRAPID Day 4.0



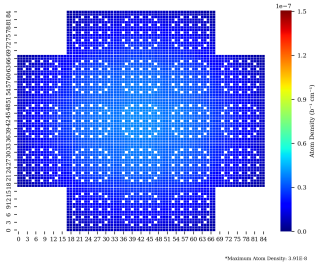
(n) Serpent Day 4.0



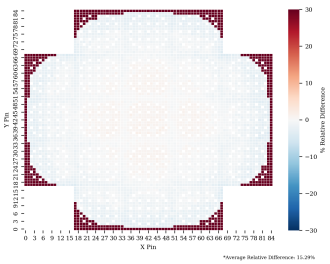
(o) Relative Difference Day 4.0



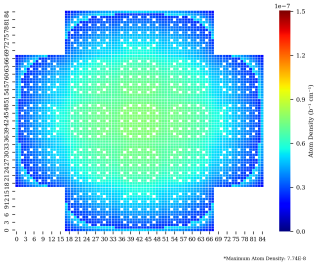
(p) bRAPID Day 7.0



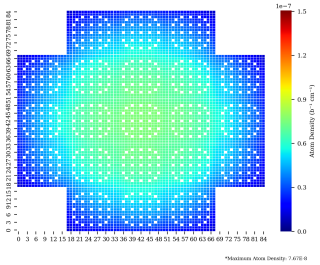
(q) Serpent Day 7.0



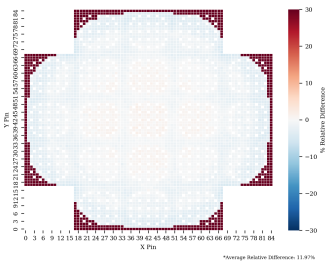
(r) Relative Difference Day 7.0



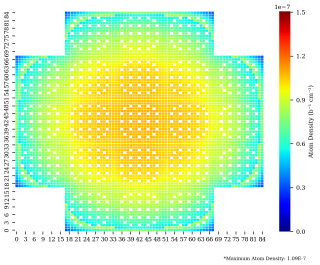
(s) bRAPID Day 14.0



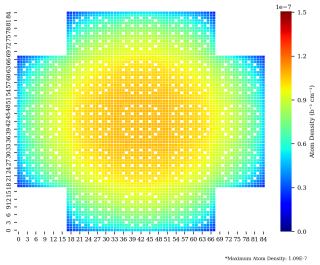
(t) Serpent Day 14.0



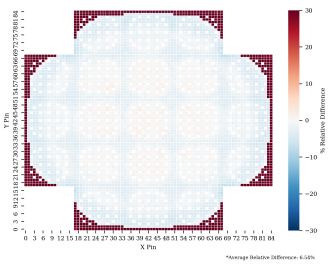
(u) Relative Difference Day 14.0



(v) bRAPID Day 30.438

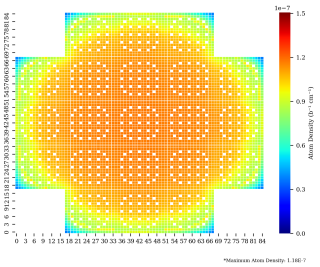


(w) Serpent Day 30.438

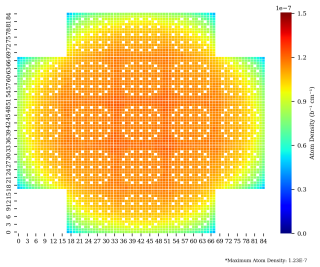


(x) Relative Difference Day 30.438

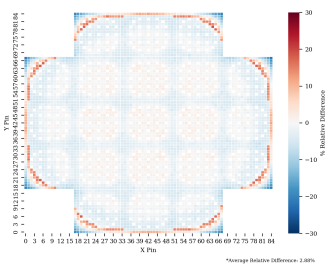
Figure B.2 bRAPID and Serpent pin-wise ^{149}Sm distribution and relative differences (cont.)



