DEVELOPMENT OF SERPENT-BASED METHODS FOR I2S-LWR DEPLETION MODELING AND SENSITIVITY STUDIES

A Thesis
Presented to
The Academic Faculty

By

Kyle Michael Ramey

In Partial Fulfillment
Of the Requirements for the Degree
Masters of Science in Nuclear and Radiological Engineering

Georgia Institute of Technology
May 2017

DEVELOPMENT OF SERPENT-BASED METHODS FOR I2S-LWR DEPLETION MODELING AND SENSITIVITY STUDIES

Approved by:

Dr. Bojan Petrovic, Advisor
Nuclear and Radiological Engineering Program
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Nolan E. Hertel
Nuclear and Radiological Engineering Program
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Dingkang Zhang
Nuclear and Radiological Engineering Program
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: December 6, 2016

ACKNOWLEDGEMENTS

My experience at Georgia Tech has been a very enriching part of my education, from both an academic and life lesson standpoint. While I am confident that this “Florida Boy” could not have possibly braved the elements of more northerly academic institutions, Georgia Tech and Atlanta have proved to be a very good fit academically for my graduate studies. The person most responsible for this is my academic advisor Dr. Bojan Petrovic, whose guidance and support has greatly broadened and deepened my knowledge in nuclear engineering. He is likely the hardest working person I know and definitely one of the smartest people I will ever meet. His leadership and interactions in academia as well as collaborative work with industry serve as examples every day for what it truly means to be a professional in our field. This thesis would not have been possible without him and I am greatly appreciative for all the discussions we have had along the way toward the compilation of this document. He is a great academic role model and I hope that one day, I too can do reactor calculations in my head and still be almost as witty as him.

I would like to recognize my other thesis committee members Dr. Nolan Hertel and Dr. Dingkang Zhang for taking the time from their busy schedules to evaluate this body of work. Their willingness to participate is of great value and further illustrates the benefit of being part of the Georgia Tech community.

David Salazar of Westinghouse Electric Company provided invaluable assistance by simulating some of my models in Westinghouse’s reactor physics package. He was always very quick to provide feedback and went out of his way to help me whenever asked.

Although not emphasized enough in the document, David greatly shaped the direction of my design work and led me down paths I would not have considered without him. I am grateful that he made his industry knowledge so openly available and as a result has made me see the value of collaborative work and seeking second opinions from colleagues.

The members of Dr. Petrovic's research group have all been good friends along the way through the process of conducting this research. Sometimes on the rare occasion our discussions would even be relevant to my research. Some of their own research endeavors related to the I²S-LWR are referenced in this document and highlight how our seemingly distinct works are still able to mesh together. Notably from the lab, I would like to thank Tim Flaspoehler, Matt Lynch, and Tim Wyant. Tim Flaspoehler assisted me countless times when running Serpent simulations on the group's computer cluster and I hope one day I can be the remote computing guru that he is. Matt Lynch assisted to review this document was always there to share a joke. Although Tim Wyant's time with the lab predated my own; his Master's Thesis, also involving studies done using Serpent, has been of great relevance to my own thesis.

Last but certainly not least, I am deeply grateful for my mother Helen's support through my whole academic career. She has always been my biggest advocate and believed in me even when I did not. From a young age, she taught me self-reliance and always encouraged me to aim higher. Although she is hundreds of miles away, her unconditional love has been felt every step of the way.

This research was performed through the help of funding received from the DOE Office of Nuclear Energy's Nuclear Energy University Programs (NEUP).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two-Dimensional
3D	Three-Dimensional
AO	Axial Offset
ATF	Accident Tolerant Fuel
BOC	Beginning of Cycle
BWR	Boiling Water Reactor
CBC	Critical Boron Concentration
CVCS	Chemical and Volume Control System
EFPD	Effective Full Power Days
EOC	End of Cycle
FeCrAl	Iron Chromium Aluminum – referring to the major constituent metals in a particular group of stainless steel alloys.
F_q	Total hot channel factor
F_Z	Axial hot channel factor
$F_{\Delta H}$	Enthalpy-rise hot channel factor
GT	Georgia Institute of Technology (also Guide Tube); the meaning is always clear from the context
HFP	Hot Full Power (normal operation)
HM	Heavy Metal (usually used with kgHM, or kilograms of heavy metal)
HZP	Hot Zero Power
I^2S-LWR	Integral Inherently Safe Light Water Reactor
IFBA	Integral Fuel Burnable Absorber
k	Eigenvalue or Multiplication Factor (unless otherwise specified, the effective eigenvalue, k_{eff})
LOFA	Loss of Flow Accident
LWR	Light Water Reactor
MC	Monte Carlo
PWR	Pressurized Water Reactor
SBO	Station Blackout
SCA	Single Channel Analysis
SMR	Small Modular Reactor
US	United States (of America)
WEC	Westinghouse Electric Company

SUMMARY

The I²S-LWR is an innovative nuclear power plant design with the power level and size of conventional large LWRs. While other new reactor designs use exotic coolants or are very small in size with unconfirmed claims of scalability, the I²S-LWR design promises safer operating and shutdown features in a package with proven economic power level.

The research in this thesis analyzes the I²S-LWR core design using 2D and 3D Serpent models. Work completed using the 2D model is used to verify the accuracy of uncertainties reported by Serpent and provide guidance for moving forward with the 3D design. Serpent-reported uncertainties are compared to observed uncertainties from replica runs and also symmetric groupings. In moving toward creating a 3D model, the 2D model results are compared with an industry code. Knowledge derived from these endeavors is implemented in the 3D model through results and observations.

The 3D model and depletion methodology is the principal focus of this thesis. It is described in detail and its depletion of a first core using a candidate core loading pattern is evaluated based on power performance. The depletion analysis uses temperature feedback for the fuel, IFBA, and coolant as well as density variation for the coolant. For a design using only four fuel enrichments, the achieved power peaking factors are better than expected at the quarter assembly level: $F_{\Delta H}$ stays below 1.25, F_Z stays below 1.45, and F_q stays below 1.7 for the entire fuel cycle. While the equilibrium core design for this study has a target average burnup of 348 EFPD, this first core operates for a slightly longer period of time (438 EFPD or about 15 months) for the economic reason of increasing the discharge burnup of the fuel discharged after the first cycle.

CHAPTER 1 INTRODUCTION

1.1 Motivation

After the events which occurred at the Fukushima Daiichi nuclear power plants in Japan in March 2011, there has been a strong push in the nuclear power industry to engineer passive safety features into the next generation of reactors. Most of these new designs feature accident-tolerant fuels[1] (less prone to melting during an accident scenario) and accident-tolerant claddings (continued robust performance under accident conditions while eliminating or significantly reducing hydrogen gas production currently challenging traditional zirconium alloy claddings), along with passive heat removal systems. One such reactor design is the Integral Inherently Safe Light Water Reactor (I^2S-LWR), which uses a uranium silicide fuel form (U_3Si_2) and APMT[2] advanced stainless steel (a type of FeCrAl steel) for cladding and other in-core structures. This thesis focuses on developing a methodology and analyzing important neutronic behaviors of the I^2S-LWR core geometry and design parameters.

1.2 Objective

As a non-standard design, I^2S-LWR models need to be accurately created to capture physical behavior since there are limited results dealing with U_3Si_2-APMT fuel. Monte

[1] Accident Tolerant Fuel (ATF) normally refers to both the fuel and cladding as a pair, since their combined performance has a substantial impact on how a core acts both during and beyond normal operation. Here, the fuel is specified by itself. Distinctions in the remainder of the paper will be made for when referring to ATF as the fuel-cladding pair or when specifically addressing fuel alone.

[2] APMT refers to a specific iron-chromium-aluminum (FeCrAl) stainless steel alloy which is composed (by weight) of: Fe 69.82%, Cr 21%, Al 5%, Mo 3%, Si 0.7%, Mn 0.4%, and C 0.08% [14].

Carlo is used to accomplish this, since when given the computational resources, it can produce accurate results in most general systems. Specific analyses on the I²S-LWR core include 2D radial reflector studies, 3D quarter-assembly level depletion, uncertainty analysis, and others.

1.3 Scope

All Monte Carlo simulations conducted in this thesis were carried out using Serpent. The initial studies were conducted on a 2D Serpent model. It was believed that this was adequate to capture the radial behavior of the I²S-LWR system. More detailed analyses requiring accurate axial behavior used a 3D Serpent model. It was used primarily for a first core depletion study of a core loading pattern created for this thesis (that is, differing from the actual first core loading pattern of the I²S-LWR design).

In both Serpent models, accuracy was the focus when creating the geometry. However, for features outside the core, simplications were made in the interest of reducing the model size to increase performance. An example of this is both the axial and radial reflectors, which are homogenized between APMT steel and water.

This thesis analyzes the depletion of a fresh core, i.e. it is beyond the scope to quantitatively consider additional core loadings on an approach toward an equilibrium cycle. While a worthwhile effort in future work for more seriously considered core loading patterns; this first cycle depletion demonstrates applicability toward future I²S-LWR studies using the Serpent models and the developed methodology expressed in this thesis.

1.4 Document Layout

Chapter 2 focuses on background pertinent to the analyses of later chapters. This includes neutronics, Monte Carlo, and more details about the I^{2}S-LWR. Chapter 3 introduces the 2D Serpent Model used for preliminary studies while Chapter 4 provides the results of those studies. Chapter 5 introduces the 3D Serpent Model used for the first core depletion analysis and the methodology implemented to account for the thermal feedback. Chapter 6 contains a majority of the results for that analysis.

CHAPTER 2 BACKGROUND

2.1 Neutron Transport

The most essential component to analyzing any nuclear reactor core is tracking the interactions and movement of neutrons in the core region, or neutron transport. As neutrons are born from fission and interact in the core, the material composition of the core changes with time. Fortunately, being able to calculate the distribution of neutrons at one point in time will enable future distributions to be predicted as well, making possible the design of a fueling scheme that utilizes the same fuel assemblies for years. While following this time evolution cannot be solved exactly, several techniques exist which simplify the neutron transport equation to arrive at a still very accurate approximation.

The neutron transport equation governs the distributions of neutrons in seven dimensions: three in space (location), three in velocity (alternatively, one in energy and two in angle of travel), and time. The distribution is found by knowing the five gain/loss mechanisms for neutrons in a medium: the transient term (time dependence), net streaming (movement into and out of the region), removal by interacting, neutrons introduced by fission and scattering events, and independent sources (Equation 1) [1].

$$\begin{aligned}&\frac{\partial n}{\partial t}+v\hat{\Omega}\cdot\nabla n+v\Sigma_t\left(r,E\right)n\left(\vec{r},E,\hat{\Omega},t\right)=\\&\int_{4\pi}d\hat{\Omega}'\int_0^\infty dE'v'\Sigma_s\left(E',\hat{\Omega}'\rightarrow E,\hat{\Omega}\right)n\left(\vec{r},E',\hat{\Omega}',t\right)+s\left(\vec{r},E,\hat{\Omega},t\right)\end{aligned}\qquad\text{Equation 1}$$

However, the density of neutrons is sometimes not the most desirable parameter for expressing the distribution. The neutron transport equation can instead be rewritten in terms of the angular flux (ψ). Additionally, for systems without independent sources, criticality is usually discussed in terms of an eigenvalue k. These together are used to obtain the more relevant Equation 2 [1].

$$\begin{aligned}
&\frac{1}{v}\frac{\partial\psi}{\partial t}+\hat{\Omega}\cdot\nabla\psi+\Sigma_t\left(\bar{r},E\right)\psi\left(\bar{r},E,\hat{\Omega},t\right)= \\
&\int_{4\pi}d\hat{\Omega}'\int_0^\infty dE'\Sigma_s\left(E',\hat{\Omega}'\rightarrow E,\hat{\Omega}\right)\psi\left(\bar{r},E',\hat{\Omega}',t\right) \\
&+\frac{X(E)}{k}\int_{4\pi}d\hat{\Omega}'\int_0^\infty dE'\nu\Sigma_f\left(\bar{r},E\right)\psi\left(\bar{r},E',\hat{\Omega}',t\right)
\end{aligned} \qquad \text{Equation 2}$$

Although the neutron transport equation exactly describes the distribution of neutrons at each location and point in time, it is too difficult to explicitly solve, even with modern technology and computing capabilities. Instead, certain simplifying assumptions can be made in order to obtain several different types of approximations of varying degrees of accuracy. In the early days of nuclear core analysis and even up to this day, one of the most widespread and simple-to-implement approximations to use is diffusion theory. While some industry codes still use some aspects of diffusion as a quick way to obtain results, it is avoided in cases where accuracy is of concern since diffusion performs poorly in regions with large neutron gradients (near strongly absorbing materials, near neutron sources, near surfaces and boundaries with very different behaviors, and where scattering is strongly anisotropic) [2].

Another approximation to the neutron transport equation includes making the equation tractable for computers by reducing the single difficult equation into manysimpler ones through a process call discretization. Instead of treating the seven variables continuously, some or all can be discretized to create systems of equations. The benefit is that this approach can be implemented quite effectively with modern computing, while at the expense of losing resolution within the problem by the degree of coarseness. The highest fidelity is maintained by the Monte Carlo (MC) method, which requires no or minimum approximations, and will be discussed next.

2.2 Monte Carlo

While some other approximations to the neutron transport equation discretize the variables in order to find the continuously-treated neutron distribution, the Monte Carlo method essentially maintains the continuous nature of the variables and instead discretizes the solution. In MC, neutrons are randomly sampled from sources (fissionable or independent) and are tracked using statistically-based physics until they are removed from the problem[3]. As the number of particles used increases, the solution should become more precisesince the level of statistical uncertainty is reduced. As with any method, MC has a few additional caveats which will be addressed below.

2.2.1 Advantages and Drawbacks

2.2.1.1 Accuracy of Models

The main reason why there has been so much interest in using the Monte Carlo method for the past few decades is its accuracy in modeling complex systems [3]. While spatial discretization usually cannot accurately represent complex shapes, energy discretization loses some behaviors by lumping a spectrum of energies together, and angular discretization can neglect ray effects; MC is able to replicate a problem's geometry and physics to a very high degree. With enough generalized surfaces to be used in models

[3] A vast majority of removals are due to either absorption or leaving the problem boundary. Rarely, having parameters beyond a specified threshold (such as having too low of energy) can also result in the code regarding a particle as being removed. Similarly, a particle can also be treated as being removed if it has not leaked or been absorbed after experiencing a set number of collisions.

to replicate virtually anything physical, on paper MC seem only held back by the accuracy of its physics subroutines and the cross sections used for its interaction probabilities.

2.2.1.2 Stochastic Uncertainties

Since MC relies on using randomly-based events in its physics routines, no single particle is able to come remotely close to being representative of even very simple problems. For this reason, MC needs to use a large number[4] of sample particles approximate the parameters of a problem. Even with a large sample size, there will always be some statistical uncertainty in the results, because the simulated ratios of one event happening to another event will never exactly match reality. However, by simulating a very large number of particle histories, the statistical uncertainty may be reduced to a level which eventually may become acceptable.

After the discussion of increasing the number of particles used in order to decrease the statistical uncertainty, the following question is begged, “how many more particles need to be run to achieve a certain amount of improvement?” This question has been asked many times and the result is well known. To arrive at the result, we first need to look at the definitions of sample mean (Equation 3) and sample standard deviation (Equation 4).

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{Equation 3}$$

$$\hat{\sigma}_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\overline{x} - x_i\right)^2} \qquad \text{Equation 4}$$

[4] For the global parameters of a very simple problem, only millionss of sampled particles may be sufficient. To accurately measure local parameters of a complex problem, billions or more total particles may be necessary. Accuracy in Monte Carlo is relative and highly dependent upon the complexity of the model and the parameters of interest.

From Equation 4, it can be seen that Equation 5 is just a consequence of the definition of standard deviation when N is large.

$$\hat{\sigma}_x \propto \frac{1}{\sqrt{N}} \qquad \text{Equation 5}$$

The significance of this result is that as N grows (for a MC simulation, N would be the number of particle histories followed), the standard deviation decreases as $1/\sqrt{N}$ [3]. So for example, to achieve an order of magnitude of improvement in the stochastic uncertainty of a MC model, 100 times more particles would need to be used. Further, since the time needed to complete a MC simulation is roughly linear for large N, one could expect that this improvement would require 100 times more computational resources or runtime. This illustrates that although MC can get an accurate result with great precision, it could require tremendous computational effort relative to one with less precision.

2.2.1.3 Source Convergence

Although MC is valued for being so versatile in its application, another downside in how it is applied is that there is no prior knowledge of the fission source distribution, and accurately converging on fission source distributions can be extremely time-intensive if only brute force is used[5]. In the very common case of modeling a reactor core, a good starting point for the neutron distribution is to first assume it is uniform in the geometry. One quickly discovers that the neutrons only originate in the fuel materials and that each fuel region contributes differently to the distribution. An example exotic case where MC would likely fail was suggested by Whitesides in a publication in 1971, where he proposed

[5] There exist many methods for handling exotic distributions in Monte Carlo, such as source biasing for sources far from the regions of interest, but these will not be addressed in this thesis.

how a MC code would compute the eigenvalue of the world [4]. While the result should be unity due to all reactors being critical, the MC code would likely never actually sample within a critical region to obtain this result. Thus, since the simulation would not converge upon the source, the code would likely conclude the world is subcritical and would not yield the correct answer.

So as not to let the initial “blind guess” at the neutron distribution skew the actual results, sample neutrons in MC are run in generations— group of neutrons still large enough to provide some resolution on the neutron distribution of the problem, but each generation only making up a small fraction of the total number of particles used in the model. Each generation uses the source history of the previous generation for sourcing its own particles, which interact and end up becoming the source for the next generation.

While the process of converging on the neutron source distribution is happening, the results leading up to source convergence cannot be used and should be discarded. The generations up to that point are called inactive cycles and the generations after are called active cycles. The evolution over successive generations from the initial guess to the correct distribution depends on the geometry. Simple cases such as mirrored pins or small assemblies could require only a few dozen inactive cycles, whereas full cores could take thousands.

2.2.1.4 Intercycle Source Dependence

A final challenge for interpreting MC results comes from the coupling of stochastic uncertainty and source convergence. When a MC code expresses the statistical uncertainty of a parameter, it normally does so as the sample standard deviation (Equation 4) of that parameter over all active cycles. However, this step assumes that the result is correct given that the sample selection was independent, but from the source convergence discussion, generations in a MC run obtain their source distribution from the previous generation, i.e., there is intergenerational correlation.

By the incorrect assumption that generations are independent, the stochastic uncertainty estimates provided by MC codes are biased [5] and can underestimate the uncertainty, typically by up to a factor of five [6] [7]. The consequence of this is that most of the parameters (eigenvalue, individual pin power contributions, etc.) output by a MC simulation have smaller reported variances than their true values [8]. While many artificial remedies have been developed [7] [8] to correct this underestimation, it is still an issue present in almost all MC codes used today. The effect can even be greatly exacerbated in situations where the source has not converged well. Local tallies (such as pin powers) in a case with poor convergence can have uncertainties underpredicted by more than a factor of 40 [7].

2.2.2 Serpent

Serpent is a three-dimensional (3D) continuous-energy Monte Carlo reactor physics burnup calculation code [9]. As a MC code, using Serpent entails having the advantages and disadvantages addressed previously. The differentiating characteristic of Serpent from other MC codes is that it was developed specifically with depletion studies in mind. The focus on burnup calculation capabilities during the early stages of creation resulted in efficient runtimes and good scalability with conventional high-performance parallel computing. Serpent was selected as the reactor physics analysis tool for the work summarized in this thesis since it was known from the outset that depletion studies would be necessary. In retrospect, this is seen as a good decision because Serpent was both straightforward to implement and was able to provide the desired results for the work being done.

Like most other MC codes, the current publicly distributed version of Serpent does not account for thermal feedback. Although work is currently being conducted toward implementing multi-physics in Serpent [10], is it not yet available in the code package. Therefore, if thermal hydraulic coupling is desired with the neutronics, some means external to Serpent will need to be used.

2.3 I^2S-LWR

The I^2S-LWR design is the focus of the work expressed in this thesis. The two main objectives of the I^2S-LWR design are to incorporate many of the safety features associated with ATFs, while at the same time producing a large amount of power on a scale comparable to the current fleet of LWRs [11]. Both before and especially after the accident at the Fukushima-Daiichi nuclear power plants in Japan in March 2011, many of the other newer reactor designs in the field are Small Modular Reactors (SMRs).

Most SMR designs implement some aspects of ATF, like I^2S-LWR does, in the reactor core. SMRs are designed to be flexible in power production capabilities since multiple units can be installed at a single site to match the current power demand and then scaled to a larger fleet in the future by adding additional units at the site. Although many questions remain outstanding as to the actual licensing and economic scalability of SMRs [12], I^2S-LWR implements the safety features of new SMRs while retaining the large output from a single reactor with which industry is accustomed.

2.3.1 Reactor Core

The I^2S-LWR core has 121 19x19 fuel assemblies, which in total produce 2850 MW_t. In each of these assemblies, there are 336 fuel rods, 24 guide tubes, and a single instrumentation tube in the center (Figure 2.1 [13]).

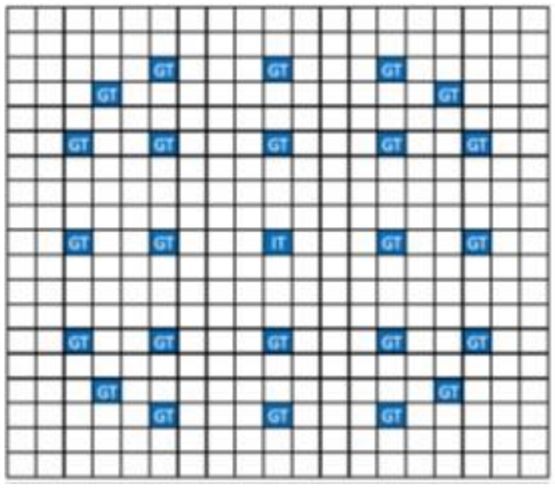


Figure 2.1 Locations of the single instrumentation tube (IT) and 24 guide tubes (GT) in the 19x19 fuel assembly.

The core is surrounded by a stainless steel radial reflector with coolant flow channels, which are smaller and closer to each other next to the core and get larger moving away from the core. Beyond the radial reflector is a solid stainless steel core barrel (effectively a continuation of the radial reflector without cooling channels) and then finally past the core barrel is a coolant downcomer region.

2.3.2 ATF Considerations

In light of the recent push for the industry to seek alternative fuels for conventional UO_2-zirconium alloy fuel, the I^2S-LWR uses uranium silicide instead of uranium dioxide for fuel (U_3Si_2 vs UO_2) and high-performance stainless steel (specifically APMT) instead of zirconium alloy for cladding. Each of the constituent materials was selected to have more favorable performance during an accident; most notably a Loss Of Flow Accident (LOFA) during a station black out. Since U_3Si_2 has higher thermal conductivity than UO_2 under both operating and accident conditions [14], the I^2S-LWR fuel will be at a lower temperature than conventional fuels since it is more effective at removing heat. By combining the lower temperature behavior under accident conditions of the fuel with the oxidation resistance of the APMT cladding, the I^2S-LWR core should be less prone to hydrogen gas production[6] and the ensuing consequences of hydrogen gas explosions potentially resulting in containment failure.

2.4 Neutron Economy

In a nuclear reactor, there are many engineered mechanisms that exist to control the population of neutrons. These controls must be very finely tuned to be able to maintain the system in a delayed-critical state. Too few neutrons would lead to the reactor shutting down, because there would be a decreasing number of neutrons in each generation causing the reaction to essentially fizzle out. Too many could result in catastrophic failure due to a prompt criticality accident whereby the fission reaction rate grows uncontrollably. Core designs focus on safely operating between these two thresholds to maintain a critical system during the course of the fuel cycle [2].

[6] Passive decay heat removal features have also been engineered into the I^2S-LWR design to avoid core damage during a LOFA under SBO. By providing passive cooling to the core, the fuel and cladding should not reach critical temperatures for melting or oxidizing. Thus, the failure mode experienced at the Fukushima-Daiichi plants has essentially been engineered out of the I^2S-LWR design.

When a conventional (nonbreeding) reactor core is being operated, the fissile fuel sustaining the fission chain is slowly consumed[7] (or burned up) by each fission reaction. Over time, progressively less fuel remains to keep the reactor operating in a critical state[8]. For this reason, reactors need to be loaded with more fissile fuel at the beginning of the fuel cycle than required to be critical to accommodate the depletion of fuel. To control this excess reactivity, several features exist for a reactor to both stay below prompt criticality and maintain a critical fission chain for several months. The methods used in conventional Light Water Reactors (LWRs) are discussed in the rest of this Chapter.

2.4.1 Control Rods

The purpose of control rods is to maintain criticality and adjust the power level during operation and to keep the reactor subcritical when shutdown. During the operation of some reactor types, such as Boiling Water Reactors (BWRs), control rods can also be used for power shaping. Since the steam at the top of the core moderates less effectively than the liquid water toward the bottom, there would normally be a strong Axial Offset (AO) in the power profile toward the bottom. However, by having partially inserted control rods[9] from the bottom, the negative reactivity helps to counteract the AO and push it closer to the midplane.

[7] No matter the design of a uranium-fuel core, there will inevitably be some fuel production through neutron captures in ^{238}U (a fertile isotope), which transmutes into ^{239}U and is then able to eventually decay into ^{239}Pu (a fissile isotope). So even though fissile fuel is produced by operating a conventional LWR, there is net loss of fissile material since the amount of fission-related losses of fissile material is greater than the amount of fissile material gain due to absorptions and subsequent transmutations into fertile materials.
[8] In addition to each fission reaction consuming a fissile atom and contributing to the depletion of fuel, the resulting fission products can also negatively impact the neutron economy of the reactor. Several fission product isotopes have large absorption cross sections, which introduce additional detriment to the neutron population beyond loss of fuel. These fission product poisons have been studied in great detail since the very early days of nuclear power but will not be discussed further in this section of the thesis.
[9] More accurately, BWRs use control blades as opposed to control rods. The blades are cruciform-shaped and designed to be inserted in the center of a group of four fuel assemblies. However, since this thesis focuses on a PWR system, expanding the discussion to include control blades was foregone.

Unlike in other reactor designs where the control rods are moved through the core during operation; Pressurized Water Reactors (PWRs) operate with control rods essentially withdrawn— control rods are only inserted during core shutdown and some operational transients. Since control rods are not used during normal operation, which was the scope of this analysis, they have not been considered in this analysis.

2.4.2 Chemical Shim

Whereas BWRs use control rods to control reactivity and maintain criticality, PWRs accomplish the same by using boric acid (H_3BO_3) chemical shim. Boron acts as a strong neutron poison, specifically due to the very large thermal absorption cross section of ^{10}B[10]; so increasing the concentration of boric acid in the coolant of PWRs reduces the reactivity of the system. By using the precise concentration needed to maintain criticality, reactor operators are able to control the fission reaction without physically moving any core components.

Over the course of a fuel cycle, the core goes from having very high excess reactivity at the Beginning of Cycle (BOC) to being just barely able to maintain criticality at End of Cycle (EOC). As reactivity is lost due to the depletion of fuel and the build-in of fission product poisons., the critical boron concentration (CBC) of the coolant generally decreases as well over time[11]. This phenomenon is called boron letdown, since the process is gradual over the fuel cycle[12].

[10] ^{10}B accounts for about 19.9% of natural boron with the remainder being ^{11}B, which does not have any remarkable neutronics properties.
[11] Brief periods of increase near the beginning of the cycle are also common. These typically occur after the fission products have been given enough time to reach equilibrium. They are caused by the integral burnable absorber materials burning at a faster rate than the fuel, resulting in a higher reactivity and thus requiring a high CBC.
[12] Sharp drops in CBC are observed during startup as fission product poisons approach equilibrium concentration.

2.4.3 Burnable Poisons

An additional method used to compensate for excess reactivity in a reactor core is to use burnable poisons. While chemical shim in a PWR system is very effective for keeping the entire system critical, it cannot be controlled locally in the core. Shim cannot be used for radial power shaping or controlling the reactivity of only certain regions since essentially the same concentration of boron[13] can be found everywhere in the coolant. For more local control of reactivity, burnable poisons are physically placed in the fuel assemblies. Most of these poisons utilize either boron or gadolinium as the neutron absorber. As the name suggests, burnable poisons are intended to only absorb neutrons for a finite amount of time. As the poisons absorb neutrons during normal operation, they "burn out" and have much less of an impact in the core toward the EOC as they become depleted.

Burnable neutron absorbers can come in many physical manifestations. Some of the first designs had entire rods where instead of fuel a strong absorbing material for power suppression was used, such as: borated glass, Al_2O_3/B_4C, and Gd_2O_3. These discrete rods performed acceptably, but a push for improved fuel performance in the 1980s eventually led to the use of an Integral Fuel Burnable Absorber (IFBA)— a small amount of a strong neutron absorbing material applied to the fuel pellets. The most commonly used design is a thin ZrB_2 layer coating the outside of the current fleet of UO_2 fuel. An example of how IFBA is applied to fuel can be seen in Figure 2.2[14] [15].

[13] The same concentration of boron (neglecting small depletion of boron moving up through the core) can be found per unit mass of coolant. However, as the coolant water gets heated moving up the core, its density decreases and thus so does the concentration of boron per unit volume of coolant. The boron density is only held constant per unit mass of the coolant.

[14] This happens to be the same geometry of how IFBA is applied to fuel in the I2S-LWR. However, the ^{10}B loading differs (I²S-LWR uses only 2.5 mg/in but utilizes three different application patterns).

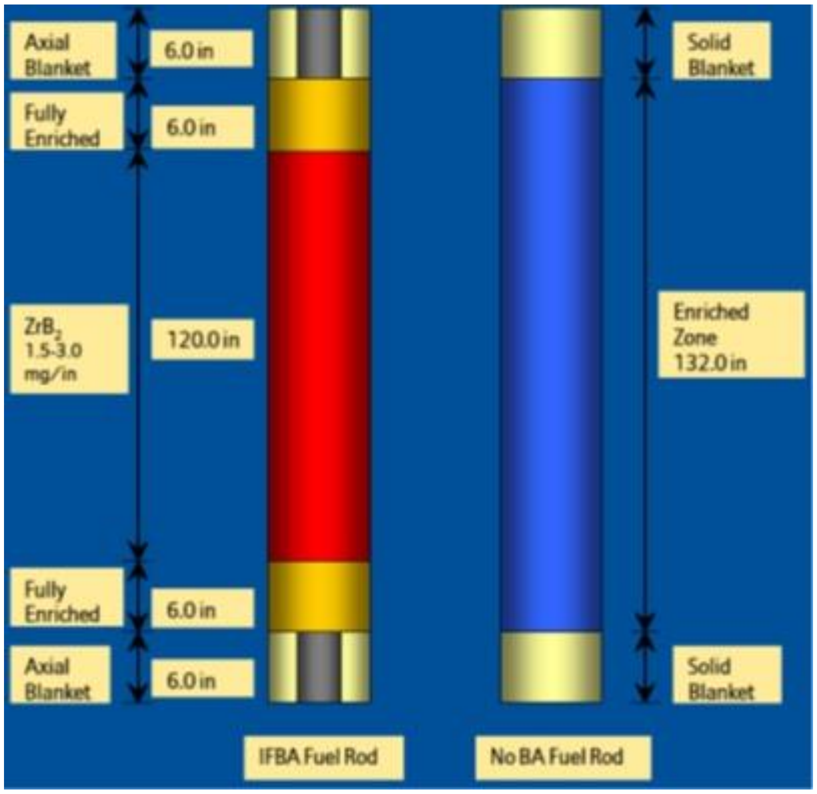


Figure 2.2 IFBA application to a fuel rod.

As a neutron absorber, boron is advantageous over other materials because it has only two naturally-occurring isotopes. ^{10}B is the strongly absorbing isotope and the remaining isotope (^{11}B) has a low absorption cross section, so it does not interact very much beyond scattering. The problem with other elements used as burnable poisons (such as gadolinium) is that they have many more naturally-occurring isotopes. Some of these have the desired characteristic of having larger neutron absorption cross sections, but other isotopes have smaller ones but are still able to transmute to the more parasitic isotopes during residency in the reactor via an absorption event. As mentioned earlier, a good burnable poison should be very impactful at the BOC and much less so at EOC. The problem with transmutation during the cycle is that these parasitic isotopes are never fully depleted since they are also created from the less reactive isotopes, meaning that there can be a significant neutronics penalty from these burnable poisons at EOC [16]. To compensate for this, fuel enrichment might need to be increased to reach the desired cycle length— a potentially costly solution.

Despite this advantage over other neutron absorbers, boron also has a major drawback. As a byproduct of the ^{10}B-neutron absorption reaction, helium is produced. By using a boron-containing IFBA, this helium is released in the fuel rod during operation. Along with the fission product gases released from the fuel during burnup, a significant amount of pressure could be created inside the fuel rod cladding [16]. Thus, the helium gas production needs to be taken into account[15] when designing the fuel so that the cladding does not rupture due to high internal pressure.

[15] A study of fission product gas and helium releases in the fuel of the I^{2}S-LWR has already been completed as part of another student's Master's Thesis work [23].

CHAPTER 3 2D I²S-LWR SERPENT MODEL

3.1 Overview

This model was developed to preform preliminary studies on the I²S-LWR core as well as serve as a hands-on example for learning the input syntax required to run Serpent. In the beginning, the US base distribution version of Serpent (1.1.7) was used for runs. As work became more detailed and advanced, work transitioned to using a newer version of Serpent (2.1.23). Although the 2D model described here was used to carry-out a few studies, the intention from the beginning was to eventually progress to using a 3D model to capture axial effects of the I²S-LWR core. Discussion of this 3D model is provided in Chapter 5.

3.1.1 Objective

This 2D Serpent model is intended to investigate the reactor core design and fuel cycle of the I²S-LWR. As a 2D model, axial effects are ignored. Instead, this model still allows for analyses of radial core behavior; namely $F_{\Delta H}$, radial power profiles, and radial reflector reactivity impact.

3.1.2 Scope

Most aspects of the 2D Serpent model were selected based on the following two metrics: how impactful the design element is to the model and how feasible it is to accurately include the element. Since the in-core geometry and other associated features with the fuel are extremely important for neutronics behavior, they are modeled in a detailed manner. Beyond the core, simplifications are made since increasing the fidelity of the representation further would not change the results too significantly for a model already limited in accuracy by being 2D. Most of these simplifications are addressed below and again more technically in the model description.

3.1.3 Approach

The 2D Serpent model radial geometry is meant to be captured accurately in the core: matching fuel assembly lattice and pin dimensions. The geometry tries to capture other reactor core design features as accurately as possible, but certain simplifying assumptions were made in the creation of the 2D Serpent model. Reactivity control mechanisms (namely IFBA and soluble boron) are to be modeled in the 2D core; although their accuracy will not be very high since IFBA usage and soluble boron volume density both vary axially. IFBA is modeled as the thickness and density found in the largest, central section of the core; no averaging is done to account for the lack of IFBA at the top and bottom sections of the active fuel.

As a 2D Serpent model neglecting axial variations, some work had to be done to at least come close to capturing the axial properties of the core. For most material compositions and temperatures used in the 2D model, the axial average is used for both simplicity and an attempt at accuracy. Averaging is also used to account for discrete axial features, like spacer grids. Since modeling the grids at multiple axial locations is not possible in a 2D model, they are homogenized through averaging by volume with the coolant for simplicity.

3.2 Model

3.2.1 Geometry

The exact dimensions used in the 2D Serpent model can be found in Table 3.1. The IFBA coating thickness was selected based on assuming a ZrB_2 density of 6.08 g/cm^3, requiring the ^{10}B content to be 2.5 mg/in, and assuming that the boron is enriched to 60 a% ^{10}B. When comparing this model to other I^{2}S-LWR models, note that modifying any of these assumptions will impact the IFBA thickness. The important feature to keep constant is the 2.5 mg/in ^{10}B content, but being aware of the other parameters is also relevant.

Table 3.1 Geometry parameters for solid fuel form

Quantity [unit]	Value
Solid fuel outer radius (diameter) [cm]	0.40513 (0.81026)
IFBA coating outer radius (diameter) [cm]	0.4057226 (0.8114452)
Outer gas gap radius (diameter) [cm]	0.41656 (0.83312)
Cladding outer radius (diameter) [cm]	0.4572 (0.9144)
Fuel pin pitch [cm]	1.21006
Fuel assembly layout [-]	19x19
Fuel pins per assembly [-]	336
Guide tubes per assembly [-]	24
Instrumentation tubes per assembly [-]	1
IFBA fuel pins per assembly [-]	0, 84, 100, 156
Non-IFBA fuel pins per assembly [-]	336, 252, 236, 180
Fuel assembly pitch [cm]	23.1013
Interassembly gap [cm]	0.11016
Reflector outer radius (diameter) [cm]	160 (320)
Core barrel outer radius (diameter) [cm]	165 (330)
Downcomer radius (diameter) [cm]	245 (490)

Visual renderings for the assembly- and whole core-level geometries can be seen in Figure 3.1 and Figure 3.2. In Figure 3.1, one would normally expect to see spacer grids for certain axial cross-cuts through any given assembly. However, for this 2D model, the spacer girds are simply homogenized into the coolant.

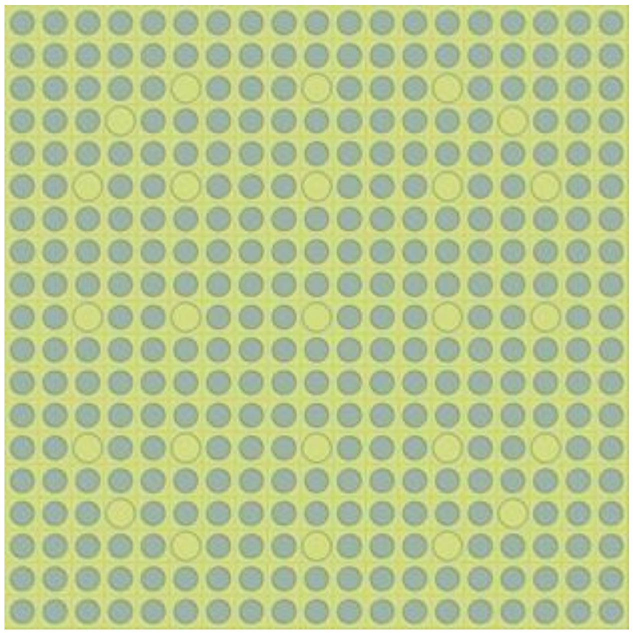


Figure 3.1 I²S-LWR assembly modeled in Serpent using solid fuel form[16].

[16] The distinction is made here since in the earliest stages of the I²S-LWR design, an annular fuel design was considered to accomidate a then-unknown amount of swelling of the fuel during burnup. Since that time, studies have shown that a solid fuel form should perform acceptably and has been the design of choice. All models in this thesis use the solid fuel geometry for I²S-LWR studies.

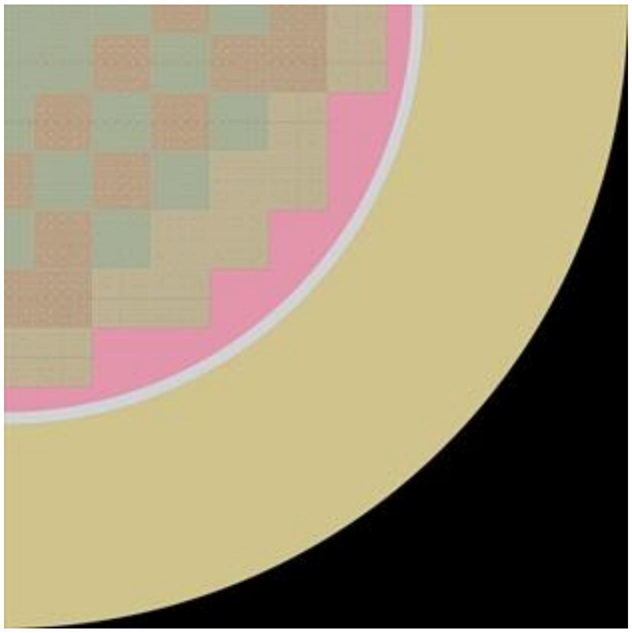


Figure 3.2 I²S-LWR quarter core modeled in Serpent.

Three fuel enrichments are used 2D Serpent model. They are arranged in an “out-in” pattern, where the fuel with the highest enrichment is on the periphery and decreasing in enrichment when moving toward the center. The core loading pattern with applied IFBA pattern can be seen in Figure 3.3. The enrichments selected for this model do not match those used in the actual I²S-LWR design, nor were they optimized over the entire cycle. They were chosen based on the approximately 12-month cycle length of the I²S-LWR as well as the 2D assembly-wise radial power profile with no boron at BOC.

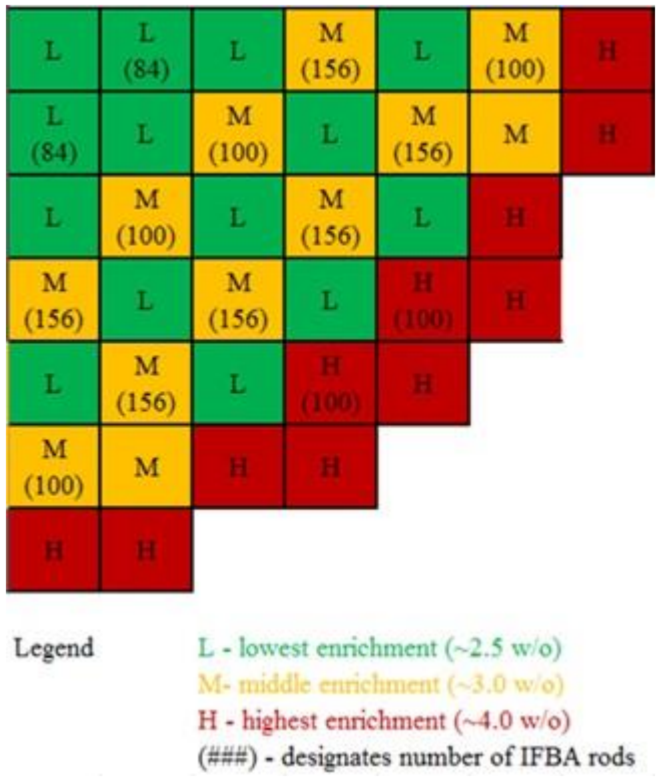

L
L (84)
L
M (156)
L
M (100)
H
L (84)
L
M (100)
L
M (156)
M
H
L
M (100)
L
M (156)
L
H
M (156)
L
M (156)
L
H (100)
H
L
M (156)
L
H (100)
H
M (100)
M
H
H
H
H
Legend
L - lowest enrichment (~2.5 w/o)
M- middle enrichment (~3.0 w/o)
H - highest enrichment (~4.0 w/o)
(###) - designates number of IFBA rods

Figure 3.3 Core loading pattern used in the 2D Serpent model.