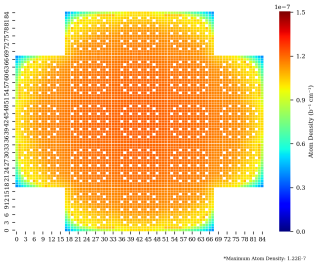
(y) bRAPID Day 60.875



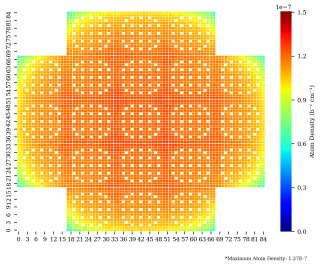
(z) Serpent Day 60.875



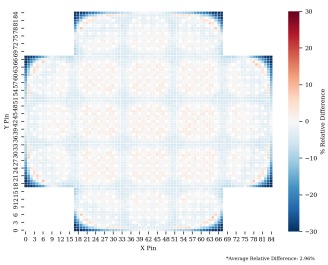
(aa) Relative Difference Day 60.875



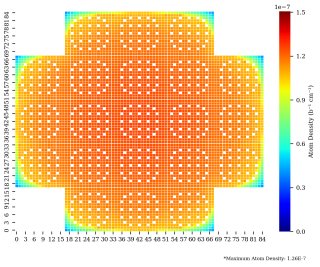
(ab) bRAPID Day 91.313



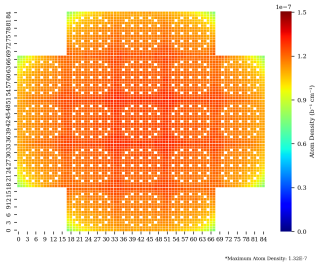
(ac) Serpent Day 91.313



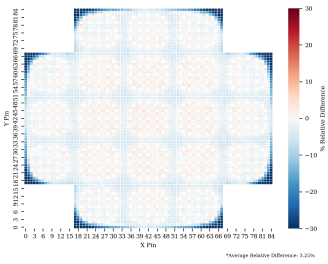
(ad) Relative Difference Day 91.313



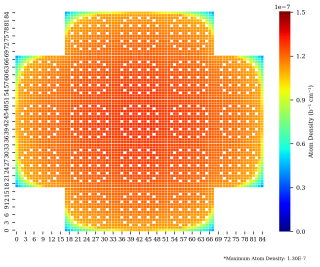
(ae) bRAPID Day 121.751



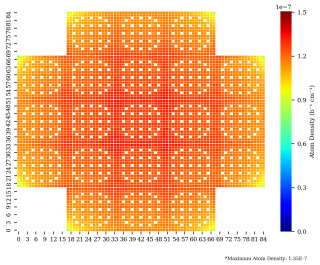
(af) Serpent Day 121.751



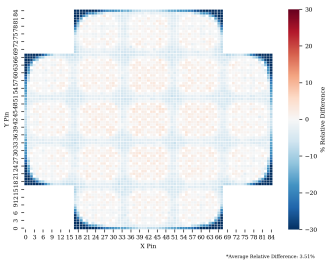
(ag) Relative Difference Day 121.751



(ah) bRAPID Day 152.188

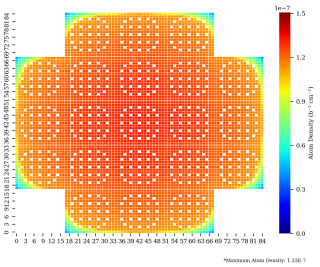


(ai) Serpent Day 152.188

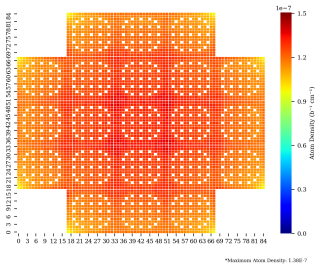


(aj) Relative Difference Day 152.188

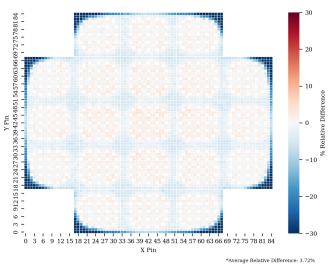
Figure B.2 bRAPID and Serpent pin-wise ^{149}Sm distribution and relative differences (cont.)



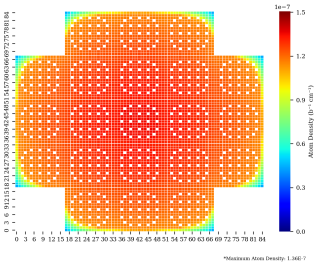
(ak) bRAPID Day 182.625



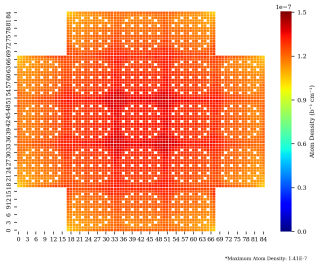
(al) Serpent Day 182.625



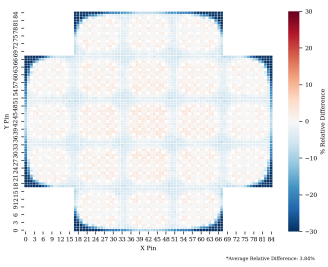
(am) Relative Difference Day 182.625



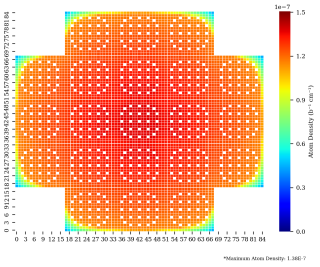
(an) bRAPID Day 213.063



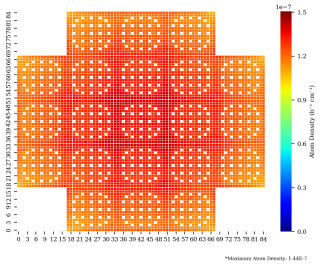
(ao) Serpent Day 213.063



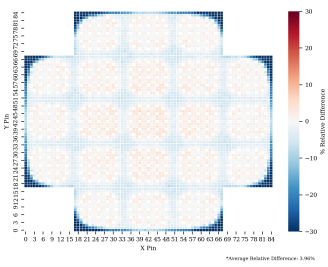
(ap) Relative Difference Day 213.063



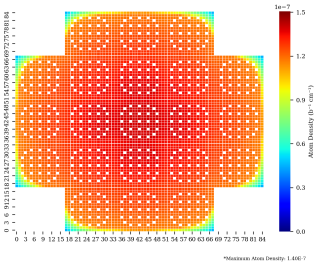
(aq) bRAPID Day 243.5



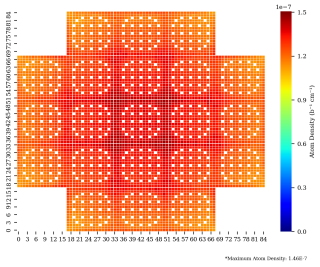
(ar) Serpent Day 243.5



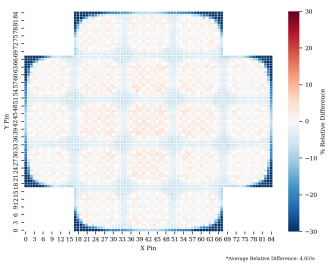
(as) Relative Difference Day 243.5



(at) bRAPID Day 273.938

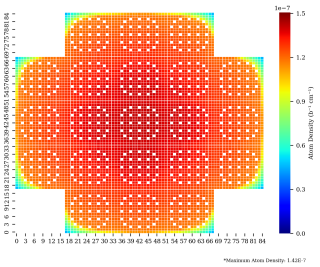


(au) Serpent Day 273.938

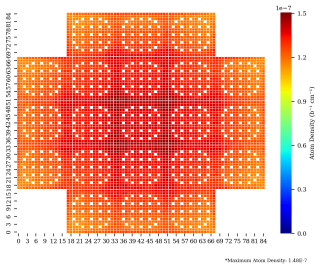


(av) Relative Difference Day 273.938

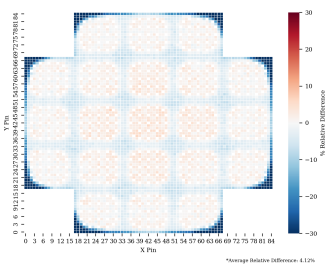
Figure B.2 bRAPID and Serpent pin-wise ^{149}Sm distribution and relative differences (cont.)



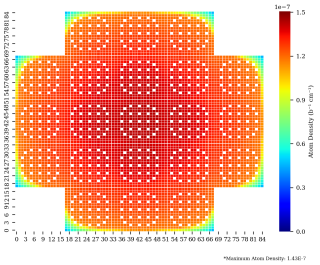
(aw) bRAPID Day 304.376



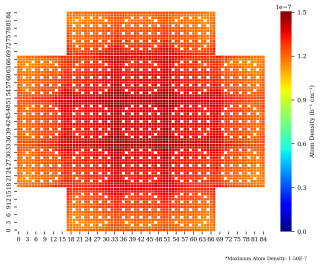
(ax) Serpent Day 304.376



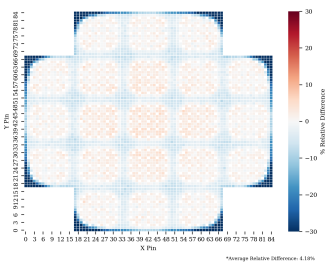
(ay) Relative Difference Day 304.376



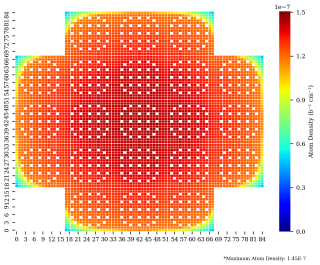
(az) bRAPID Day 334.813



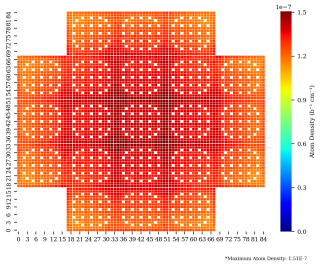
(ba) Serpent Day 334.813



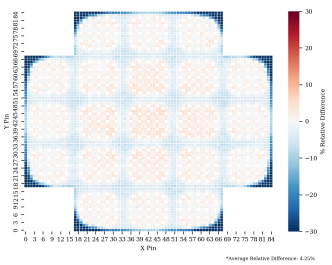
(bb) Relative Difference Day 334.813



(bc) bRAPID Day 365.25



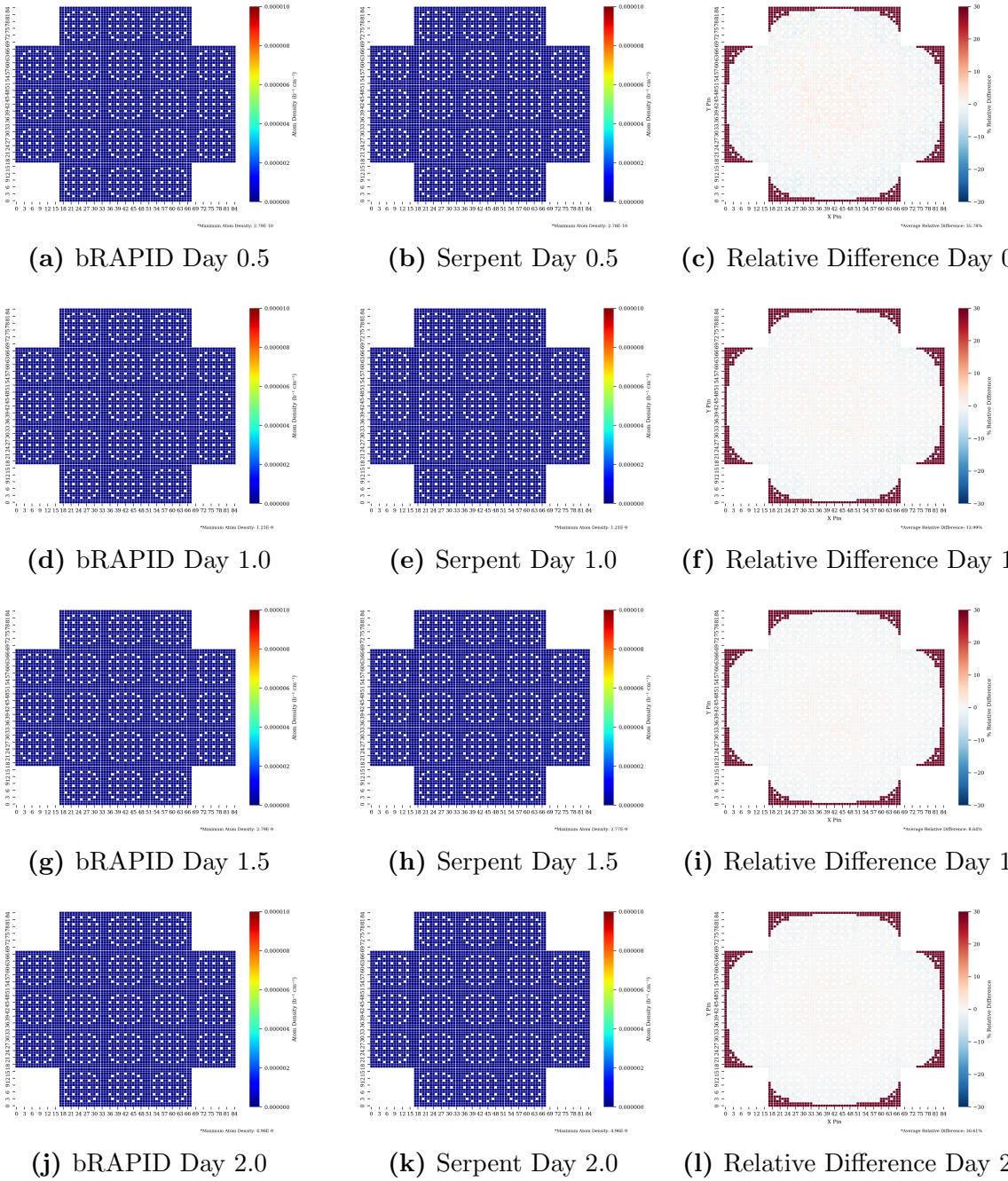
(bd) Serpent Day 365.25



(be) Relative Difference Day 365.25

Figure B.2 bRAPID and Serpent pin-wise ^{149}Sm distribution and relative differences (cont.)

B.4 ^{131}Xe



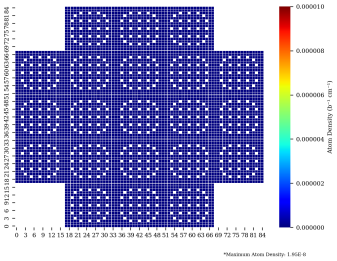
(a) bRAPID Day 0.5 **(b)** Serpent Day 0.5 **(c)** Relative Difference Day 0.5

(d) bRAPID Day 1.0 **(e)** Serpent Day 1.0 **(f)** Relative Difference Day 1.0

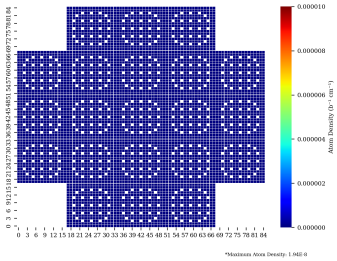
(g) bRAPID Day 1.5 **(h)** Serpent Day 1.5 **(i)** Relative Difference Day 1.5

(j) bRAPID Day 2.0 **(k)** Serpent Day 2.0 **(l)** Relative Difference Day 2.0

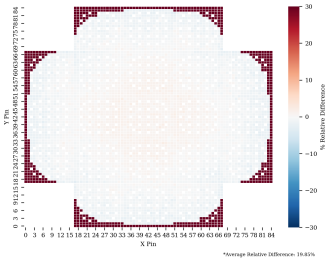
Figure B.3 bRAPID and Serpent pin-wise ^{131}Xe distribution and relative differences