3.2.2 Materials

The materials used in the 2D Serpent model are specified in

Table 3.2. The fuel density used (11.7161 g/cm^3) is derived from the theoretical density of U_3Si_2 (12.2 g/cm^3), assuming as-manufactured density to be 97.2% of theoretical density, and also taking into account the pellet dishing fraction of 1.2%. The fuel temperature of 735 K corresponds to the average fuel temperature experienced in the core (for all enrichments and axial locations). The coolant temperature of 590 K was selected based upon the average of the core inlet and outlet temperatures. IFBA, helium, cladding, and radial reflector temperatures were selected based upon reasonable estimates with respect to the fuel and coolant temperatures.

Table 3.2 List of materials used in the 2D Serpent model

Material	**Density**	**Temperature [K]**
4.0 w% Fuel (U_3Si_2)	11.7161 g/cm^3	735
Si	0.864303 g/cm^3	
^{234}U (0.04 w%)	0.004341 g/cm^3	
^{235}U	0.434072 g/cm^3	
^{238}U	10.413384 g/cm^3	
3.0 w% Fuel (U_3Si_2)	11.7161 g/cm^3	735
Si	0.864303 g/cm^3	
^{234}U (0.04 w%)	0.004341 g/cm^3	
^{235}U	0.325554 g/cm^3	
^{238}U	10.521902 g/cm^3	
2.5 w% Fuel (U_3Si_2)	11.7161 g/cm^3	735
Si	0.864303 g/cm^3	
^{234}U (0.04 w%)	0.004341 g/cm^3	
^{235}U	0.271295 g/cm^3	
^{238}U	10.576161 g/cm^3	
IFBA (ZrB_2, 60 a% ^{10}B enrichment)	6.08 g/cm^3	650
^{10}B	1.2 atom	
^{11}B	0.8 atom	
Zr	1 atom	
Helium Gas Gap (200 psi at 20 C)	0.002264 g/cm^3	625
He	1 atom	
APMT Advanced Steel	7.25	600
C	0.08 w%	
Al	5 w%	
Si	0.7 w%	
Cr	21 w%	
Mn	0.4 w%	
Fe	69.82 w%	
Mo	3 w%	
Light water (with 0.6% volume APMT to account for grid spacers)	0.727014 g/cm^3	590
^{1}H	4.56966E+22 atom/cc	

Table 3.2 continued

Material	**Density**	**Temperature [K]**
O	2.28483E+22 atom/cc	
C	1.74478E+18 atom/cc	
Al	4.85437E+19 atom/cc	
Si	6.52910E+18 atom/cc	
Cr	1.05798E+20 atom/cc	
Mn	1.90729E+18 atom/cc	
Fe	3.27511E+20 atom/cc	
Mo	8.19042E+18 atom/cc	
Core Reflector (10% water, 90% APMT by volume)	6.593764 g/cm^3	600
^{1}H	4.59724E+21 atom/cc	
O	2.29862E+21 atom/cc	
C	2.61717E+20 atom/cc	
Al	7.28156E+21 atom/cc	
Si	9.79366E+20 atom/cc	
Cr	1.58697E+22 atom/cc	
Mn	2.86094E+20 atom/cc	
Fe	4.91266E+22 atom/cc	
Mo	1.22856E+21 atom/cc	

CHAPTER 4 2D SERPENT MODEL STUDIES

4.1 Overview

This section addresses some applications of the 2D Serpent model: a comparison of the model with a 3D model made by Westinghouse Electric Company (WEC), a power density uncertainty analysis of Serpent, and a homogenized radial reflector study.

4.1.1 Objectives

After the creation of the I²S-LWR 2D Serpent model, it was necessary to verify that the model was correct. To accomplish this, WEC was asked to use the same parameters as the 2D Serpent model in their 3D nodal code package. They then provided the results and a comparison was then possible.

From the outset of the research and work described in this thesis, Serpent was identified as the program which would be used to model and simulate the physics of the I²S-LWR core. However, it was unknown how accurate the Serpent simulations, specifically of the I²S-LWR core, would actually be. In order to have confidence in the statistical uncertainties reported by Serpent, an analysis was done on the power densities at the quarter assembly level.

For the homogenized radial reflector study, the goal was to investigate the impact of varying the composition of the APMT stainless steel radial reflector around the core. Radial reflectors are meant to improve the neutron economy of the core by having some leakage neutrons return to core. A second benefit of radial reflectors is that they can reduce the neutron fluence and resulting radiation damage to structural components beyond the core [17] [18]. In the I^{2}S-LWR design, there are cooling channels in the radial reflector since it is of sufficient thickness to require cooling due to gamma heating. By volume, these channels account for approximately 10% of the reflector, leaving the steel with the remaining 90%. Understanding the degree of impact these channels have on the core neutronics will be beneficial when moving forward with the design of the I^{2}S-LWR.

4.1.2 Approach

To fairly compare the 2D Serpent model to WEC's 3D model, axial variations needed to be reduced to minimize the number of dissimilarities. This is why a Hot Zero Power (HZP) case with no soluble boron was selected for comparison, since the 2D model would come closest to matching the 3D model under these conditions due to the absence of axial variation in density and temperature of core materials. For the WEC comparison, the 2D Serpent model needed to be tweaked slightly to match their input parameters. Their nodal diffusion code uses the same cross section library to generate albedos for the radial reflector for all I^{2}S-LWR simulations. This set was generated at average operating conditions (so higher in temperature than what HZP would normally produce) with 500 ppm soluble boron (despite this test case having no soluble boron). The results from their 3D nodal code package were received as whole assembly relative power densities. The results from the 2D Serpent model gave relative power densities at the quarter assembly level, which were then averaged over assemblies to produce whole assembly relative power densities for equitable comparison with the WEC results.

The power density uncertainty analysis was conducted using two techniques: replica runs[17] and eighth-core symmetric averaging. While using replica runs is more correct for this type of analysis, the technique of symmetric averaging was also incorporated as a means of analyzing by using only a single MC simulation.

The homogeneous radial reflector study looked at the reactivity impact due to material variation in the radial reflector. Consideration was made for compositions of 0% steel and 100% water to 100% steel and 0% water (all % by volume). The step size was 10% from 0-60% steel and 5% from 65-100% steel. The reason for this step size selection was that most radial reflector designs have less than 40% coolant volume, hence the smaller step size for 60-100% steel, but included 0-60% simply for completeness in the analysis.

4.2 Model Comparison with WEC[18]

To verify the I²S-LWR 2D Serpent model, an entirely fresh core (no previously depleted assemblies) with three enrichments was used. The radial power distribution obtained by Serpent at the quarter-assembly level can be seen in Figure 4.1 for HZP conditions with a CBC of 1816 ppm. Runtime parameters include six million particles/cycle with 700 active and 300 inactive cycles. The Serpent simulation yielded k_{eff} of 0.99571 ± 0.00001 with a peak reported assembly error of 50 pcm.

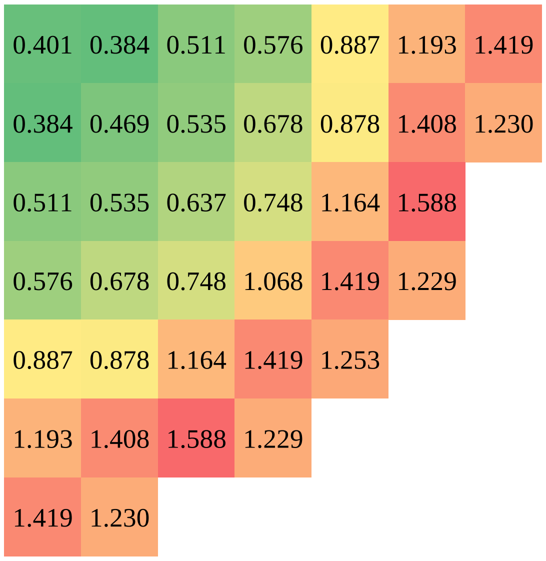
In order to compare these results with the WEC results, the quarter assembly-wise power distribution shown in Figure 4.1 was collapsed down to the whole assembly level. The Serpent results can then be seen in Figure 4.2.

[17] Replica runs are done by conducting the exact same Monte Carlo simulation several times. The only difference among the simulations is the random number seed used in each case. The purpose is to observe the statistical uncertainty between the separate simulations, since the same result is expected (within statistics) for all of the individual runs.
[18] The work expressed in this section, as well as in section 4.4 is part of a previous publication [24].

0.401	0.370	0.399	0.491	0.531	0.535	0.616	0.799	0.974	1.127	1.260	1.618	1.220
0.370	0.437	0.469	0.493	0.542	0.617	0.694	0.756	0.926	1.324	1.459	1.568	1.123
0.399	0.469	0.502	0.528	0.578	0.656	0.743	0.819	1.009	1.411	1.437	1.345	0.883
0.491	0.493	0.528	0.590	0.635	0.646	0.757	0.999	1.222	1.726	1.519		
0.531	0.542	0.578	0.635	0.690	0.726	0.864	1.122	1.314	1.723	1.383		
0.535	0.617	0.656	0.646	0.726	0.917	1.074	1.390	1.484	1.573	1.193		
0.616	0.694	0.743	0.757	0.864	1.074	1.205	1.437	1.365	1.268	0.883		
0.799	0.756	0.819	0.999	1.122	1.390	1.437	1.529	1.273				
0.974	0.926	1.009	1.222	1.314	1.484	1.365	1.273	0.936				
1.127	1.324	1.411	1.726	1.723	1.573	1.268						
1.260	1.459	1.437	1.519	1.383	1.193	0.883						
1.618	1.568	1.345										
1.220	1.123	0.883										

Figure 4.1 Quarter assembly level power distribution from Serpent.

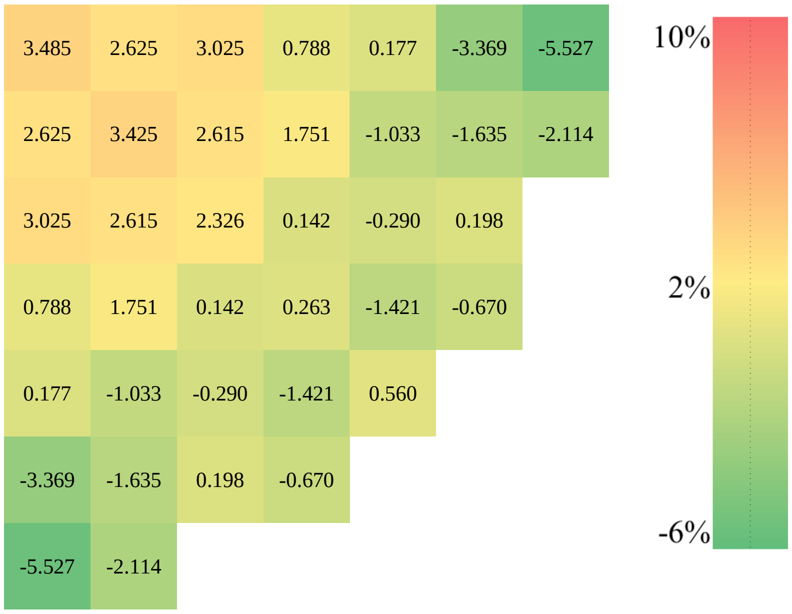
WEC analyzed the same core as a 3D nodal model in their code package suite. The radial reflector used by WEC is homogeneous 90% stainless steel, 10% coolant with 500 ppm boron at average core temperature, with transport-corrected cross sections. The Serpent model matched these parameters but used the ENDB-VII cross section library. While there will be some differences between WEC's 3D nodal model and the 2D Serpent continuous energy model results, it was expected that they would be relatively small and therefore the comparison would reveal any errors or significant differences between the two models.



0.401	0.384	0.511	0.576	0.887	1.193	1.419
0.384	0.469	0.535	0.678	0.878	1.408	1.230
0.511	0.535	0.637	0.748	1.164	1.588	
0.576	0.678	0.748	1.068	1.419	1.229	
0.887	0.878	1.164	1.419	1.253		
1.193	1.408	1.588	1.229			
1.419	1.230					

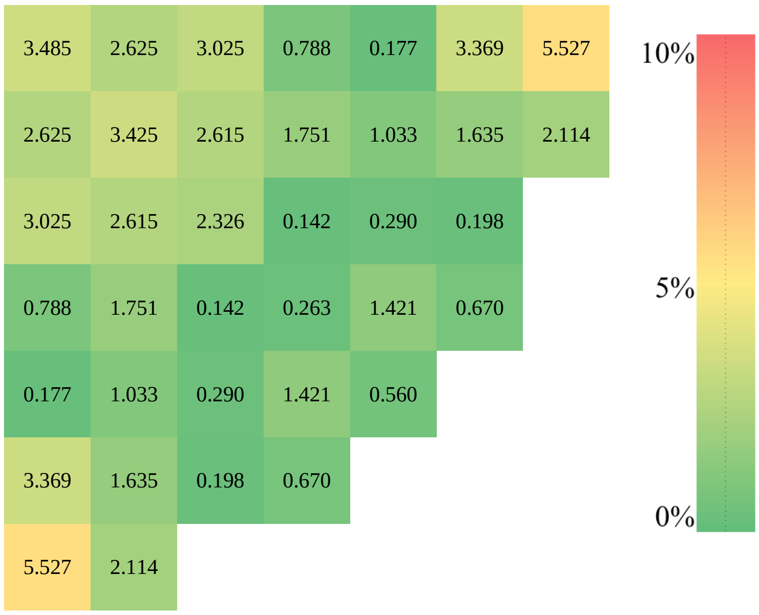
Figure 4.2 Whole assembly level power distribution from Serpent.

Although the WEC results are not provided here, the differences between the two models will be discussed below. The raw difference between the two models can be seen in Figure 4.3. To more accurately gage the magnitude of the differences, the absolute differences for each assembly location are shown in Figure 4.4. The largest difference in magnitude is along the periphery, but there also appears to be slight disagreement near the center region.



3.485	2.625	3.025	0.788	0.177	-3.369	-5.527
2.625	3.425	2.615	1.751	-1.033	-1.635	-2.114
3.025	2.615	2.326	0.142	-0.290	0.198	
0.788	1.751	0.142	0.263	-1.421	-0.670	
0.177	-1.033	-0.290	-1.421	0.560		
-3.369	-1.635	0.198	-0.670			
-5.527	-2.114					

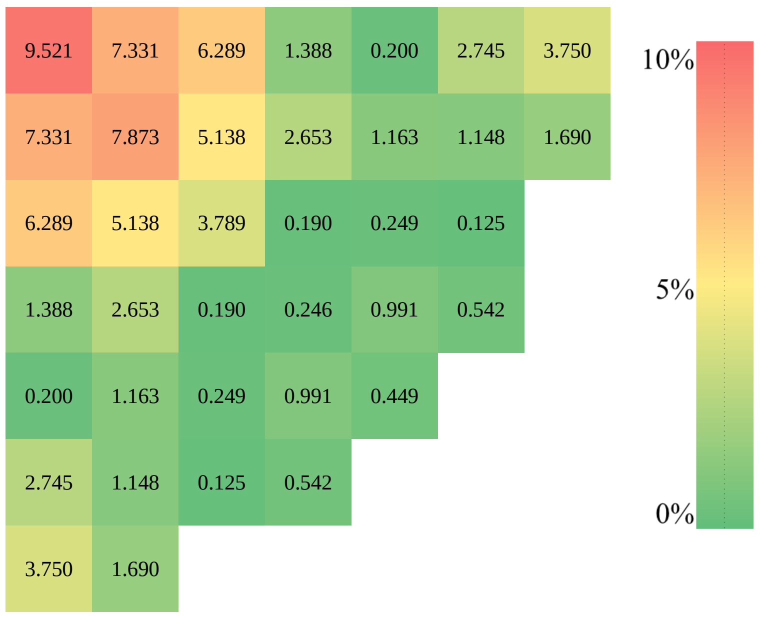
Figure 4.3 (Serpent – WEC) percent difference in assembly power.



3.485	2.625	3.025	0.788	0.177	3.369	5.527
2.625	3.425	2.615	1.751	1.033	1.635	2.114
3.025	2.615	2.326	0.142	0.290	0.198	
0.788	1.751	0.142	0.263	1.421	0.670	
0.177	1.033	0.290	1.421	0.560		
3.369	1.635	0.198	0.670			
5.527	2.114					

Figure 4.4 |Serpent – WEC| absolute percent difference in assembly power.

The largest relative differences is observed at the center, where the 2D Serpent model predicts about 10% higher power in the central assembly and about 7% higher power in surrounding assemblies (Figure 4.5). However, the relative power is fairly low near the center for this test configuration and a higher percent difference is acceptable when accounting for this. Relative differences decrease when moving toward the periphery of the core, where a majority of assemblies are within 1% and the maximum difference is just over 3%.



9.521	7.331	6.289	1.388	0.200	2.745	3.750
7.331	7.873	5.138	2.653	1.163	1.148	1.690
6.289	5.138	3.789	0.190	0.249	0.125	
1.388	2.653	0.190	0.246	0.991	0.542	
0.200	1.163	0.249	0.991	0.449		
2.745	1.148	0.125	0.542			
3.750	1.690					

Figure 4.5 |Serpent – WEC| / WEC absolute relative difference in assembly power.

4.3 Power Density Uncertainty Analysis[19]

This work addresses some of the issues of MC mentioned in Section 2.2.1 Advantages and Drawbacks, and focuses on investigating the credibility of MC-reported uncertainties at a static time by both comparing them with replica runs and the sample standard deviation of the corresponding eighth-symmetry group of quarter assemblies (Figure 4.6). While more traditional analyses of this sort relied on using only replica runs,

[19] The work expressed in this section is part of a previous publication [25].

it can become computational expensive to run many instances of the same MC simulation. Instead, symmetric grouping is evaluated as a potential alternative which offers the advantage of needing only a single simulation instance.

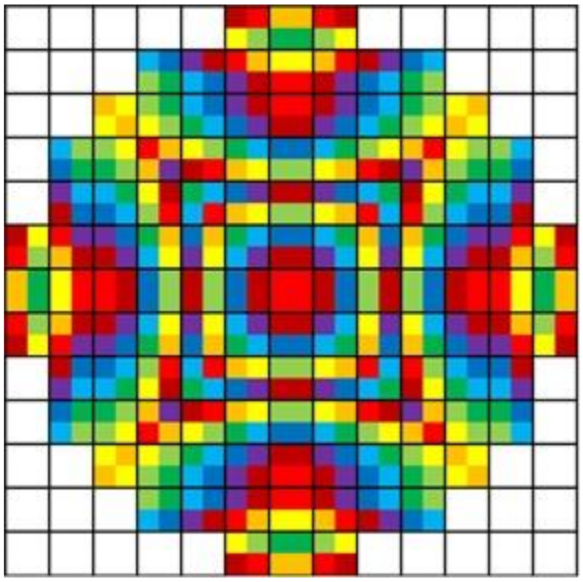


Figure 4.6 Eighth-symmetry locations[20] in the 484 quarter assembly locations of the 121 assembly I²S-LWR core (65 groups in total).

4.3.1 Replica Run Analysis

A series of replica runs were run as a means to compare the symmetric grouping cases against a more traditional method. For the replica run analysis, 30 identical simulations were performed with 1x10^5 particles per cycle over 3000 active and 500 inactive cycles. Each simulation ran on a single processor (no parallelization). After the 30 runs finished, a **reported** (relative) uncertainty was found for each of the 484 quarter

[20] Colors are repeated for different groups. The pallet used was selected for high contrast of adjacent groups.

assembly locations by averaging the Serpent-reported (relative) power density uncertainties for the 30 runs. These can be seen in Figure 4.7.

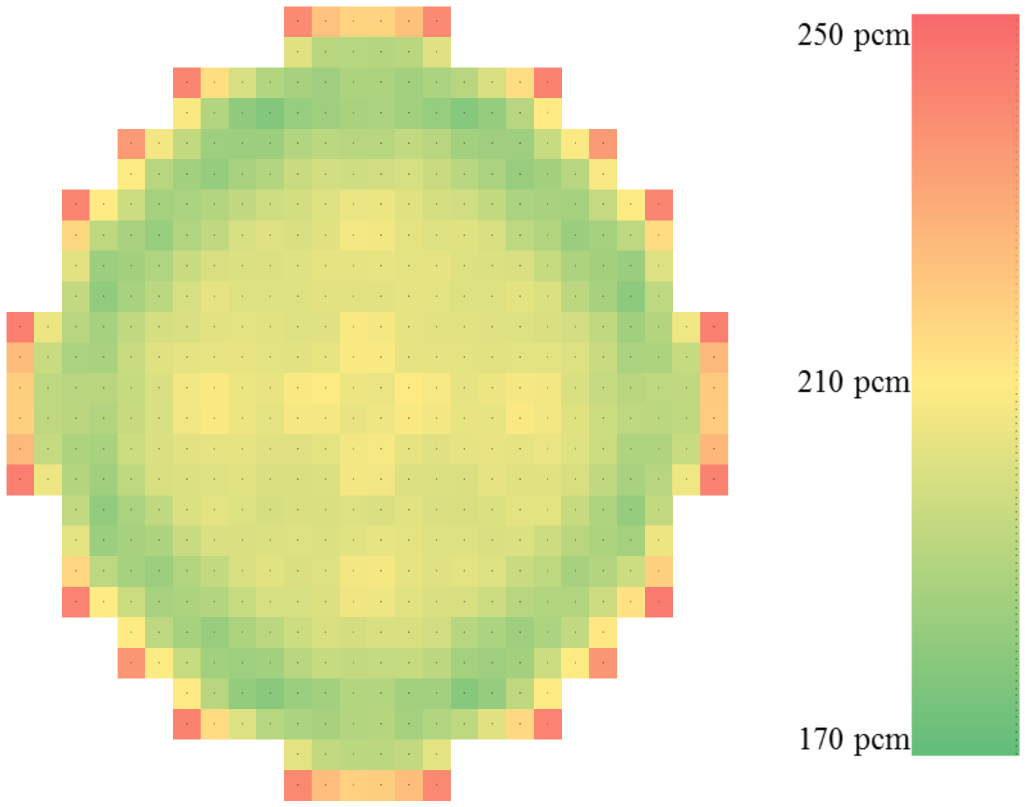


Figure 4.7 Reported relative uncertainty from Serpent for the replica runs.

As can be seen from Figure 4.7, averaging the reported uncertainties from the 30 replica runs yields a fairly symmetric uncertainty profile. As expected, the largest uncertainties are located at the periphery (where the flux, and thus particle sampling, is lowest and has the largest gradient). After the reported uncertainties, a set of corresponding **observed** uncertainties was found for each of the 484 quarter assembly locations by calculating the standard deviation of the Serpent-reported power densities for the 30 runs. The standard deviations themselves are absolute, but are made relative for a fair comparison by dividing by the average value. These can be seen in Figure 4.8.

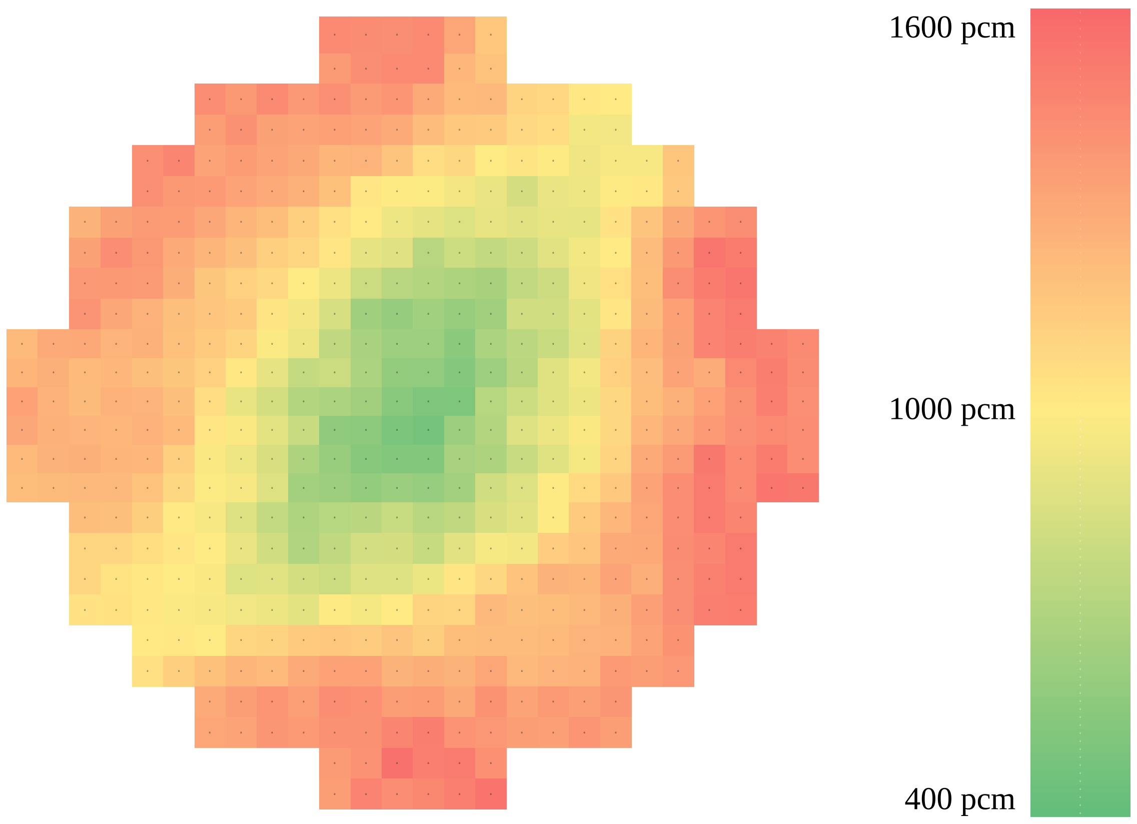


Figure 4.8 Observed relative uncertainty (calculated) from the replica runs.

Unlike the very symmetric results seen with the reported uncertainties, the observed uncertainties of Figure 4.8 have noticeable radial tilts in the results. While undesired, these tilts are characteristic of any MC simulation due to statistical uncertainties propagating through multiple particle generations. Another observation made between the two, as evident from their respective scales, is that the observed uncertainty is higher than the reported uncertainty. Further comparison between the two can be seen in Figure 4.9, which takes the observed uncertainty result and divides it by the reported uncertainty at each individual quarter assembly location. This is the factor by which the uncertainty is underestimated.

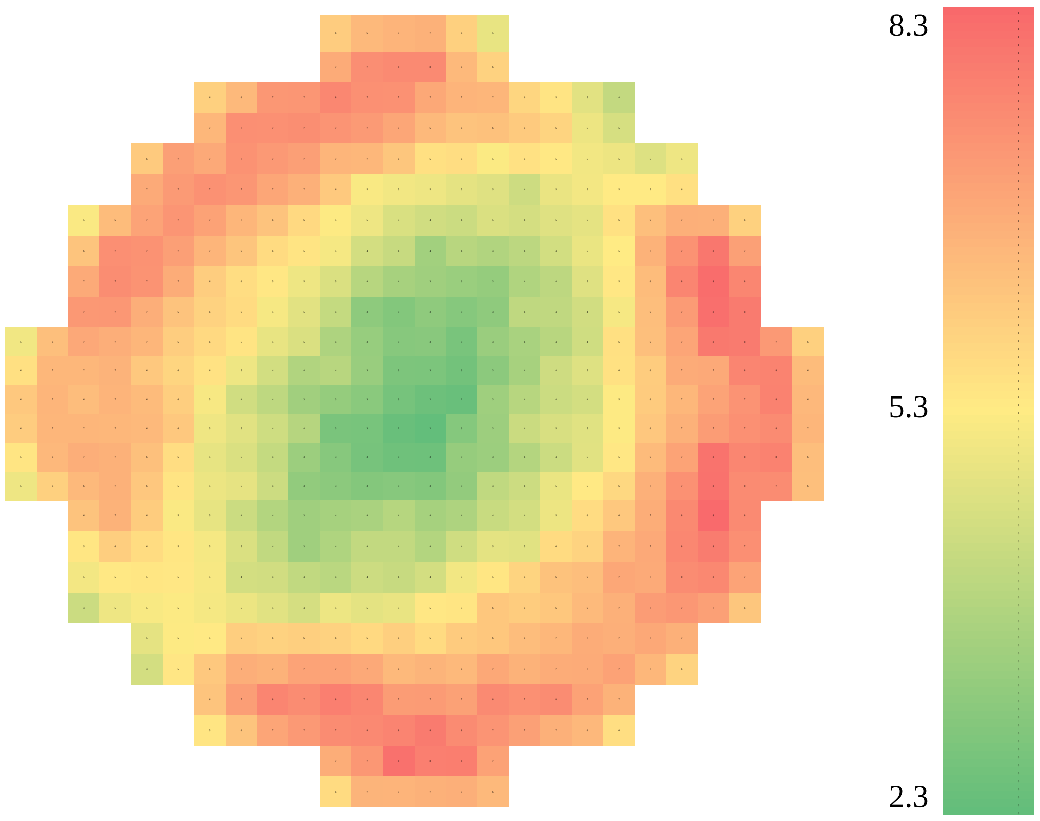


Figure 4.9 Replica run comparison – ratio of observed / reported relative uncertainties.

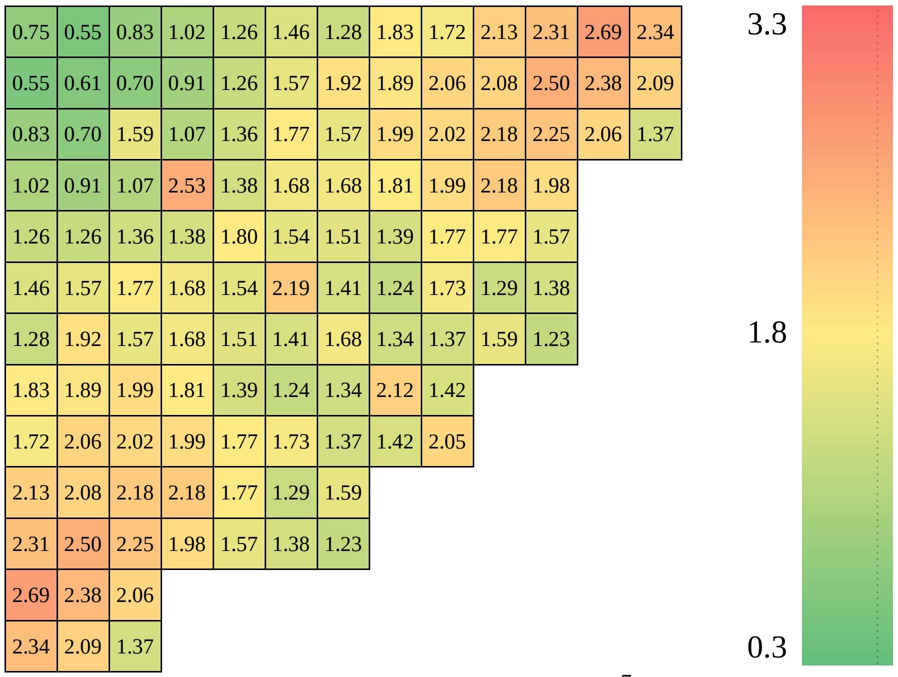
For the results shown in Figure 4.9, the average ratio is 5.67 ± 1.41 with extreme values of 2.32 and 8.28. Since the minimum was 2.32, the observed uncertainty was greater than the reported uncertainty for each of the 484 quarter assembly locations.

4.3.2 Symmetric Grouping Analysis

Due to octant symmetry, all locations representing quarter assemblies may be divided into groups of either four (along the diagonals) or eight (elsewhere) symmetric locations. Due to the symmetry of the 2D Serpent model, results in each group should ideally be identical, or in the case of MC simulations could be expected to be spread consistent with the estimated statistical uncertainty. The Serpent-**reported** power contributions and associated relative uncertainties (standard deviation) for each eighth-symmetric group location were averaged (Equation 3 from Section 2.2.1.2 Stochastic Uncertainties). Additionally, the **observed** uncertainty (standard deviation) was found by applying (Equation 4 from Section 2.2.1.2 Stochastic Uncertainties) to each group of 4 or 8 power values at symmetric locations. It was then subsequently used to compare it with the MC-reported uncertainty.

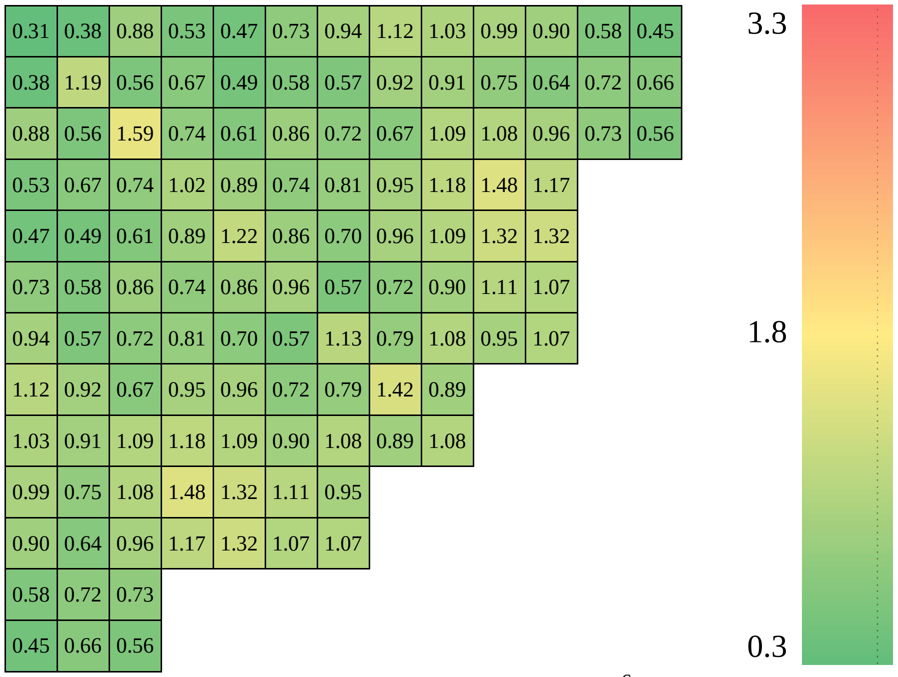
To clarify: for these symmetric grouping cases, instead of running replica runs, a single Serpent job output was used to find an observed uncertainty at each eighth-symmetric location using all the corresponding assemblies in the full core model. It should be noted here that calculating a sample standard deviation is usually only valid for larger sample sizes; however, it is used in this study as a rough estimation metric for groups of size four and eight.

The **ratio** between the **observed** uncertainty (sample standard deviation) in relative power to the MC-**reported** statistical uncertainty in relative power at each quarter assembly location is calculated for each of several cases. Ideally, it should be around unity (MC result matches observed result) everywhere. For consistent comparison, two groups of cases were considered. The first group used the same total number of active particles ($3x10^9$) while varying the number of active cycles and particles per cycle used: 100 active cycles (Figure 4.10), 1000 active cycles, (Figure 4.11), and 10000 active cycles (Figure 4.12). The second group use the same number of particles per cycle ($3x10^5$) while varying the number of active cycles used: 100 active cycles (Figure 4.14), 1000 active cycles (Figure 4.13), and 10000 active cycles (Figure 4.12).



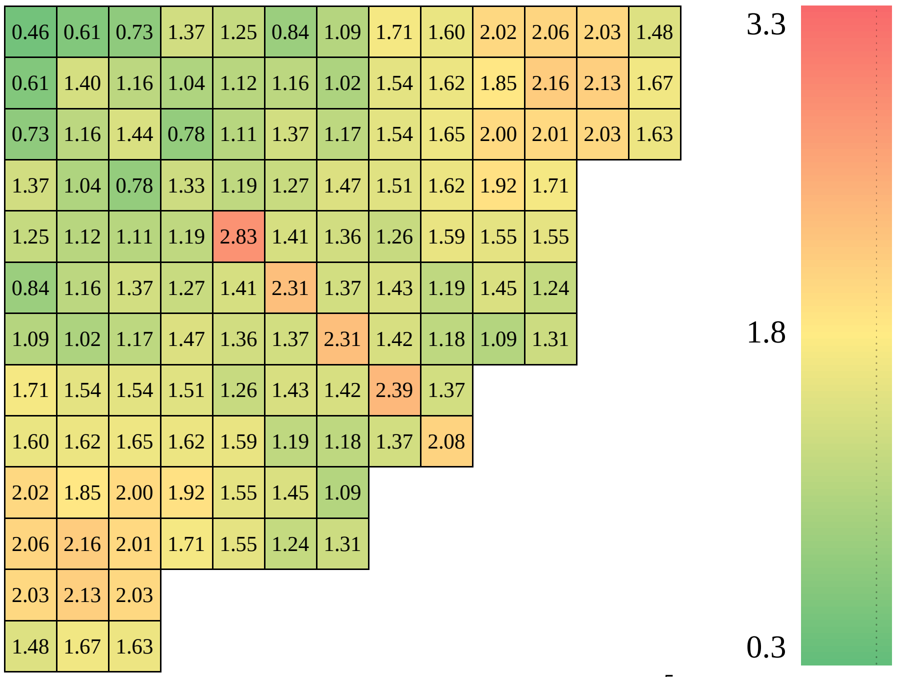
0.75	0.55	0.83	1.02	1.26	1.46	1.28	1.83	1.72	2.13	2.31	2.69	2.34
0.55	0.61	0.70	0.91	1.26	1.57	1.92	1.89	2.06	2.08	2.50	2.38	2.09
0.83	0.70	1.59	1.07	1.36	1.77	1.57	1.99	2.02	2.18	2.25	2.06	1.37
1.02	0.91	1.07	2.53	1.38	1.68	1.68	1.81	1.99	2.18	1.98		
1.26	1.26	1.36	1.38	1.80	1.54	1.51	1.39	1.77	1.77	1.57		
1.46	1.57	1.77	1.68	1.54	2.19	1.41	1.24	1.73	1.29	1.38		
1.28	1.92	1.57	1.68	1.51	1.41	1.68	1.34	1.37	1.59	1.23		
1.83	1.89	1.99	1.81	1.39	1.24	1.34	2.12	1.42				
1.72	2.06	2.02	1.99	1.77	1.73	1.37	1.42	2.05				
2.13	2.08	2.18	2.18	1.77	1.29	1.59						
2.31	2.50	2.25	1.98	1.57	1.38	1.23						
2.69	2.38	2.06										
2.34	2.09	1.37										

Figure 4.10 Observe/Reported uncertainty ratios for 3x10^7 particles per cycle, 100 active cycles, and 200 inactive cycles. Average ratio: 2.69 ± 0.47.



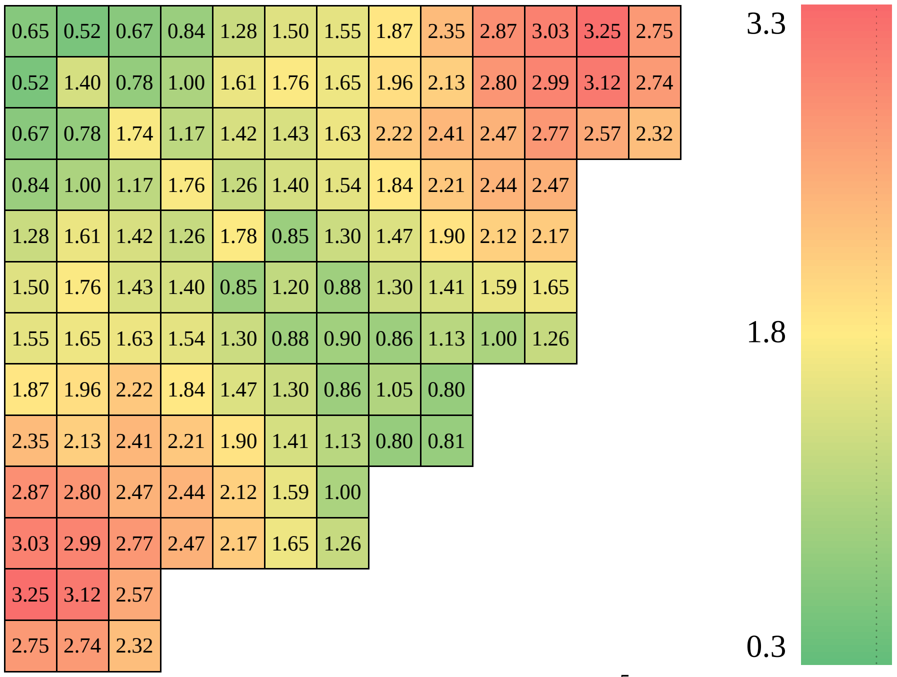
0.31	0.38	0.88	0.53	0.47	0.73	0.94	1.12	1.03	0.99	0.90	0.58	0.45
0.38	1.19	0.56	0.67	0.49	0.58	0.57	0.92	0.91	0.75	0.64	0.72	0.66
0.88	0.56	1.59	0.74	0.61	0.86	0.72	0.67	1.09	1.08	0.96	0.73	0.56
0.53	0.67	0.74	1.02	0.89	0.74	0.81	0.95	1.18	1.48	1.17		
0.47	0.49	0.61	0.89	1.22	0.86	0.70	0.96	1.09	1.32	1.32		
0.73	0.58	0.86	0.74	0.86	0.96	0.57	0.72	0.90	1.11	1.07		
0.94	0.57	0.72	0.81	0.70	0.57	1.13	0.79	1.08	0.95	1.07		
1.12	0.92	0.67	0.95	0.96	0.72	0.79	1.42	0.89				
1.03	0.91	1.09	1.18	1.09	0.90	1.08	0.89	1.08				
0.99	0.75	1.08	1.48	1.32	1.11	0.95						
0.90	0.64	0.96	1.17	1.32	1.07	1.07						
0.58	0.72	0.73										
0.45	0.66	0.56										

Figure 4.11 Observe/Reported uncertainty ratios for 3x10^6 particles per cycle, 1000 active cycles, and 500 inactive cycles. Average ratio: 0.86 ± 0.26.