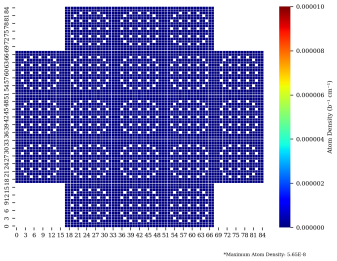
(m) bRAPID Day 4.0



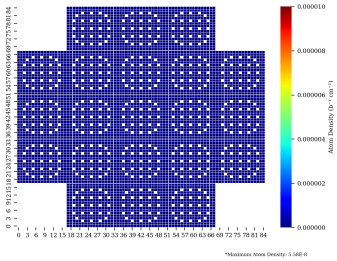
(n) Serpent Day 4.0



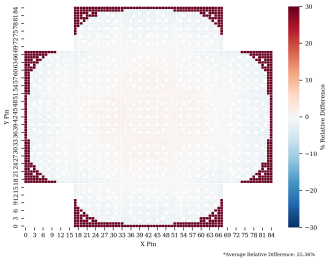
(o) Relative Difference Day 4.0



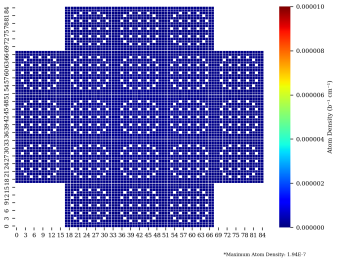
(p) bRAPID Day 7.0



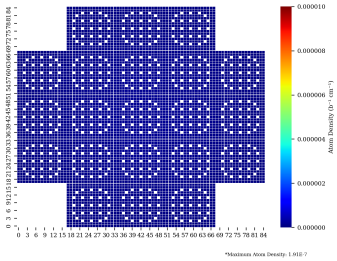
(q) Serpent Day 7.0



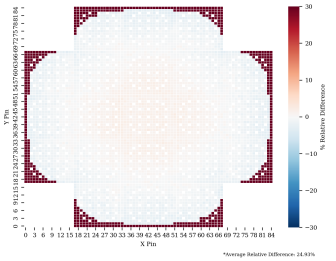
(r) Relative Difference Day 7.0



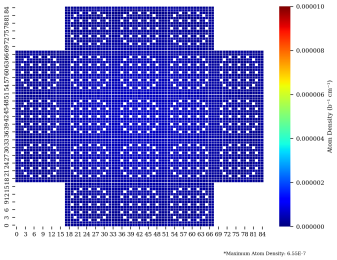
(s) bRAPID Day 14.0



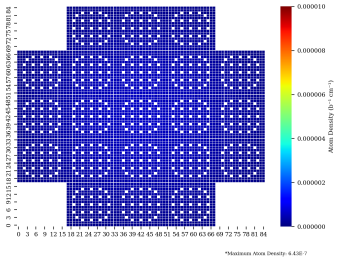
(t) Serpent Day 14.0



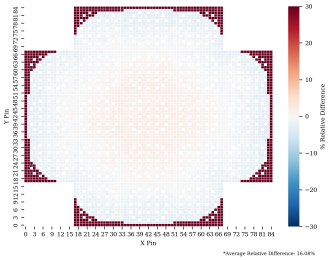
(u) Relative Difference Day 14.0



(v) bRAPID Day 30.438

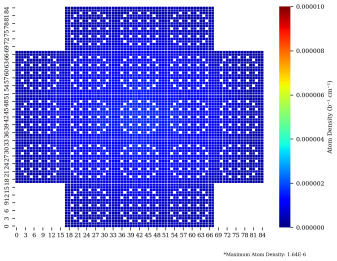


(w) Serpent Day 30.438

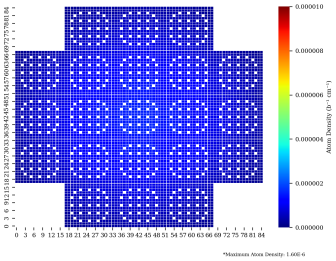


(x) Relative Difference Day 30.438

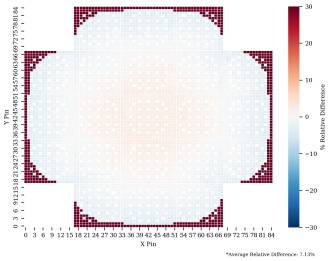
Figure B.3 bRAPID and Serpent pin-wise ^{131}Xe distribution and relative differences (cont.)