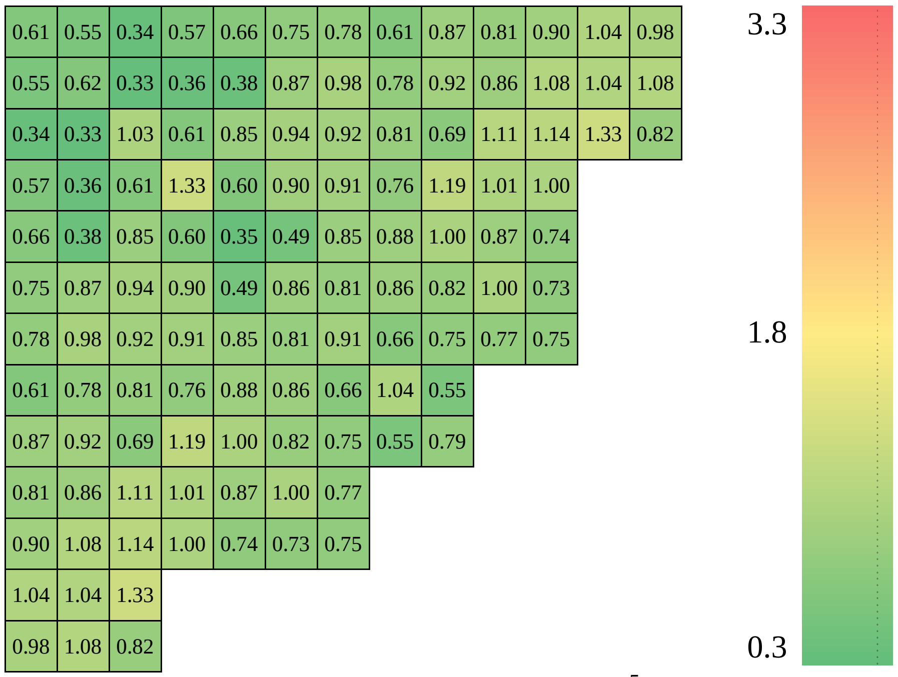
0.46	0.61	0.73	1.37	1.25	0.84	1.09	1.71	1.60	2.02	2.06	2.03	1.48
0.61	1.40	1.16	1.04	1.12	1.16	1.02	1.54	1.62	1.85	2.16	2.13	1.67
0.73	1.16	1.44	0.78	1.11	1.37	1.17	1.54	1.65	2.00	2.01	2.03	1.63
1.37	1.04	0.78	1.33	1.19	1.27	1.47	1.51	1.62	1.92	1.71		
1.25	1.12	1.11	1.19	2.83	1.41	1.36	1.26	1.59	1.55	1.55		
0.84	1.16	1.37	1.27	1.41	2.31	1.37	1.43	1.19	1.45	1.24		
1.09	1.02	1.17	1.47	1.36	1.37	2.31	1.42	1.18	1.09	1.31		
1.71	1.54	1.54	1.51	1.26	1.43	1.42	2.39	1.37				
1.60	1.62	1.65	1.62	1.59	1.19	1.18	1.37	2.08				
2.02	1.85	2.00	1.92	1.55	1.45	1.09						
2.06	2.16	2.01	1.71	1.55	1.24	1.31						
2.03	2.13	2.03										
1.48	1.67	1.63										

Figure 4.12 Observe/Reported uncertainty ratios for 3x10^5 particles per cycle, 10000 active cycles, and 1000 inactive cycles. Average ratio: 1.48 ± 0.41.



0.65	0.52	0.67	0.84	1.28	1.50	1.55	1.87	2.35	2.87	3.03	3.25	2.75
0.52	1.40	0.78	1.00	1.61	1.76	1.65	1.96	2.13	2.80	2.99	3.12	2.74
0.67	0.78	1.74	1.17	1.42	1.43	1.63	2.22	2.41	2.47	2.77	2.57	2.32
0.84	1.00	1.17	1.76	1.26	1.40	1.54	1.84	2.21	2.44	2.47		
1.28	1.61	1.42	1.26	1.78	0.85	1.30	1.47	1.90	2.12	2.17		
1.50	1.76	1.43	1.40	0.85	1.20	0.88	1.30	1.41	1.59	1.65		
1.55	1.65	1.63	1.54	1.30	0.88	0.90	0.86	1.13	1.00	1.26		
1.87	1.96	2.22	1.84	1.47	1.30	0.86	1.05	0.80				
2.35	2.13	2.41	2.21	1.90	1.41	1.13	0.80	0.81				
2.87	2.80	2.47	2.44	2.12	1.59	1.00						
3.03	2.99	2.77	2.47	2.17	1.65	1.26						
3.25	3.12	2.57										
2.75	2.74	2.32										

Figure 4.13 Observe/Reported uncertainty ratios for 3x10^5 particles per cycle, 1000 active cycles, and 500 inactive cycles. Average ratio: 1.75 ± 0.71.



0.61	0.55	0.34	0.57	0.66	0.75	0.78	0.61	0.87	0.81	0.90	1.04	0.98
0.55	0.62	0.33	0.36	0.38	0.87	0.98	0.78	0.92	0.86	1.08	1.04	1.08
0.34	0.33	1.03	0.61	0.85	0.94	0.92	0.81	0.69	1.11	1.14	1.33	0.82
0.57	0.36	0.61	1.33	0.60	0.90	0.91	0.76	1.19	1.01	1.00		
0.66	0.38	0.85	0.60	0.35	0.49	0.85	0.88	1.00	0.87	0.74		
0.75	0.87	0.94	0.90	0.49	0.86	0.81	0.86	0.82	1.00	0.73		
0.78	0.98	0.92	0.91	0.85	0.81	0.91	0.66	0.75	0.77	0.75		
0.61	0.78	0.81	0.76	0.88	0.86	0.66	1.04	0.55				
0.87	0.92	0.69	1.19	1.00	0.82	0.75	0.55	0.79				
0.81	0.86	1.11	1.01	0.87	1.00	0.77						
0.90	1.08	1.14	1.00	0.74	0.73	0.75						
1.04	1.04	1.33										
0.98	1.08	0.82										

Figure 4.14 Observe/Reported uncertainty ratios for 3x10^5 particles per cycle, 100 active cycles, and 400 inactive cycles. Average ratio: 0.82 ± 0.22.

For all cases above, it is evident that the center assemblies consistently have a lower observed uncertainty than the Serpent-reported uncertainty. The reason for this might be that since the assemblies are all close in proximity, their results are closely coupled and averaging produces rather small differences. Moving radially outwards, the ratios consistently increase. There did not seem to be many noticeable trends in the behavior between the cases in either of the two groupings. It should be noted that the five cases above all used Serpent 2.1.23 with Open MP for parallelization. An additional case ran with the US distributive version, Serpent 1.1.7, with 3x10^6 active particles and 100 active cycles. The parallelization method used for this case was MPI[21] and the results can be seen in Figure 4.15.

[21] Open MP and MPI are two common types of protocols used in parallel computing. Using MPI *duplicates* the task at hand across the available processors; having them independently generate a simulation and report the results back when they have finished. Using Open MP *shares* the task among the available processors; allowing for constant communication during the computing process.

3.5	2.3	4.5	4.6	5.8	4.4	5.9	6.3	8.4	10.1	11.1	10.0	8.6
2.3	2.1	4.8	5.2	5.1	4.5	4.0	5.3	7.3	10.0	10.2	10.1	8.9
4.5	4.8	6.5	5.5	4.8	4.8	4.0	5.2	6.4	8.7	10.0	9.4	8.5
4.6	5.2	5.5	13.8	5.8	6.8	5.1	8.2	9.3	9.9	10.9		
5.8	5.1	4.8	5.8	12.5	6.9	8.2	9.5	11.0	11.5	10.1		
4.4	4.5	4.8	6.8	6.9	9.2	8.2	9.8	10.5	11.7	10.8		
5.9	4.0	4.0	5.1	8.2	8.2	11.4	10.8	11.7	10.4	10.3		
6.3	5.3	5.2	8.2	9.5	9.8	10.8	19.5	10.1				
8.4	7.3	6.4	9.3	11.0	10.5	11.7	10.1	14.4				
10.1	10.0	8.7	9.9	11.5	11.7	10.4						
11.1	10.2	10.0	10.9	10.1	10.8	10.3						
10.0	10.1	9.4										
8.6	8.9	8.5										

Figure 4.15 Observe/Reported uncertainty ratios for 3x10^6 particles per cycle, 1000 active cycles, and 1000 inactive cycles. Average ratio: 8.08 ± 2.91. (Serpent 1.1.7)

When comparing Figure 4.15 to Figure 4.11, which were identical MC simulations other than the version of Serpent used, it is clear that Serpent 1.1.7 underpredicts the statistical uncertainty compared to Serpent 2.1.23 (since the observed uncertainties are similar). The difference between the two will be discussed in the next section.

4.3.3 Comparison between the Two Uncertainty Analysis Methods

When comparing Figure 4.9 to the symmetric grouping cases done using Serpent 2.1.23 (Figure 4.10 - Figure 4.14), the behavior is not very similar. While the general shape of having the lowest ratio at the center and the largest near the periphery in maintained, the results still differ by a rather large factor. From this, symmetric grouping and averaging in Serpent 2 could be considered acceptable for capturing the relative uncertainty behavior of replica runs, but gives too low of an estimate for the observed uncertainty by a factor of approximately five. This is still valuable to know though and could be of use, since factor of difference is known and the general behavior is correct. Additional analyses using symmetric averaging in Serpent 2.1.23 could simply claim the factor on top of its profile and that should be acceptable for a rough estimate of the actual uncertainty.

When comparing the general behavior of the replica run results of Figure 4.9 to the Serpent 1.1.7 results of Figure 4.15, it appears that these two are more similar than the replica runs versus Serpent 2.1.23 results. The reason for this is believed to be that the MPI parallelization of Serpent 1.1.7 acts very much like using replica runs, since the task at hand is duplicated to all processors.

4.4 Homogenized Radial Reflector Study

From Section 4.2 Model Comparison with WEC, the fact that there is strong agreement along the periphery of the core (low relative differences) between the 2D Serpent model and the 3D WEC model is very positive toward supporting the 2D Serpent model's use in radial reflector studies. Since the peripheral assemblies are the closest in proximity to the reflector, they are the most important to an investigation. While the relative differences at the center of the core are quite large, the model overall seems acceptable for radial reflector studies.

For this analysis, the 2D Serpent simulations were performed using various soluble boron concentrations (0, 500, 1000, and 1500 ppm) at HFP conditions. The case of having no soluble absorber was used in order to capture only the effect of varying water and steel composition and most closely relates to core performance at EOC[22]. Each case used 75,000

[22] Most PWRs do not reach zero boron concentration at EOC due to the challenge of diminishing reactivity returns from a significant volume of water required to dilute the system. While 0 ppm was used as a simple benchmark for this study, typical EOC boron concentration values for existing PWRs would be closer to 50 ppm.

particles per cycle with 10,000 active cycles and 250 inactive cycles. Most simulations had an uncertainty of 7 pcm, which ran in parallel on 8 processors.

4.4.1 0 PPM Soluble Boron Concentration

Having soluble boron would of course affect the coolant to have less reactivity compared to the case without; so an unbiased result for water/steel composition would use no boron. As for HFP, it was selected since the reactivity impact of interest is during normal operation. Other notable parameters include using the average coolant temperature and density (317 °C and 0.68764 g/cm^3) for all water inside the core, in the radial reflector, and outside the core in the downcomer region. Figure 4.16 shows the results after running all the different composition cases for 0 ppm.

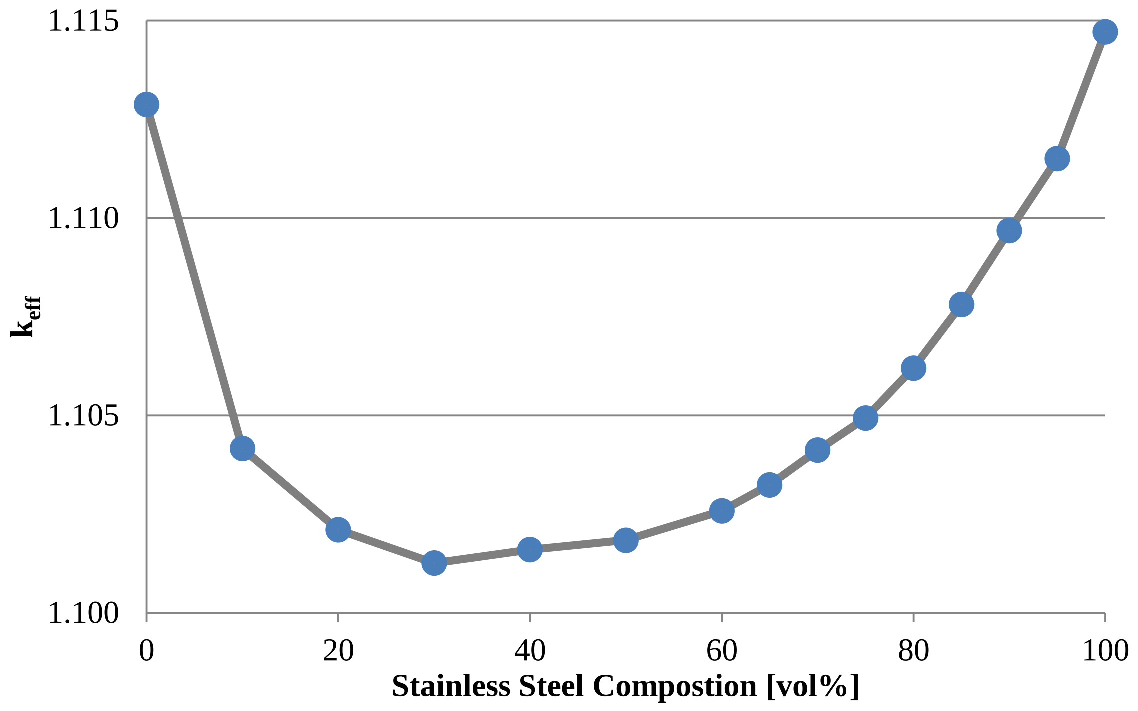


Figure 4.16 Reactivity impact of radial reflector material composition variation with 0 ppm soluble boron concentration. Maximum reported error: 21 pcm.

It should be noted here that the results observed in Figure 4.16 are very similar to those obtained from a radial reflector material composition study done on the International Reactor Innovative and Secure back in 2002 [19]. As can be seen, the reactivity impact of the radial reflector can be quite significant. While the pure water and steel cases both yield high eigenvalues, there appears to be some competing physics in the mixture cases. For the 100% steel case, fast neutrons are reflected back into the core. While many scattering events may occur before returning, they are still able to travel back into the core to increase the neutron economy. For the 100% water case, neutrons are thermalized and eventually return to the core. For the steel/water mixture cases, neutrons are both thermalized by the water and reflected by the steel. This results in the neutrons slowly scattering around in the mixed medium with now a higher likelihood of absorption losses in the steel. These phenomena are summarized in Figure 4.17.

While the fuel used in this simulation is for BOC-like conditions since it is fresh, the coolant used is for EOC-like conditions since it has no soluble boron. Carefully engineering the radial reflector with its cooling channels to have the most beneficial reactivity performance could, by Figure 4.17, provide a nontrivial amount of reactivity at EOC. This could potentially extend the cycle length or reduce the cost of fuel via lower fuel enrichment requirements; both consequences enhancing the economics of the overall I^{2}S-LWR design.

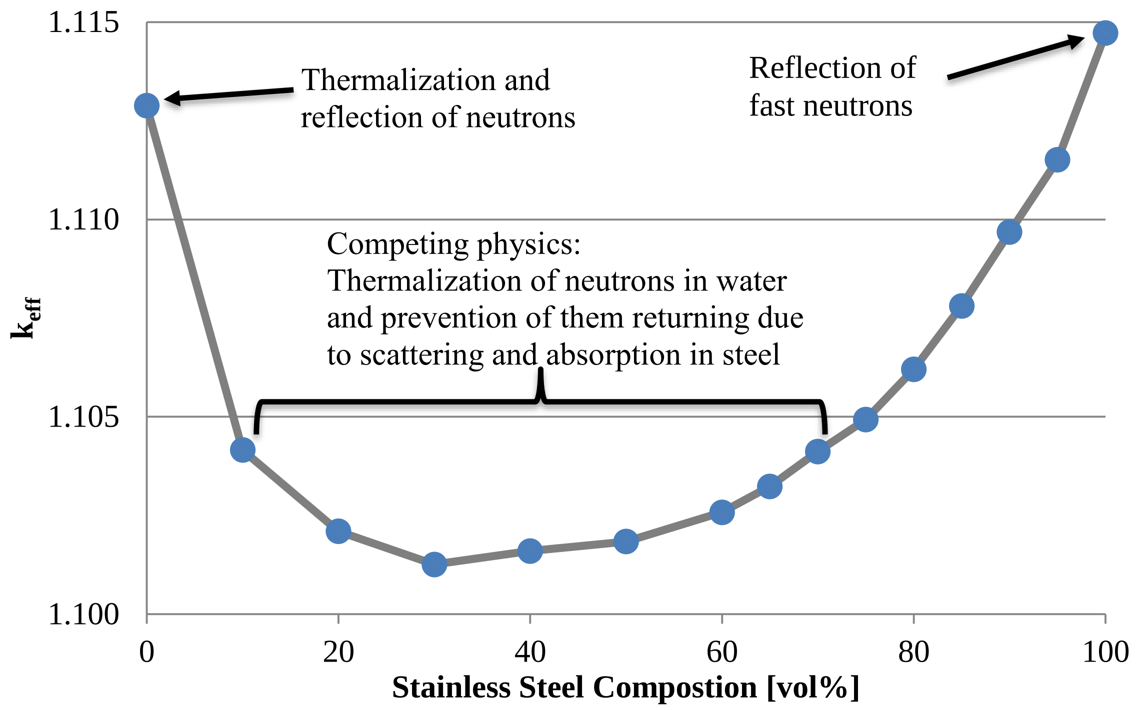


Figure 4.17 Phenomena observed in radial reflector composition variation.

4.4.2 500 ppm Soluble Boron Concentration

The results for the 500 ppm case can be seen in Figure 4.18. As with the 0 ppm analysis and as one would expect, the highest reactivity is observed for the pure steel case. However, with the addition of soluble boron into the model, the reactivity is penalized more for the pure borated water case, as one would expect.

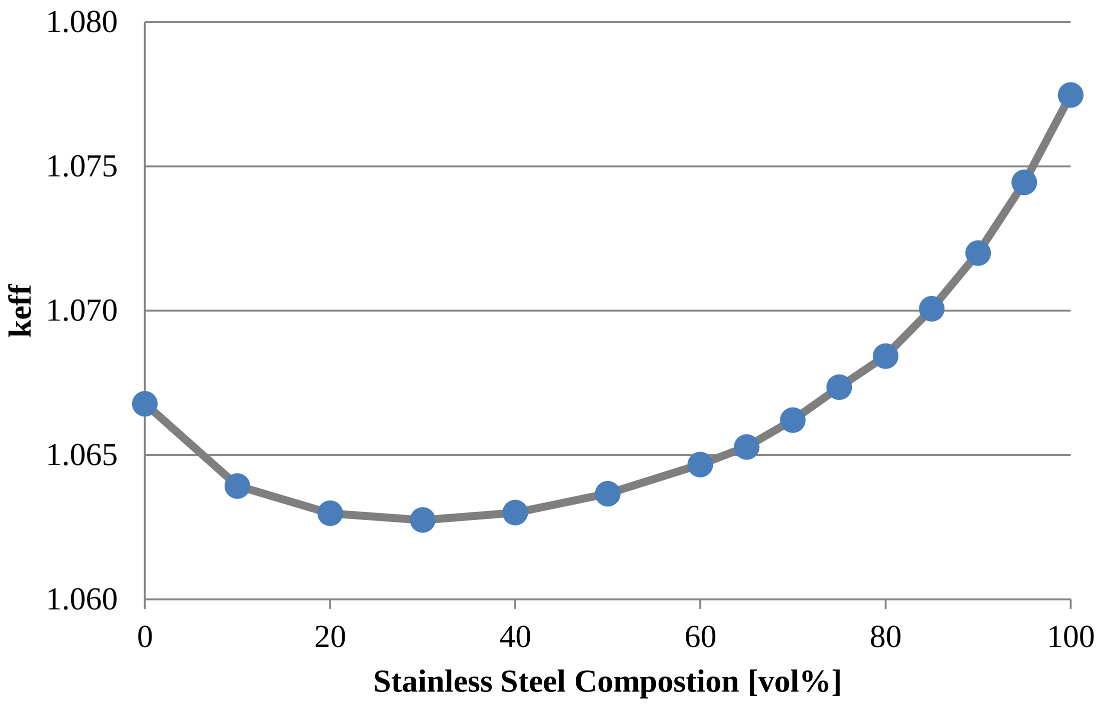


Figure 4.18 Reactivity impact of radial reflector material composition variation with 500 ppm soluble boron concentration. Maximum reported error: 7 pcm.

4.4.3 1000 ppm Soluble Boron Concentration

The results for the 1000 ppm case can be seen in Figure 4.19. Just like in the previous two analyses, the highest reactivity is observed for the pure steel case. While the 0 ppm case is the closest approximation to EOC conditions, the 500 and 1000 ppm cases are more characteristic of what would be experienced during the middle of the cycle. The final 1500 ppm case to follow would be a good approximation for BOC conditions.

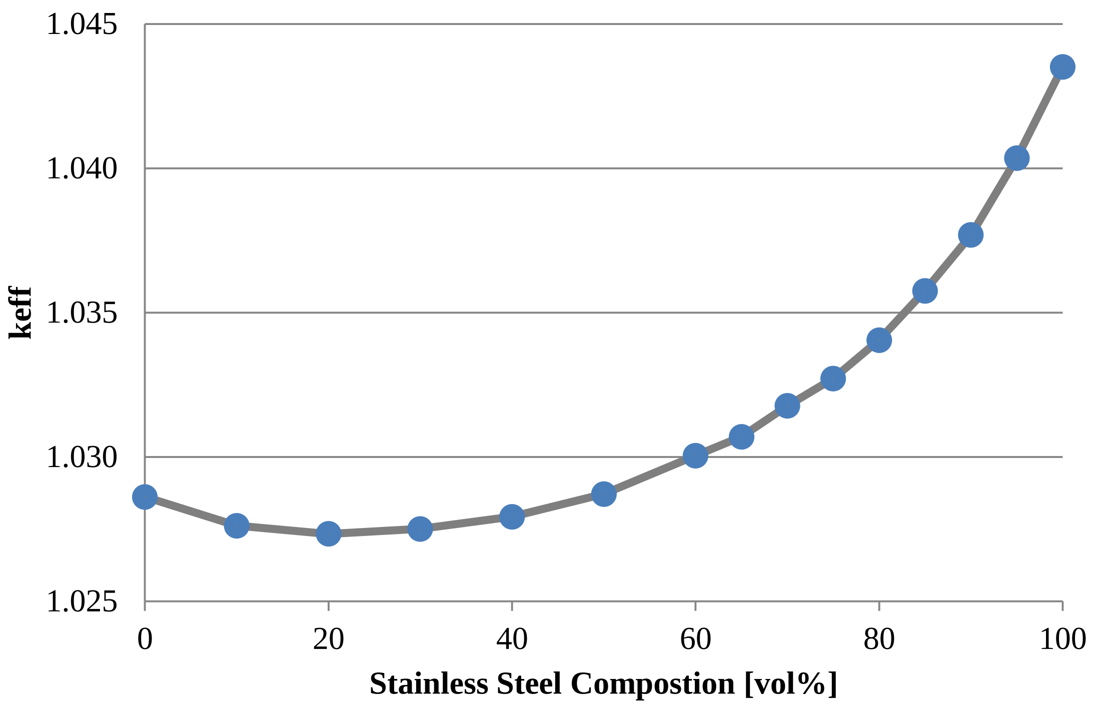


Figure 4.19 Reactivity impact of radial reflector material composition variation with 1000 ppm soluble boron concentration. Maximum reported error: 7 pcm

4.4.4 1500 ppm Soluble Boron Concentration

The results for the 1500 ppm case can be seen in Figure 4.20. Consistent with the other analyses, the highest reactivity is for the pure steel case.

All four plots are shown together in Figure 4.21. While each individual plot is less telling here than when shown separately, the general impact of soluble boron on the I^{2}S-LWR system can be observed. An interesting observation from all four plots is that even in the cases with significant soluble boron, pure coolant reflector performs better than 90/10 v% coolant/steel reflector. While this region is far away from the design space of the I^{2}S-LWR (~10/90 v% coolant/steel), it is a peculiarity which would cease to be true as boron continues to increase. At some point, a monotonic k_{eff} profile would be expected.

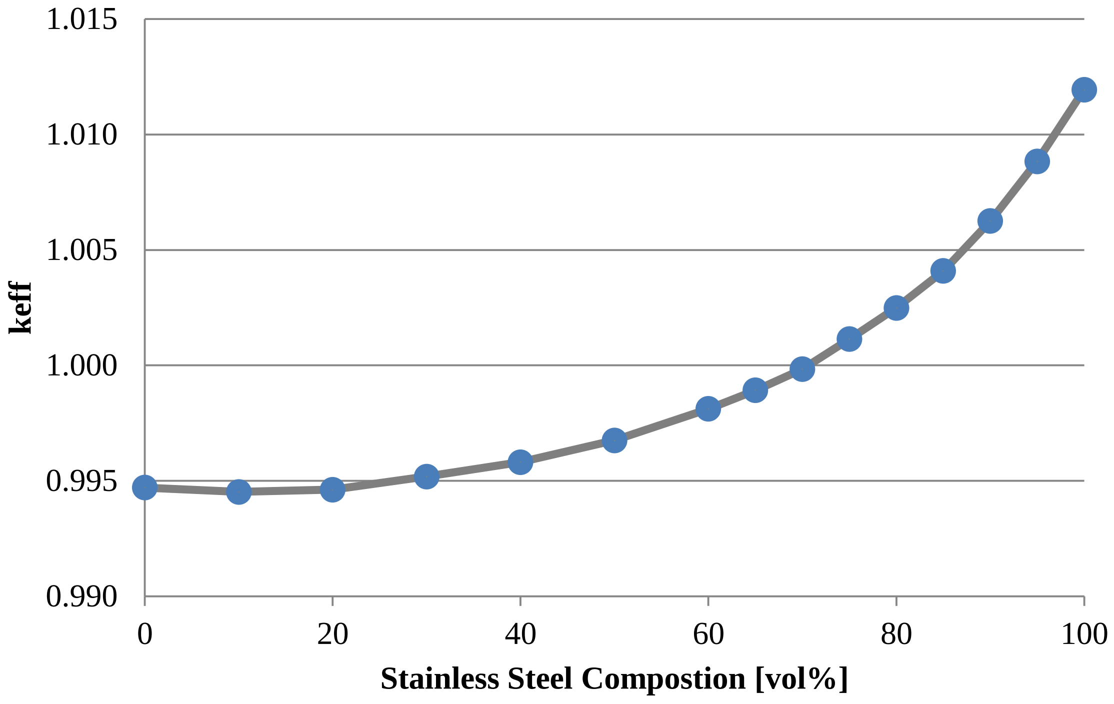


Figure 4.20 Reactivity impact of radial reflector material composition variation with 1500 ppm soluble boron concentration. Maximum reported error: 9 pcm.

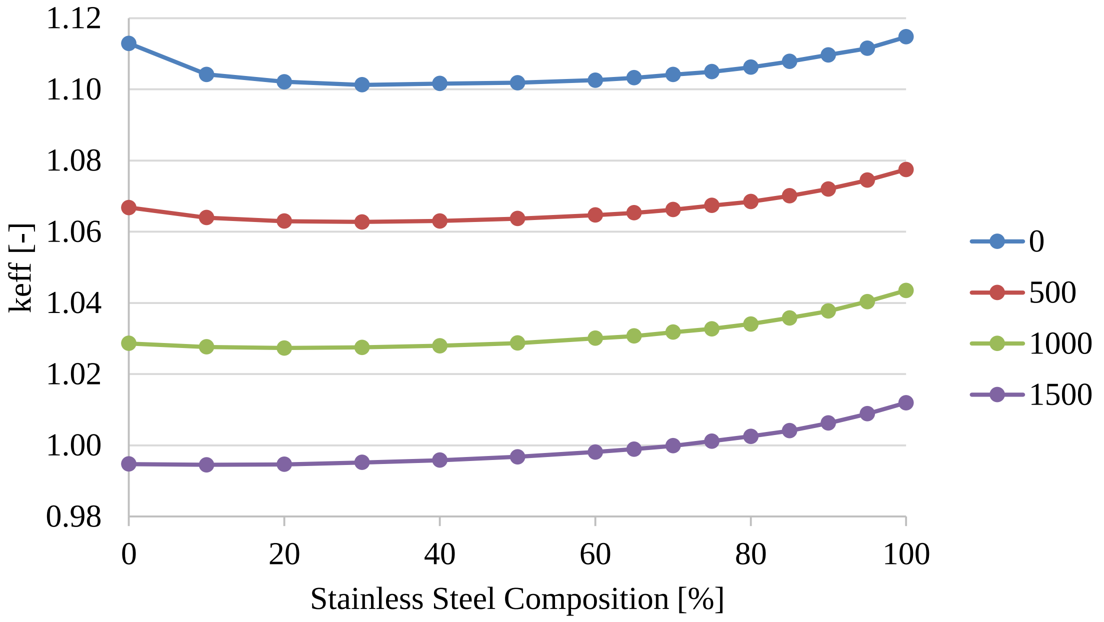
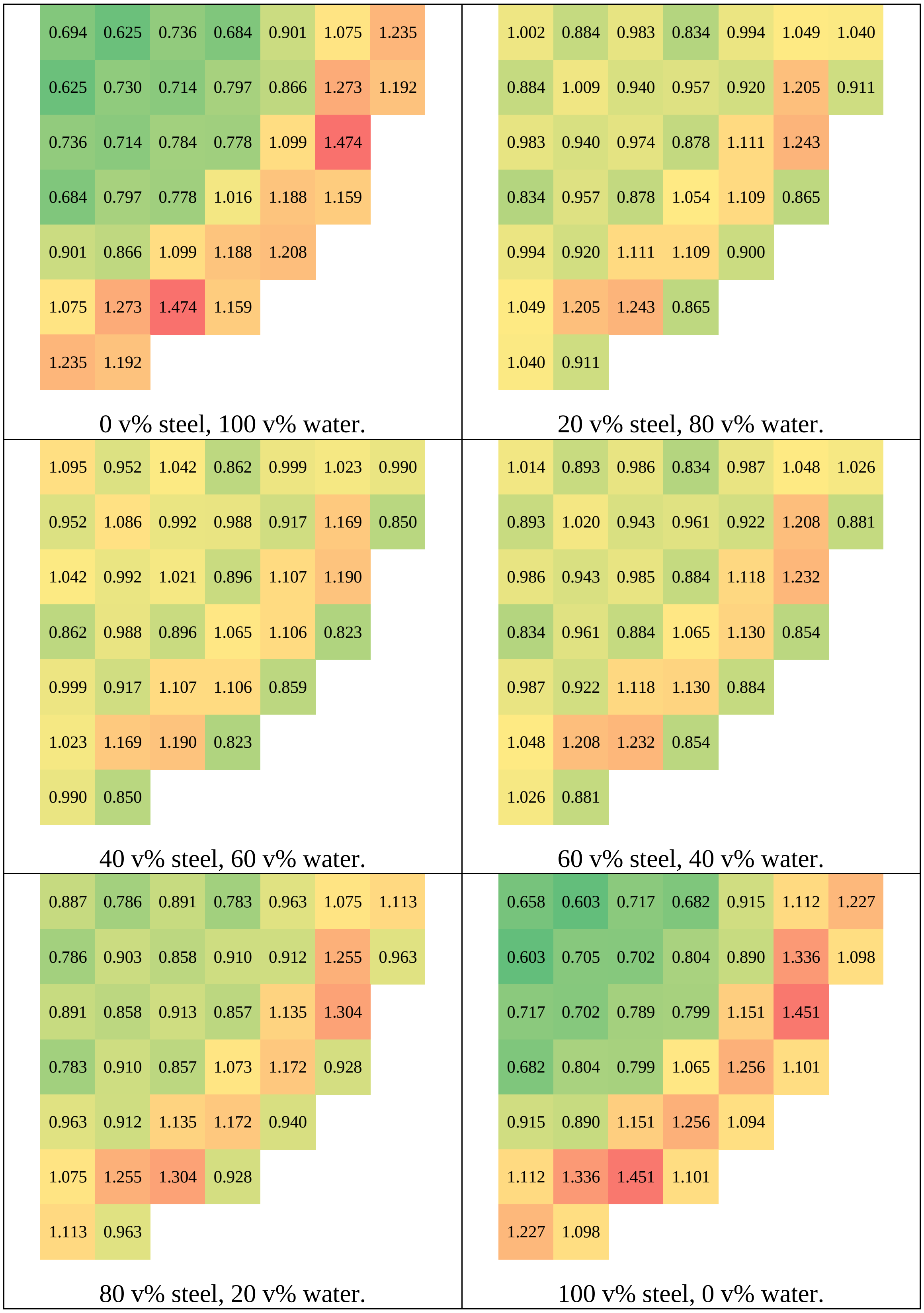


Figure 4.21 Reactivity impact of radial reflector material composition variation for various soluble boron concentrations.

4.4.5 Radial Power Distributions

Another notable behavior from these simulations is the effect of reflector composition on the radial power distribution. Below, find two sets of plots: one using 0 ppm soluble boron but varying the reflector composition (Figure 4.22) and another using 90 v% steel reflector but varying the soluble boron concentration (). The objective of these two sets is to analyze the independent impact of varying just one of the parameter of boron concentration and reflector composition.

Presented in Figure 4.22, one can see the sizable impact that the radial reflector has on the power distribution. As a note, each of these six power profiles corresponds with an eigenvalue datapoint in Figure 4.16. The power is noticeably concentrated near the periphery in the cases of pure water and pure steel. These results were anticipated, since in Figure 4.16 these had the largest eigenvalue. By the nature of having a larger eigenvalue, the two pure reflector compositions are more beneficial for core neutronics and promote more power near the reflector (i.e. at the periphery). From Figure 4.16, there is an obvious valley of less favorable neutronics conditions between the two pure composition cases. For mixtures of steel and water, one observes the expected result of the periphery of the core being less favorable and thus the power distribution migrates toward the center of the core. As a general observation to be applied to the 500, 1000, and 1500 ppm power distributions not shown here, one would expect that the higher the eigenvalue for a given reflector composition, the more power would be generated at the periphery.



0.694	0.625	0.736	0.684	0.901	1.075	1.235
0.625	0.730	0.714	0.797	0.866	1.273	1.192
0.736	0.714	0.784	0.778	1.099	1.474	
0.684	0.797	0.778	1.016	1.188	1.159	
0.901	0.866	1.099	1.188	1.208		
1.075	1.273	1.474	1.159			
1.235	1.192					

0 v% steel, 100 v% water.

1.002	0.884	0.983	0.834	0.994	1.049	1.040
0.884	1.009	0.940	0.957	0.920	1.205	0.911
0.983	0.940	0.974	0.878	1.111	1.243	
0.834	0.957	0.878	1.054	1.109	0.865	
0.994	0.920	1.111	1.109	0.900		
1.049	1.205	1.243	0.865			
1.040	0.911					

20 v% steel, 80 v% water.

1.095	0.952	1.042	0.862	0.999	1.023	0.990
0.952	1.086	0.992	0.988	0.917	1.169	0.850
1.042	0.992	1.021	0.896	1.107	1.190	
0.862	0.988	0.896	1.065	1.106	0.823	
0.999	0.917	1.107	1.106	0.859		
1.023	1.169	1.190	0.823			
0.990	0.850					

40 v% steel, 60 v% water.

1.014	0.893	0.986	0.834	0.987	1.048	1.026
0.893	1.020	0.943	0.961	0.922	1.208	0.881
0.986	0.943	0.985	0.884	1.118	1.232	
0.834	0.961	0.884	1.065	1.130	0.854	
0.987	0.922	1.118	1.130	0.884		
1.048	1.208	1.232	0.854			
1.026	0.881					

60 v% steel, 40 v% water.

0.887	0.786	0.891	0.783	0.963	1.075	1.113
0.786	0.903	0.858	0.910	0.912	1.255	0.963
0.891	0.858	0.913	0.857	1.135	1.304	
0.783	0.910	0.857	1.073	1.172	0.928	
0.963	0.912	1.135	1.172	0.940		
1.075	1.255	1.304	0.928			
1.113	0.963					

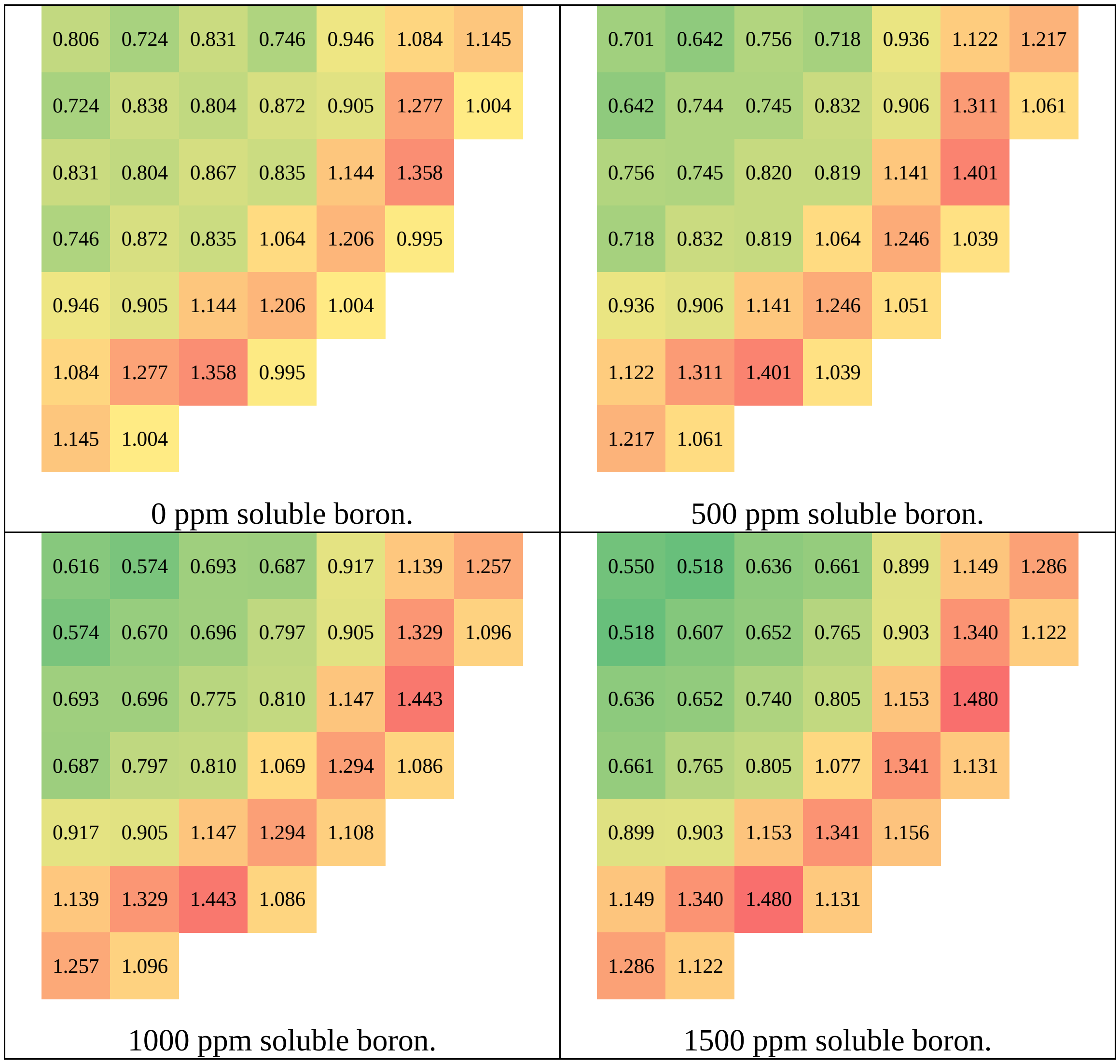
80 v% steel, 20 v% water.

0.658	0.603	0.717	0.682	0.915	1.112	1.227
0.603	0.705	0.702	0.804	0.890	1.336	1.098
0.717	0.702	0.789	0.799	1.151	1.451	
0.682	0.804	0.799	1.065	1.256	1.101	
0.915	0.890	1.151	1.256	1.094		
1.112	1.336	1.451	1.101			
1.227	1.098					

100 v% steel, 0 v% water.

Figure 4.22 Radial profiles for 0 ppm boron with varying radial reflector compositions.

Figure 4.23 shows the radial power distribution for the homogenized I²S-LWR reflector composition of 90 v% steel and 10 v% water for various soluble boron concentrations. An immediate observation when comparing this set of figures to the previous set is that soluble boron variation has less impact on the power distribution than changing the reflector composition. It is also evident that increasing soluble boron concentration causes the power to shift toward the periphery. This is an important note because as a core depletes and the CBC decreases over the cycle, one could expect that the power would migrate toward the center.



0.806	0.724	0.831	0.746	0.946	1.084	1.145
0.724	0.838	0.804	0.872	0.905	1.277	1.004
0.831	0.804	0.867	0.835	1.144	1.358	
0.746	0.872	0.835	1.064	1.206	0.995	
0.946	0.905	1.144	1.206	1.004		
1.084	1.277	1.358	0.995			
1.145	1.004					

0 ppm soluble boron.

0.701	0.642	0.756	0.718	0.936	1.122	1.217
0.642	0.744	0.745	0.832	0.906	1.311	1.061
0.756	0.745	0.820	0.819	1.141	1.401	
0.718	0.832	0.819	1.064	1.246	1.039	
0.936	0.906	1.141	1.246	1.051		
1.122	1.311	1.401	1.039			
1.217	1.061					

500 ppm soluble boron.

0.616	0.574	0.693	0.687	0.917	1.139	1.257
0.574	0.670	0.696	0.797	0.905	1.329	1.096
0.693	0.696	0.775	0.810	1.147	1.443	
0.687	0.797	0.810	1.069	1.294	1.086	
0.917	0.905	1.147	1.294	1.108		
1.139	1.329	1.443	1.086			
1.257	1.096					

1000 ppm soluble boron.

0.550	0.518	0.636	0.661	0.899	1.149	1.286
0.518	0.607	0.652	0.765	0.903	1.340	1.122
0.636	0.652	0.740	0.805	1.153	1.480	
0.661	0.765	0.805	1.077	1.341	1.131	
0.899	0.903	1.153	1.341	1.156		
1.149	1.340	1.480	1.131			
1.286	1.122					

1500 ppm soluble boron.

Figure 4.23 Profiles for 90 v% steel radial reflector with varying boron conentrations.

CHAPTER 5 3D I²S-LWR SERPENT MODEL

5.1 Overview

This section details work made toward scaling up the 2D Serpent Model addressed in the previous two chapters to a 3D model. This 3D Serpent Model was the goal from the outset of this research endeavor. With it, many studies could be conducted in the future on the I²S-LWR. One in-depth depletion analysis in particular has been completed using this model and will be covered in Chapter 6.