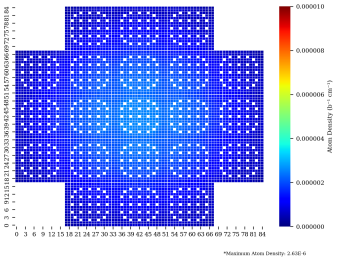
(y) bRAPID Day 60.875



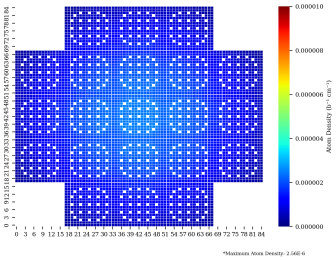
(z) Serpent Day 60.875



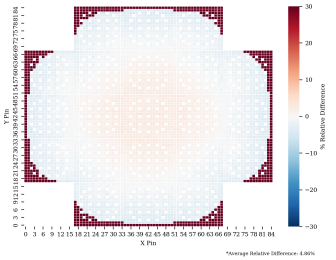
(aa) Relative Difference Day 60.875



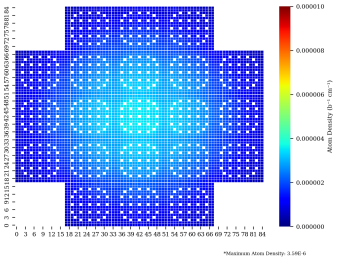
(ab) bRAPID Day 91.313



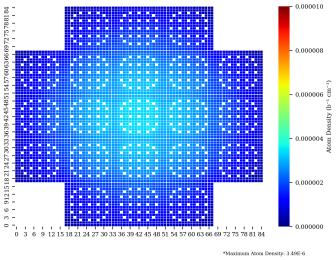
(ac) Serpent Day 91.313



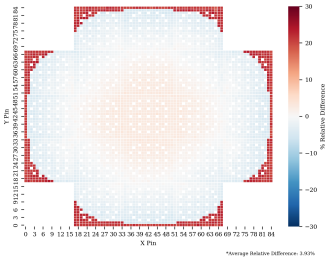
(ad) Relative Difference Day 91.313



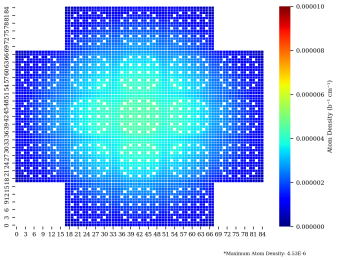
(ae) bRAPID Day 121.751



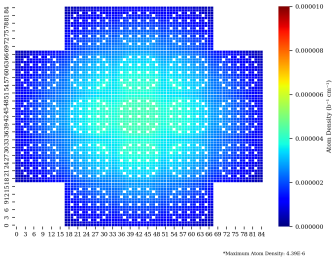
(af) Serpent Day 121.751



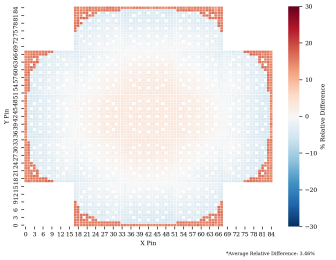
(ag) Relative Difference Day 121.751



(ah) bRAPID Day 152.188

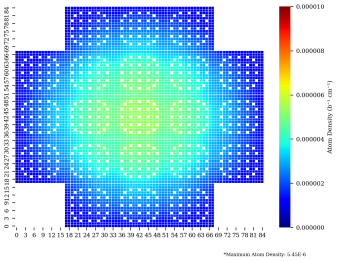


(ai) Serpent Day 152.188

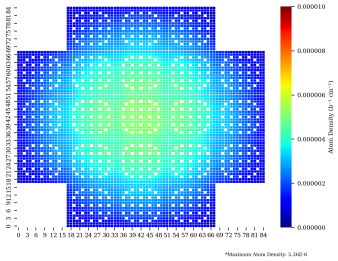


(aj) Relative Difference Day 152.188

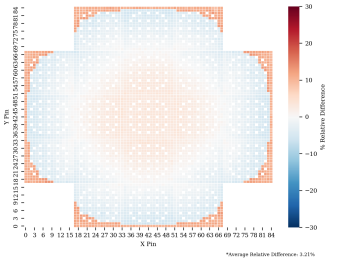
Figure B.3 bRAPID and Serpent pin-wise ^{131}Xe distribution and relative differences (cont.)