5.1.1 Objectives

The major objective of moving to a 3D model is to model axial phenomena. While the 2D Serpent Model was adequate for capturing some of the radial behavior of the I²S-LWR core, the 3D Serpent Model is necessary to analyze any of the axial trends. Coolant density and temperature variation became possible by being able to subdivide the core into several axial regions. Since the core physics of any LWR system is very sensitive to moderator properties, this improvement should enhance the global accuracy of the results over the 2D model.

5.1.2 Scope

While having more axial regions should in general produce more precise results, it is infeasible to subdivide the model into a very large number of segments due to memory and computational limitation concerns. Previous depletion studies [20] done in Serpent using the same computer cluster as the one for this research indicated that the maximum number of trackable burnable regions[23] was around 1000[24]. For this reason, ten axial partitions were chosen based upon the number of required burnable materials in each axial segment.

Additional approximations were made for the axial reflector. Instead of taking many different axial sections to compose the top and bottom axial reflectors, a single homogenized section at each end was used. While explicit detail in these regions would have been unnecessary, some consideration was placed into whether several homogenized axial sections (about seven at each end) might better represent the axial reflector. While this option was foregone for this endeavor, it remains available for consideration for future work.

5.1.3 Approach

Creating the 3D Serpent Model was fairly straightforward since it continued off of the 2D Serpent Model. The geometry in each section is largely the same save for small differences such as the application of IFBA or enrichment of fuel in the fuel blankets at the fuel rod top and bottom section. Thus, the 3D Serpent Model was created by essentially stacking the 2D Serpent Model upon itself several times with minor adjustments in each segment to produce the I²S-LWR core in 3D.

[23] A former graduate student at GT (Tim Wyant) found that the memory on the cluster used within the research group could support up to around 1000-1100 burnable materials that required depletion tracking. Depending on IFBA usage (7 axial sections with IFBA and 3 without), an axial slice could require tracking up to a maximum of 117 burnable materials and down to a minimum of 65.

[24] This limit of about 1000 burnable regions in Serpent only applies when using the most time-efficient, memory-intensive cross section optimization mode in Serpent. This mode stores the microscopic cross sections for each material, saving time later. Other modes could have allowed for additional axial partitions to be used by not storing these microscopic cross sections and instead calculating them on-the-fly for each reaction, but this would have hindered computational performance at the expense of improving the axial resolution. This was deemed as being undesirable at the current stage of I²S-LWR analysis and thus the number of axial partitions was capped at ten, requiring 117*7+65*3 = 1014 burnable regions.

5.2 Geometry

The 3D I²S-LWR Serpent Model shares many characteristics with the 2D I²S-LWR Serpent Model. For completeness, all relevant parameters will be explicitly included in this section despite model similarities and redundancies (especially for Section 5.2.1 Radial Layout) from Chapter 3 on the 2D model.

5.2.1 Radial Layout

The model is represented by a quarter of the core with reflective boundaries through the middle of the core to represent full core behavior. Past the downcomer region, the geometry ends and non-reentrant (vacuum) boundary conditions are assumed. The assemblies are modeled at the quarter assembly level. So, for a quarter core model of a 121 fuel assembly system, there are 121 quarter assemblies in the 3D Serpent model.

Another way to decrease the number of burnable materials which require tracking is to exploit symmetry even further- using eighth-core symmetry. Instead of tracking 121 quarter assembly fuel regions per axial segment, that number was reduced to 65 by applying eighth core symmetry. These savings increase for the more complicated axial sections with IFBA; decreasing from 225 burnable materials to 117.

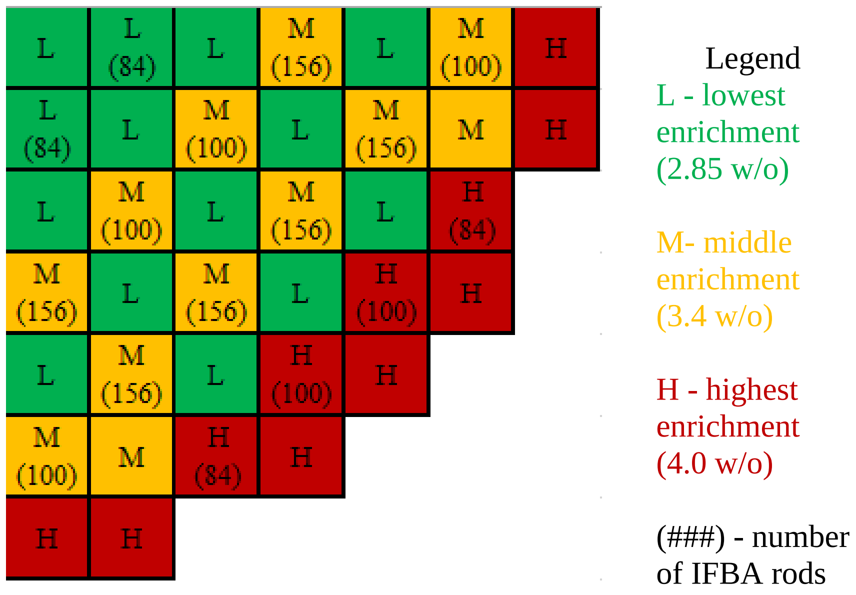
The radial reflector is assumed to be 90% APMT steel by volume. This is an approximation to account for the flow channels through the radial reflector. The composition (density and temperature) of the water making up the remaining 10% by volume used in each of the ten axial fuel sections of the radial reflector varies axially similarly to the coolant. While the actual temperature profiles will vary among the flow channels and additionally will differ from the core profiles, it was assumed that the coolant in the homogenized radial reflector of each axial section is at the average core coolant temperature for that axial section.

The exact dimensions used in the 3D Serpent model can be found in Table 5.1. The IFBA coating thickness selection criteria remained the same: based on assuming a ZrB_2 density of 6.08 g/cm^3, requiring the ^{10}B content to be 2.5 mg/in, and assuming that the boron is enriched to 60 a% ^{10}B. No changes were made to parameters below (identical to Table 3.1).

Table 5.1 Radial geometry parameters for the 3D Serpent Model

Quantity [unit]	**Value**
Solid fuel outer radius (diameter) [cm]	0.40513 (0.81026)
IFBA coating outer radius (diameter) [cm]	0.4057226 (0.8114452)
Outer gas gap radius (diameter) [cm]	0.41656 (0.83312)
Cladding outer radius (diameter) [cm]	0.4572 (0.9144)
Fuel pin pitch [cm]	1.21006
Fuel assembly layout [-]	19x19
Fuel pins per assembly [-]	336
Guide tubes per assembly [-]	24
Instrumentation tubes per assembly [-]	1
IFBA fuel pins per assembly [-]	0, 84, 100, 156
Non-IFBA fuel pins per assembly [-]	336, 252, 236, 180
Fuel assembly pitch [cm]	23.1013
Interassembly gap [cm]	0.11016
Reflector outer radius (diameter) [cm]	160 (320)
Core barrel outer radius (diameter) [cm]	165 (330)
Downcomer radius (diameter) [cm]	245 (490)

As with the 2D Serpent Model, three fuel enrichments are also used in the 3D Serpent model. They are also arranged in an “out-in” pattern, where the fuel with the highest enrichment is on the periphery and decreasing in enrichment when moving to the center. The core loading pattern with applied IFBA pattern can be seen in Figure 5.1. The enrichments selected do not match those used in the reference I²S-LWR design. They were chosen based on the results of the prior analysis of the core loading pattern discussed in Chapter 3. Parameters which were kept in mind while creating this fueling scheme included having an approximately 12-month cycle length[25,26], having the AO close to zero (power centered near the midplane), as well as keeping the hot channel factors F_q, F_z, and $F_{\Delta H}$ fairly low over the cycle.



L	L (84)	L	M (156)	L	M (100)	H
L (84)	L	M (100)	L	M (156)	M	H
L	M (100)	L	M (156)	L	H (84)	
M (156)	L	M (156)	L	H (100)	H	
L	M (156)	L	H (100)	H		
M (100)	M	H (84)	H			
H	H					

Legend
L - lowest enrichment (2.85 w/o)

M- middle enrichment (3.4 w/o)

H - highest enrichment (4.0 w/o)

(###) - number of IFBA rods

Figure 5.1 Core loading pattern used for the 3D Serpent Model.

[25] For a 12 month (nominally 365 day) fuel cycle, the actual time that the core is running at power is less than 365 days. The extra time is to account for refueling, maintenance, and other work that is performed during a scheduled outage. For the I²S-LWR, it is estimated that these activities could all be completed within 17 days [26]. This means that for a 12 month cycle, the target cycle duration is about 348 days.

[26] Even though this design is targeting a 12 month fuel cycle, there is usually some tolerance or even desire for the first cycle to be slightly longer. This is due to the fact that about a third of the fuel assemblies in the core will only be used for a single fuel cycle before being permanently discharged. In order to get better fuel utilization out of the first fuel cycle in particular, running longer than 12 months may be an economically advantageous consideration.

Note that enrichments in most regions and IFBA application on two assemblies have changed compared to the 2D model core loading pattern of Figure 3.3. Fuel enrichments were changed slightly from a 4.0/3.0/2.5 w% scheme to 4.0/3.4/2.85 w% to reduce the radial peaking factor challenges encountered in the 2D model and increase the cycle length. Although the results of this 3D model will not be discussed until Chapter 6, the core loading pattern actually worked quite satisfactorily radially. To further flatten radial peaking factors, one group of assemblies[27] which previously did not have any IFBA applied to it also received 84 IFBA pins.

5.2.2 Axial Partitioning

For this fresh core, the same cutback fuel design could not be used as the equilibrium I²S-LWR core (Figure 2.2). In the equilibrium system which has a slightly negative AO (as most PWR systems do), the higher flux in the bottom half of fresh assemblies depletes these regions more than the top. After the fuel is shuffled, the assemblies which have been burned multiple times (once, twice, and thrice) have more reactivity in the top half since the bottoms have been depleted more. While this is not enough to counteract the axial reactivity impact of the coolant temperature and density variation moving up the core, it can help push the AO closer to zero. For an entirely fresh core with an axially-symmetry core loading pattern, this benefit from the previously burned assemblies is not present. To compensate for this and to avoid a large negative AO, only a single cutback fuel region of 7.5 inches is placed near the top of the core, while there is none at the bottom as opposed to Figure 2.2.

[27] Eight assemblies in total, two of which are shown in Figure 5.1. These are the highest enrichment (red) assemblies on the periphery with 84 IFBA pins.

As addressed previously for computational speed and memory overhead reasons, ten axial sections were used for the active fuel region of the core. One is used for each of the 6-inch fuel blanket regions at the top and bottom of the core. A third is used for the 7.5-inch cutback region near the top of the core. The remaining seven sections are used for the fuel regions with IFBA. A visualization of this can be seen in Figure 5.2.

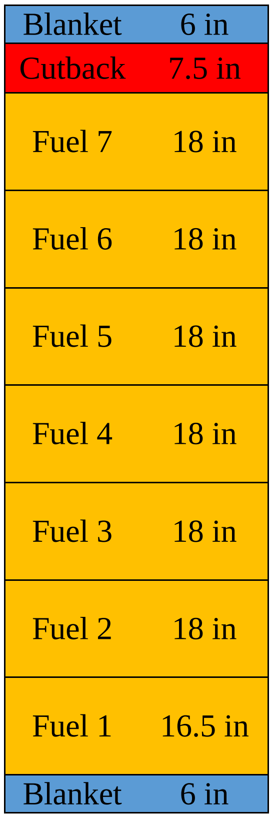


Figure 5.2 Axial partitioning used in the 3D Serpent model.

The reason why the step size varies in the “Fuel 1” segment versus the other IFBA-coated fuel regions in Figure 5.2 is to ease the transition between the blanket and fuel regions. One argument against this approach is that since one would normally expect the axial power profile (and thus the linear heat rate) to be highest near center, finer segmentation should be used there to capture the coolant temperature and density variation more accurately. Instead, the approach taken is to simply try to use a uniform step size across the seven IFBA fuel regions. The discrepancy in “Fuel 1” at 16.5 in compared to the other 18 in “Fuel X” regions is to account for the 7.5 in cutback fuel region at the top of the core.

The axial reflectors are both assumed to be 12 inch (30.48 cm) sections composed of 30% APMT steel by volume, with the bottom axial reflector having coolant at inlet conditions (298 °C) and the top axial reflector having coolant at outlet conditions (330.5 °C). Further refinement is possible for the future which would treat the axial reflector as several different homogenized layers instead of just a single homogenized layer.

5.3 Materials

A list of material compositions used can be found in Table 5.2. The temperatures listed for each material are either a best-guess based on the axial average value or, in the case of any material containing water, vary axially. Some axially-varying materials (such as water) have been omitted from Table 5.2 since listing them is space prohibitive. Even without listing the materials containing water, it should be noted that all such materials are water- APMT stainless steel mixtures. In-core coolant is actually assumed to be 99.4 vol% water to account for the homogenization of grid spacers (which are not explicitly modeled). Reflector compositions contain APMT and 10 vol% water for the radial direction and 70 vol% water for the axial direction. As a note, ^{234}U enrichment is assumed to be 0.04 w% regardless of ^{235}U/^{238}U content. The temperature of helium being 750 K was selected as the average temperature between the inner cladding and outer fuel for all fuel regions. The APMT temperature of 600K was selected as the average cladding temperature for all fuel regions.

Table 5.2 List of materials used in the 3D Serpent Model

Material	**Mass Density [g/cm^3] (unless otherwise given)**	**Temperature [K]**
U_3Si_2 theoretical density	12.2	-
Assumed fuel density fraction	0.972 [-]	-
Pellet dishing volume fraction	0.012 [-]	-
U_3Si_2 with effective 96.0336% theoretical density	11.7161	-
4.0 w% Fuel (U_3Si_2)	11.7161	Varies
Si	0.864303	
^{234}U	0.004341	
^{235}U	0.434072	
^{238}U	10.413384	
3.4 w% Fuel (U_3Si_2)	11.7161	Varies
Si	0.864303	
^{234}U	0.004341	
^{235}U	0.368961	
^{238}U	10.478495	
2.85 w% Fuel (U_3Si_2)	11.7161	Varies
Si	0.864303	
^{234}U	0.004341	
^{235}U	0.309276	
^{238}U	10.538180	
2.5 w% Fuel (U_3Si_2)	11.7161	Varies
Si	0.864303	
^{234}U	0.004341	
^{235}U	0.271295	
^{238}U	10.576161	
IFBA (ZrB_2, 60 a% ^{10}B enrichment)	6.08	Varies
^{10}B	40 a%	
^{11}B	26.666667 a%	
Zr	33.333333 a%	
Helium Gas Gap (200 psi at 20 C)	0.002264	750
He	100 a%	

Table 5.2 continued.

Material	**Mass Density [g/cm^3] (unless otherwise given)**	**Temperature [K]**
APMT Advanced Steel	7.25	600
C	0.08 w%	
Al	5 w%	
Si	0.7 w%	
Cr	21 w%	
Mn	0.4 w%	
Fe	69.82 w%	
Mo	3 w%	

5.4 Methodology Implemented for Analyses

The methodology used to analyze the depletion of a core in the 3D Serpent model is described below. It couples neutronics and thermal-hydraulics and focuses on axial coolant variation and critical soluble boron concentration usage between burnup steps. While most results from the actual depletion analysis are reserved for the next chapter, some are also provided here where appropriate to better explain and clarify what is stated in the text.

5.4.1 Axial Temperature Variation

To provide thermal hydraulic feedback for Serpent's neutronics output, a Single Channel Analysis (SCA) code was written in C++ (Appendix A) to return temperatures for the coolant, fuel, and IFBA materials as well as the coolant density compositions for iterating on the power profile.

The Serpent code tracks results axially over 96 1.5 in (covering the 144 in active fuel length) axial tally segments and radially for 121 quarter-fuel assemblies (in the quarter core model), resulting in 11616 total tally regions, to produce the power profile data to be fed into the C++ SCA script. A thermal hydraulic SCA iteration is run on each of the individual 121 quarter assembly regions until the profile converges on the prescribed inlet/outlet conditions of 298 °C and 330.5 °C, respectively. The free variable in this process is the total coolant mass flow rate through the core, which the code ultimately gives once converged. The thermal hydraulic properties of these 11616 locations are collapsed (by averaging) into the ten axial sections specified in Figure 5.2 and 65 eighth-core symmetry groups. In these 650 larger regions; 182 IFBA materials, 650 coolant materials, 650 non-IFBA-coated fuel materials, and 182 IFBA-coated fuel materials are produced in Serpent input format. These materials can then be used in the Serpent model (Appendix A) to provide the iterating capabilities necessary for a thermal hydraulically-coupled 3D Serpent model.

While the thermal conductivity of unirradiated uranium silicide at various temperatures is known, conductivity performance with burnup is not available at the writing of this thesis. Therefore, the unirradiated thermal conductivity profile is used over the entire fuel cycle (invariant with respect to burnup). An alternative approach could have been to assume that the burnup conductivity behavior of U_3Si_2 is similar to UO_2, which is well studied. The downside to using the alternative approach would be complicating the analysis for the sake of using a comparison to an entirely different material’s behavior, which could ultimately have been wrong. So instead, the thermal conductivity of unirradiated U_3Si_2, $k(T) = 7.98 + 0.0051T$; where T is in °C and k is in W/(m*K), was used for all burnup steps in order to find the centerline fuel temperatures in the thermal hydraulic model [14].

Similar to the fuel, the assumptions and parameters of thermal models for the gas gap and cladding remained the same over the fuel cycle (invariant with burnup). Thus, the gas gap model did not consider the addition of gaseous fission products nor the potential for fuel-cladding contact due to swelling and the cladding model did not include neutron fluence damage as a factor. Conductive and radiative heat transfer were considered for the gas gap, with the gas conductivity given by $k(T) = 0.00158T^{0.79}$, where T is in K and k is in W/(m*K). The thermal conductivity of the cladding is just that of APMT stainless steel: $k(T) = 10.318 + 0.016003T$; where T is in °C and k is in W/(m*K). It should be noted that this data was taken from a document prepared for materials specifically used in the I²S-LWR design [14].

5.4.2 Depletion

As any conventional LWR system depletes, the reactivity of the core changes[28]. In order to maintain criticality, reactivity control needs to be adjusted to compensate for the changes due to depletion. In the I²S-LWR and other PWRs, this is achieved by adjusting the soluble boron concentration. This section details the methodology used to model soluble boron during the depletion of the core.

[28] Generally, the reactivity of the core decreases over the cycle for an LWR. Periods of increase are also possible if burnable poisons are depleted faster relative to fuel. This typically only happens near the beginning of the cycle when the most poison is present in the core.

5.4.2.1 Finding the Critical Boron Concentration

In order to find the initial CBC at BOC, iterations had to be done between running Serpent simulations and using the SCA script previously described. Initially for the BOC, a uniform axial temperature distribution was assumed and about four iterations were used to converge to good agreement between the neutronics and thermal hydraulics. Although a desirable modeling methodology for future studies, no iteration was done for subsequent burnup steps because it was felt that by running with a fine enough step size, the power profile change from step to step would be small, and simply using the power profile at the end of each step would be sufficient. So, the boron worth from previous steps was used to estimate the CBC at the start of the next depletion simulation. The calculated CBC at each burnup step can be observed in Figure 5.3.

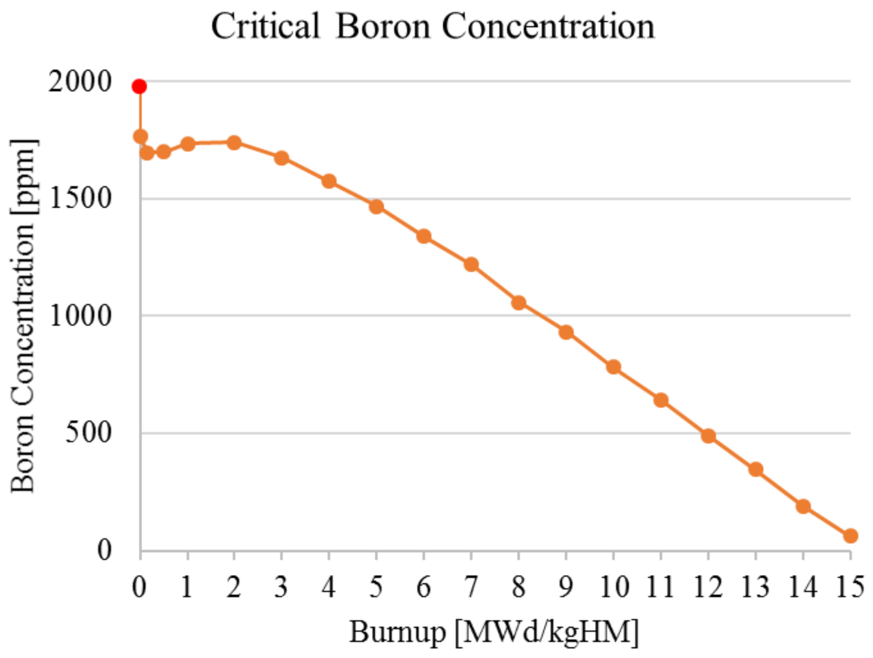


Figure 5.3 Critical boron concentration over the fuel cycle.

The red data point in Figure 5.3 is for the HZP (no power, inlet conditions everywhere, no depletion) case. BOC (full power, no depletion) is the data point just below it. The difference between the two is 213 ppm soluble boron, which is close to what one might normally expect (about 200 ppm) for existing PWR designs. The ENDF/B-VII library used for this work contains data with listings in 300 K increments from 0 K to 1200 K. For the HZP case, both the fuel and coolant are assumed to be at 564.8 K. The cross sections are Doppler broadened from 300K, since Serpent can only Doppler broaden from a lower temperature (no "Doppler narrowing" from a higher temperature). For the other HFP cases, fuel is at its thermal-hydraulically-iterated temperature always referencing the 600 K library (although some elements go over 900 K). Coolant is also at its thermal-hydraulically-converged temperature and density, with cross sections always broadened from 300 K even though some materials are above 600 K.