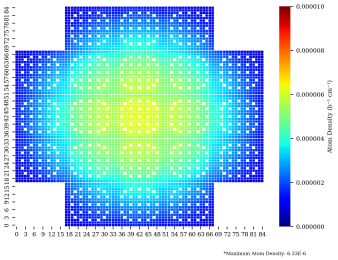
(ak) bRAPID Day 182.625



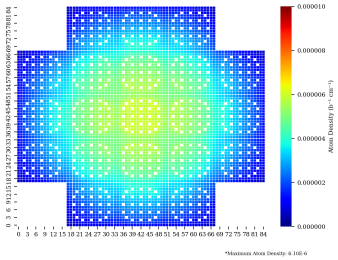
(al) Serpent Day 182.625



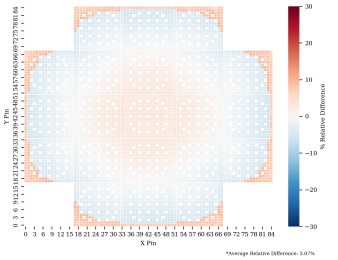
(am) Relative Difference Day 182.625



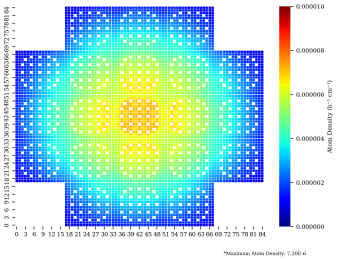
(an) bRAPID Day 213.063



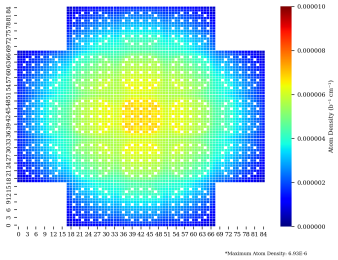
(ao) Serpent Day 213.063



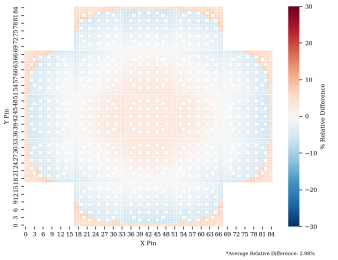
(ap) Relative Difference Day 213.063



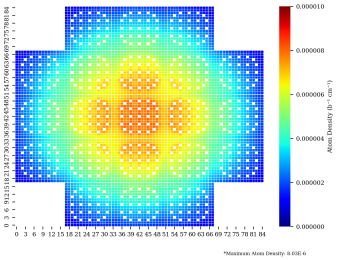
(aq) bRAPID Day 243.5



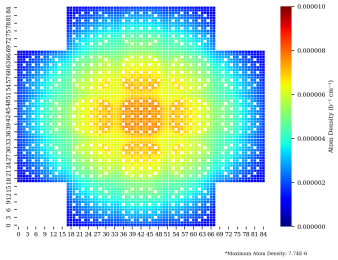
(ar) Serpent Day 243.5



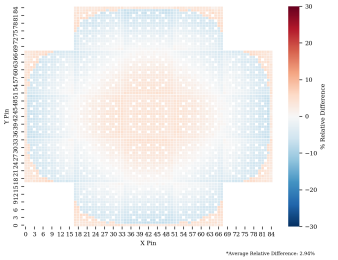
(as) Relative Difference Day 243.5



(at) bRAPID Day 273.938

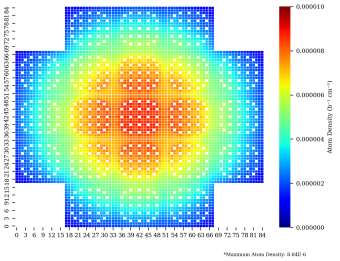


(au) Serpent Day 273.938

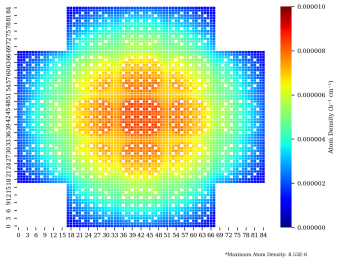


(av) Relative Difference Day 273.938

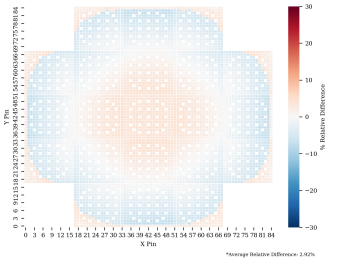
Figure B.3 bRAPID and Serpent pin-wise ^{131}Xe distribution and relative differences (cont.)



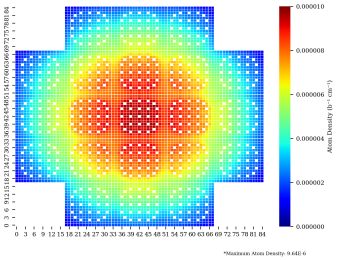
(aw) bRAPID Day 304.376



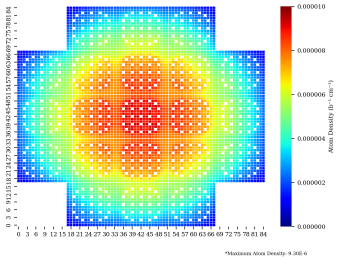
(ax) Serpent Day 304.376



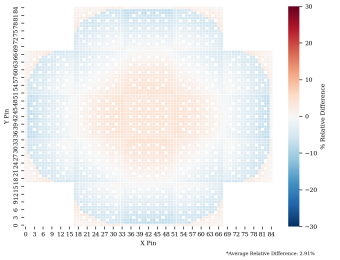
(ay) Relative Difference Day 304.376



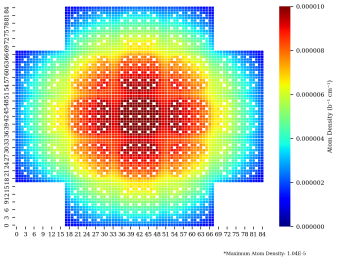
(az) bRAPID Day 334.813



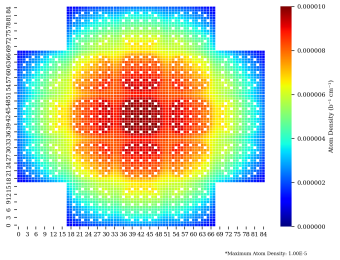
(ba) Serpent Day 334.813



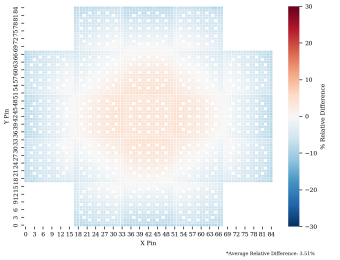
(bb) Relative Difference Day 334.813



(bc) bRAPID Day 365.25



(bd) Serpent Day 365.25



(be) Relative Difference Day 365.25

Figure B.3 bRAPID and Serpent pin-wise ^{131}Xe distribution and relative differences (cont.)

Appendix C

Algorithms and Codes Developed

C.1 Algorithms

The following two algorithms were added to the 2D bRAPID algorithm.

Algorithm 2 Pin-wise Power Density Calculation

Require: F_i^0 ▷ Initialize Fission Source Distribution
Require: $\bar{P}$ ▷ User Defined Average Power Density
Require: N_b ▷ Number of Burnable Regions

Open `gen_bN_fis2d.dat` ▷ N is the current burnup step
    Fill $F_i^0 \rightarrow F_i$ ▷ Pin-wise fission source data

$$f_i = \frac{F_i}{\sum_i F_i}$$

$p_i = (\bar{P} N_b) f_i$
return p_i

Algorithm 3 Time Dependent Boundary Correction

Require: t_bnd = [1,...,N] ▷ Define which steps to use in boundary correction
Require: bnd2xy_sN.dat ▷ Boundary correction file for step N

```
1:
2: if i > 0 then
3:    bnd2xy.dat → bnd2xy_s0.dat          ▷ Preserve the 0^th step boundary correction
4: end if
5:
6: for i ≤ I do
7:    if i in t_bnd then
8:       Replace bnd2xy_s(i).dat → bnd2xy.dat   ▷ Replace previous bnd file with i^th step file
9:    end if
10:   Reload RAPID Database
11: end for
```

C.2 Utility Codes Developed for this Thesis

Table C.1 presents a list of some of the most used utility codes created during this thesis, used for processing data and plotting.

Table C.1 Utility codes developed for thesis

Program Title	Description
`FM_parse.py`	Parses the Serpent output of an FM fixed-source calculation, produces an output which can be used in Teclot 360 for 3D visualization
`fuel_comp_parse.py`	Parses a Serpent depletion output for material composition (atom density) and creates Serpent material definition input file for given fuel regions
`k-effective-parse.py`	Parses Serpent and bRAPID output files for k-effective information, compares them
`fiss_compare.py`	Compares the fission source distribution of a Serpent and RAPID output file; plots the comparison
`ass_def.py`	Creates pin-wise cell definition cards for Serpent pin-wise burnup model
`sensitivity.py`	Parses data and creates plots for sensitivity study of NPS, NSK, and NAC for Serpent (k-effective and fission source dist.)
`pinwise_mat_parse.py`	Parses a pin-wise Serpent and bRAPID burnup model to retrieve burnup and isotopic composition distributions; creates output files for each code system individually
`bnd_corr.py`	Uses Serpent output files to create boundary correction factors for use in bRAPID calculation; also creates heatmap data for boundary correction plotting
`pw_burnup.py`	Uses data from `pinwise_mat_parse.py` to create total composition, burnup, and relative difference plots for pin-wise burnup analysis
`ass_burnup.py`	Similar to pw_burnup.py, but creates assembly-wise plots and data
`bnd_corr_map.R`	Used to plot boundary correction heatmap file
`material_plots.R`	Plots assembly-wise material composition for time step resolution study; calculates relative differences and AARD between data sets

The following libraries for Python and R were also commonly used:

- Python
  - Pandas
  - NumPy
  - Seaborn
  - Matplotlib
  - SciPy

- R
  - ggplot2
  - reshape2
  - Lattice
  - RColorBrewer
  - rasterVis

Bibliography

[1] N. Roskoff. *Development of a Novel Fuel Burnup Methodology and Algorithm in RAPID and its Benchmarking and Automation*. Ph.D. thesis, Virginia Tech (2018).

[2] W. J. Walters, N. J. Roskoff, and A. Haghighat. "The rapid fission matrix approach to reactor core criticality calculations." *Nuclear Science and Engineering*, **192(1)**: pp. 21–39 (2018).

[3] H. Trellue. "Monteburns 2.0." *An automated, multi-step Monte Carlo burnup code system, User's manual version*, **2** (2003).

[4] J. Leppänen. "Serpent–a continuous-energy monte carlo reactor physics burnup calculation code." *VTT Technical Research Centre of Finland*, **4** (2013).

[5] B. T. Rearden and M. A. Jessee. *SCALE Code System. Technical report*, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States) (2018).

[6] K. Manalo *et al.* "Penburn-a 3-d zone-based depletion/burnup solver." (2008).

[7] K. Okumura *et al.* "Validation of a continuous-energy monte carlo burn-up code mvp-burn and its application to analysis of post irradiation experiment." *Journal of Nuclear Science and Technology*, **37(2)**: pp. 128–138 (2000).

[8] J. Cetnar. "General solution of bateman equations for nuclear transmutations." *Annals of Nuclear Energy*, **33(7)**: pp. 640–645 (2006).

[9] M. Pusa and J. Leppänen. "Computing the matrix exponential in burnup calculations." *Nuclear science and engineering*, **164(2)**: pp. 140–150 (2010).

[10] W. Walters. "Application of the rapid fission matrix methodology to 3-d whole-core reactor transport." In: *Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2017)*, (pp. 16–20) (2017).

[11] V. Mascolino, A. Haghighat, and N. J. Roskoff. "Evaluation of rapid for a unf cask benchmark problem." In: *EPJ Web of Conferences*, volume 153, (p. 05025). EDP Sciences (2017).

[12] B. Ebiwonjumi *et al.* "Validation of lattice physics code stream for predicting pressurized water reactor spent nuclear fuel isotopic inventory." *Annals of Nuclear Energy*, **120**: pp. 431–449 (2018).

[13] M. Moriwaki. "Multi-assembly analysis with an advanced correlated sampling method." *Journal of nuclear science and technology*, **40(11)**: pp. 905–917 (2003).

[14] H. Hernandez Noyola. "Pin-wise loading optimization and lattice–to-core coupling for isotopic management in light water reactors." (2010).

[15] Y. Park and M. H. Kim. "Improved retrieval technique of pin-wise composition for spent fuel recycling." (2016).

[16] J.-J. Herrero *et al.* "Performance of whole core pin-by-pin calculations by domain decomposition through alternate dissections in steady state and transient calculations." (2009).

[17] A. T. Daing and M. H. Kim. "Investigation into fuel pin reshuffling options in pwr in-core fuel management for enhancement of efficient use of nuclear fuel." *Nuclear Engineering and Design*, **273**: pp. 332–349 (2014).

[18] G. I. Maldonado. "Optimizing lwr cost of margin one fuel pin at a time." *IEEE transactions on nuclear science*, **52(4)**: pp. 996–1003 (2005).

[19] D. She *et al.* "2d full-core monte carlo pin-by-pin burnup calculations with the rmc code." *Annals of Nuclear Energy*, **64**: pp. 201–205 (2014).

[20] E. E. Lewis and W. F. Miller. "Computational methods of neutron transport." (1984).

[21] A. Haghighat. *Monte Carlo Methods for Particle Transport*. Crc Press (2016).

[22] G. I. Bell and S. Glasstone. *Nuclear reactor theory. Technical report*, US Atomic Energy Commission, Washington, DC (United States) (1970).

[23] J. J. Duderstadt. *Nuclear reactor analysis*. Wiley (1976).

[24] G. Sjoden and A. Haghighat. "Pentran: A three-dimensional scalable transport code with complete phase-space decomposition." *Transactions of the American Nuclear Society*, **74**: pp. 181–183 (1996).

[25] W. J. Walters. *Development of the Adaptive Collision Source Method for Discrete Ordinates Radiation Transport*. Ph.D. thesis, Virginia Tech (2015).

[26] B. G. Petrovic and A. Haghighat. "Analysis of oscillations in the theta-weighted differencing scheme." *Transactions of the American Nuclear Society*, **71(CONF-941102-)** (1994).

[27] B. Petrovic and A. Haghighat. "New directional theta-weighted (dtw) differencing scheme and reduction of estimated pressure vessel fluence uncertainty." In: *Proceedings of the 9th International Symposium on Reactor Dosimetry, Prag*, (pp. 746–753) (1996).

[28] C. Lee and W. Yang. *MC2-3: multigroup cross section generation code for fast reactor analysis. Technical report*, Argonne National Lab.(ANL), Argonne, IL (United States) (2013).

[29] N. Greene *et al. AMPX: A modular code system for generating coupled multigroup neutron-gamma libraries from ENDF/B. Technical report*, Oak Ridge National Lab., Tenn.(USA) (1976).

[30] T. Yamamoto, T. Nakamura, and Y. Miyoshi. "Fission source convergence of monte carlo criticality calculations in weakly coupled fissile arrays." *Journal of nuclear science and technology*, **37(1)**: pp. 41–52 (2000).

[31] T. E. Booth and J. E. Gubernatis. *Improved criticality convergence via a modified Monte Carlo power iteration method. Technical report*, Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

[32] B. Shi and B. Petrovic. "Implementation of the modified power iteration method to two-group monte carlo eigenvalue problems." *Annals of Nuclear Energy*, **38(4)**: pp. 781–787 (2011).

[33] E. Dumonteil and T. Courau. "Dominance ratio assessment and monte carlo criticality simulations: dealing with high dominance ratio systems." *Nuclear Technology*, **172(2)**: pp. 120–131 (2010).

[34] W. Walters, N. Roskoff, and A. Haghighat. "A fission matrix approach to calculate pin-wise 3d fission density distribution." *Proc. M&C* (2015).

[35] N. Roskoff and W. Walters. *RAPID User Manual*. Virginia Tech Transport Theory Group , Penn State University.

[36] J. Leppänen and M. Pusa. "Burnup calculation capability in the psg2/serpent monte carlo reactor physics code." *Proc. M&C*, (pp. 3–7) (2009).

[37] X.-. M. C. Team. "Mcnp—a general monte carlo n-particle transport code, version 5." (2003).

[38] N. Roskoff. *pRAPID: Pre- and Post-processing for RAPID User Manual*. Virginia Tech Transport Theory Group.

[39] J. E. Hoogenboom, W. R. Martin, and B. Petrovic. "The monte carlo performance benchmark test-aims, specifications and first results." In: *International Conference on Mathematics and Computational Methods Applied to*, volume 2, (p. 15) (2011).

[40] A. Moghrabi and D. Novog. "Determination of the optimal few-energy group structure for the canadian super critical water-cooled reactor." *Annals of Nuclear Energy*, **115**: pp. 27–38 (2018).