A reason for using only one cross section reference temperature (as opposed to using the closest) for each individual material is for one simplicity of the model; even though applying logical references would be easily implementable (i.e. using 300 K as the reference if above 300K, using 600 K as the reference if above 600 K, etc.). A second reason is for consistency between iterations within the model. Suppose a material is near the reference cutoff temperature of 600 K and that while the results for 599.9 K and 600.1 K should be nearly identical, the actual calculated results between broadening 299.9 K from 300 K and broadening 0.1 K from 600 K might be quite different. Although these values are expected to be agreeable; it was unknown if the data for all involved isotopes in fuel, fission products, etc. would be self-consistent near the reference temperature boundaries. Differing values may have resulted in a poorly-converging thermal hydraulic model by flip-flopping above and below the temperature threshold between iterations. To both avoid this potential issue and for ease, a single cross section reference temperature was used for each individual material.

Notice that after dropping significantly due to the build-in of fission products, the CBC actually increases during the cycle for burnups of 0.5 and 1 MWd/kgHM. As alluded to in the beginning of this section, this is due to the IFBA coating burning appreciably faster than the fuel, resulting in a net positive reactivity change in the core as a result of the depletion. After this period of increase, the CBC begins to drop again and does so fairly linearly until EOC.

5.4.2.2 Depleting between Burnup Steps

The CBC values used during the depletion analysis was the CBC at the start of each burnup step. Since the depletion process assumes a constant soluble boron concentration between steps and only the initial CBC is known in the current implementation[29], each burnup step was carried out using the initial CBC of that step. While more sophisticated techniques to be used with Serpent are detailed in literature [21], this method was chosen for ease of implementation. In terms of neutronics code implementation, the Predictor/Corrector method was used in Serpent. The soluble boron concentration that is effectively used over the depletion cycle can be seen in Figure 5.4.

[29] Potentially, a guess for the ending CBC could have been made from prior simulations (an unborated depletion was run before the proper borated analysis to get an idea from the excess reactivitity how the CBC would change over cycle) or if additional runs were used for each step. To account for both concentrations in future depletion studies and something to be seriously considered later, the average soluble boron concentration would have been a more accurate treatment of the CBC between each pair of burnup steps.

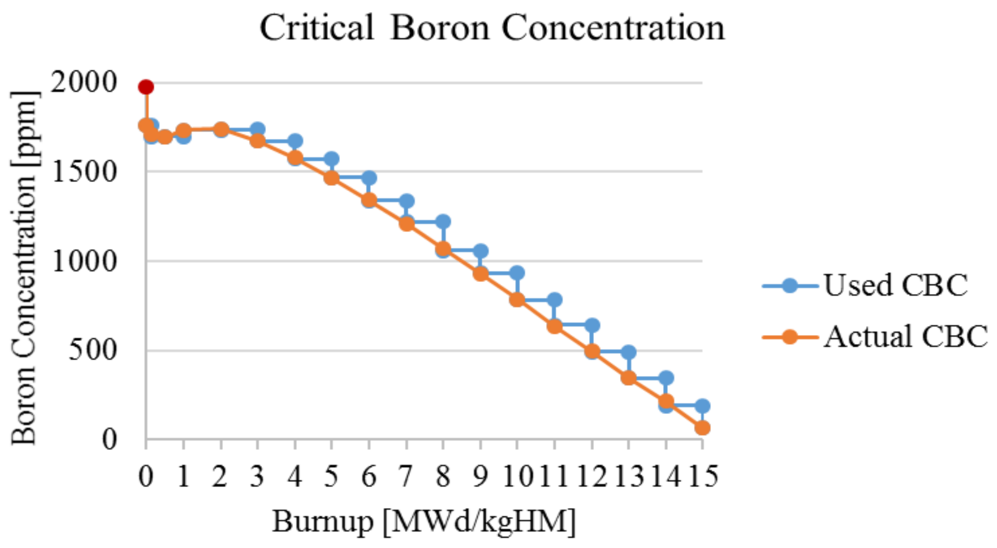


Figure 5.4 Plot of both the actual and used critical boron concentrations.

The actual[30] CBC shown in Figure 5.4 is the same profile shown in Figure 5.3, but included here again for comparison. As can be seen by the blue "Used CBC" plot, the used soluble boron centration is generally[31] higher than the actual CBC over the course of each depletion step. This was expected from only using the initial CBC.

5.5 Equilibrium Xenon Concentration Treatment

Prior to the depletion analysis discussed in Chapter 6, another depletion study was carried out with the I²S-LWR 3D model that used explicit xenon treatment. The reason this section is included in the thesis is to serve as an explanation as to why the equilibrium xenon feature in Serpent is used for the depletion simulations in Chapter 6. It also

[30] "Actual" here does not refer to experimental or verified results, but the calculated CBC over the cycle. It was chosen as the most concise way to differentiate it from the Used CBC set of values.
[31] The opposite is true in the region near the beginning of the cycle where the CBC increases due to the relatively rapid depletion of IFBA. This is difficult to see from the plot since the relative boron concentration is so slight, but worth mentioning for completeness.

demonstrates a “lesson learned” for knowing what the tool one uses is actually doing—understanding fission product poisons in Serpent depletions.

5.5.1 Characterizing the Use of Explicit Xenon Treatment in Depletion

While this older study obtained promising preliminary results, burnup steps further along in the depletion cycle produced very suspicious power profiles. As can be gathered from Figure 5.5, the axial performance of that analysis produced heavily peaked results. A curiosity was that the radial power profile seemed to behave as expected, so the trouble with the simulation rested solely with the axial results. While the reason for this anomaly was initially unknown, various tests ultimately pinned the behavior to how Serpent treated xenon-135, the most impactful fission product poison encountered in power reactor operation.

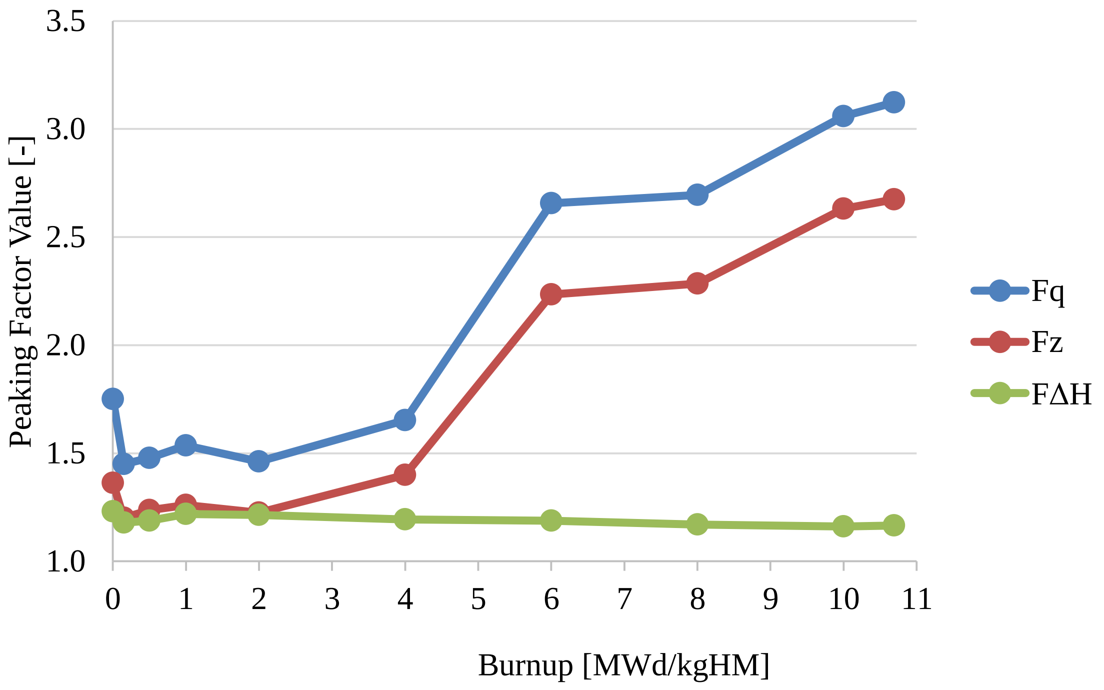


Figure 5.5 Power peaking factors for an older I²S-LWR depletion analysis which used explicit xenon treatment.

To better characterize and understand this unusual axial behavior, a simple reflected quarter assembly model was developed to aid in correcting the issue. This model used the I^{2}S-LWR assembly geometry and materials with no IFBA. Fuel enrichment was 2.6 w% and the standard 6 inch 2.5 w% blankets were also used at the top and bottom. Temperature and coolant properties did not vary axially, making the model axial symmetric. From this, it was expected that the axial power profile results would also be symmetric; implying in particular that all results obtained from the symmetric model should have zero AO.

However, this was not the case. Figure 5.6 shows how the AO progressed for a depletion analysis that did not use the predictor/corrector method. While the first depletion steps produced expected results, later steps also had the anomalous axial behavior (as with the older I^{2}S-LWR depletion).

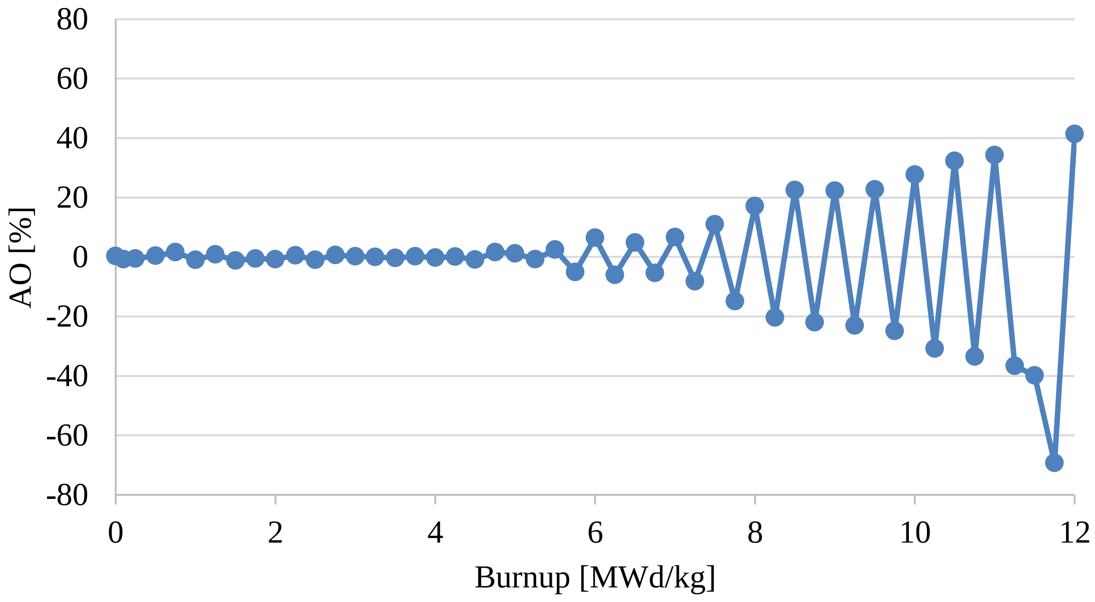


Figure 5.6 Axial offset for depletion using explicit xenon without predictor/corrector.

Notice how once the behavior diverges from 0% AO around 5.5 MWd/kg, the AO alternates between being positive and negative between burnup steps. Also notice that the magnitude of AO tends to augment once the divergent behavior starts. Both of these phenomena are induced by flux tilts due to artificial xenon oscillations. While xenon oscillations are a real phenomenon experienced in operating reactors, the distinction needs to be made that these anomalous results rest on a purely stochastic basis.

Actual xenon oscillations occur when there are regions of high/low neutron flux. In the areas of high flux, poisons are eliminated at a larger rate than in the area of lower flux. Additionally, the high flux regions experience a higher density of fissions and the opposite is true in the low flux regions. As time progresses, some fission products (namely xenon-135 precursors) decay into strong neutron poisons. Since the density of poisons will be proportional to density of prior fissions, the once-high flux regions will have more neutron poisons than the once-low flux regions and power will migrate to be higher in the former low flux region and lower in the former high flux region. At this stage, the process has arrived right where is started except with the regions reversed, whereas now the cycle will repeat and continue to oscillate from high/low. In actual practice, these oscillations can be dampened so they die out quickly. However, as seen with artificial stochastic xenon oscillations, augmentation of the oscillating behavior is also possible.

Since the model used to produce Figure 5.6 was axial symmetric, any and all results should also be symmetric up to statistical uncertainties from the nature of using Monte Carlo. However, it is these uncertainties that ultimate create the strong axial tilts observed. Consider the initial burnup step starting with an axially-symmetric fresh fuel loading. After running neutronics, a symmetric profile is produced with minor uncertainties in the power distribution. By the nature of these uncertainties, while some might balance others out, there should be some net direction (top or bottom) that observed more fission histories. In the next depletion step, this region with higher fission sampling will have a slightly higher density of fission product poisons. For the first few depletion steps, this will go on seemingly unnoticed. However, at a certain point, several steps' worth of statistical uncertainty will compound with each other to artificially produce an observable tilt in the axial power distribution. In Figure 5.6, this corresponds to the behavior starting around 5.5 MWd/kg. The magnitude increases between cycles because once the behavior starts, extra poison from the previous high flux region pushes the power even more than in that last step.

The two aspects of alternating behavior and magnitude growth will also become important when looking at Figure 5.7, which shows the case using predictor/corrector.

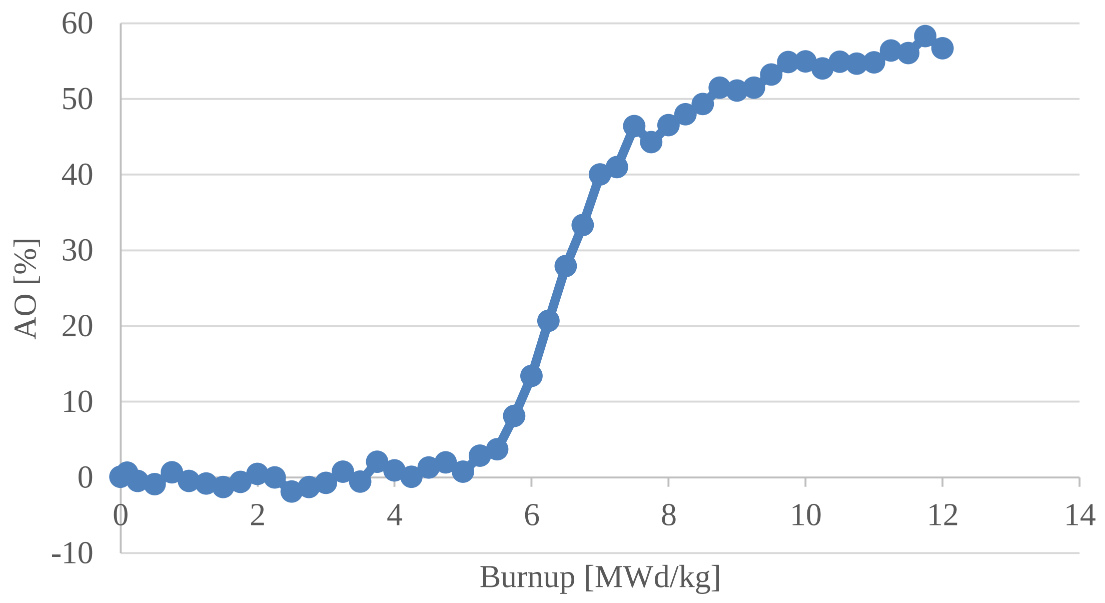


Figure 5.7 Axial offset for depletion using explicit xenon with predictor/corrector.

The only substantial difference between the depletion simulations in Figure 5.6 and Figure 5.7 is the use of predictor corrector. In Figure 5.7, notice that once again the results behave as expected until about 5.5 MWd/kg. At this point, the results diverge, but in a different manner than the case without predictor/corrector. With it, Figure 5.7 diverges in a single direction[32]. The reason for the non-alternating behavior is that the predictor/corrector for a single step can be regarded like a pair of depletions when not using predictor/corrector. Suppose that the predictor step is top-peaked with the anomalous behavior. For the corrector step, higher flux in the top during the predictor step will have driven the power toward the bottom. This in turn will cause the predictor for the next burnup step to also be top-peaked, just like the starting predictor step except likely larger in magnitude. This is why the results in Figure 5.7 appear to be non-alternating, because the reported power profiles from the predictor steps have consistent tilts with each other.

5.5.2 Characterizing the Use of Equilibrium Xenon Treatment in Depletion

When using explicit xenon treatment in Monte Carlo, the source distribution changes between cycles in a depletion step but the material compositions do not. So while the source distribution essentially has neutronics feedback between each cycle, a skewed fission product inventory will produce tilted results in each cycle resulting in the entire depletion step itself being skewed. To accommodate this materials issue, Serpent has a feature to enforce the equilibrium xenon concentration in each fissile material between

[32] Replica simulations of this depletion were carried out and divergence toward the top of the core and the bottom of the core were both observed (all positive or all negative AO in anomalous divergent region). The results in Figure 5.7 show a top-preferring case, but these only highlight the instability of the axial performance and do not suggest that the divergence always go to the top; merely that the divergence occurs.

each cycle. By using Equation 6 and Equation 7 for the concentrations of the principal precursor ^{135}I (n_I) and the ^{135}Xe itself (n_{Xe}), Serpent calculates the concentration of each of the two isotopes between all active and inactive cycles [22].

$$n_I = \frac{\gamma_I \Sigma_f \phi}{\lambda_I} \qquad \text{Equation 6}$$

$$n_{Xe} = \frac{\gamma_{Xe} \Sigma_f \phi}{\lambda_{Xe} + \sigma_{Xe} \phi} \qquad \text{Equation 7}$$

Where γ is the cumulative fission yield for each isotope, λ is the decay constant for each isotope, Σ_f is the macroscopic total fission cross-section of the material, σ_{Xe} is the microscopic capture cross-section of xenon-135, and ϕ is the total flux. By having the xenon concentration updated between each cycle, the anomalous tilting behavior is suppressed since any oscillations are likely to cancel with each other over all active cycles to produce symmetric (for this case) results.

The simulations in Section 5.5.1 were repeated using equilibrium xenon treatment instead of explicit xenon treatment. First, Figure 5.8 shows the case without predictor/corrector. This should be compared with Figure 5.6.

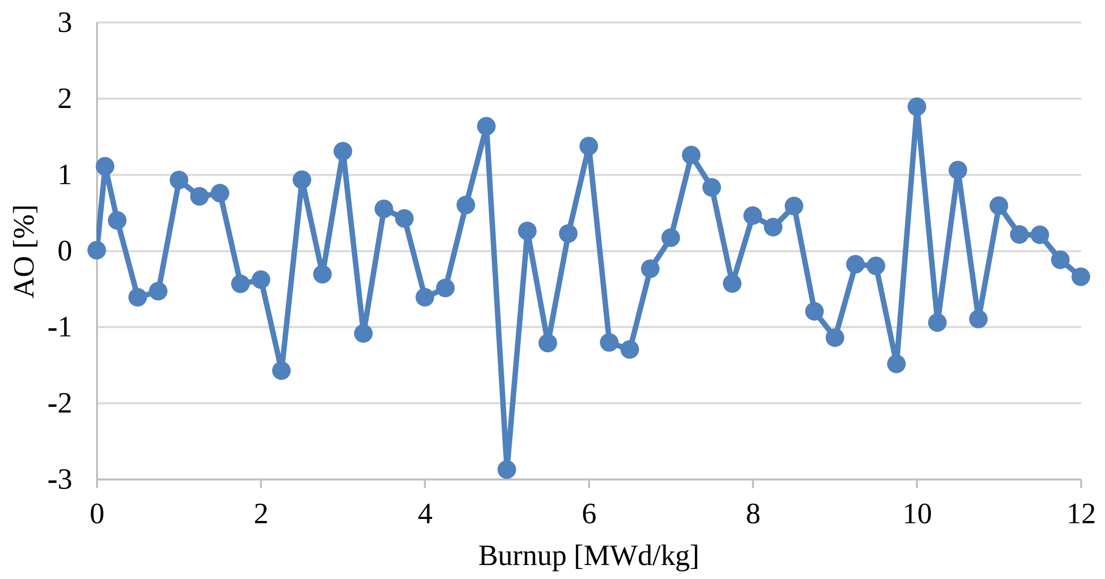


Figure 5.8 Axial offset of depletion using equilibrium xenon without predictor/corrector.

While the results of Figure 5.8 might at a glance seem poor, looking at the vertical scale tells instead that the AO values are actually rather close to zero over the cycle, with no general trend of top/bottom preference. This is what one would have expected from depleting a symmetric model. Figure 5.9 shows similar results and should be compared with Figure 5.7 for cases both using predictor/corrector.

A plot of all four AO profiles together can be seen in Figure 5.10. From this figure, it is easier to appreciate the level of improvement seen by using the equilibrium xenon feature in Serpent versus using explicit xenon treatment. As a closing remark, equilibrium xenon is suggested for any 3D depletion study (except for BOC when no xenon is present). For 2D studies, the creator of the Serpent code suggests that explicit xenon treatment may be adequate since results from those simulations did not see appreciable improvement despite the increase in computational demand [22].

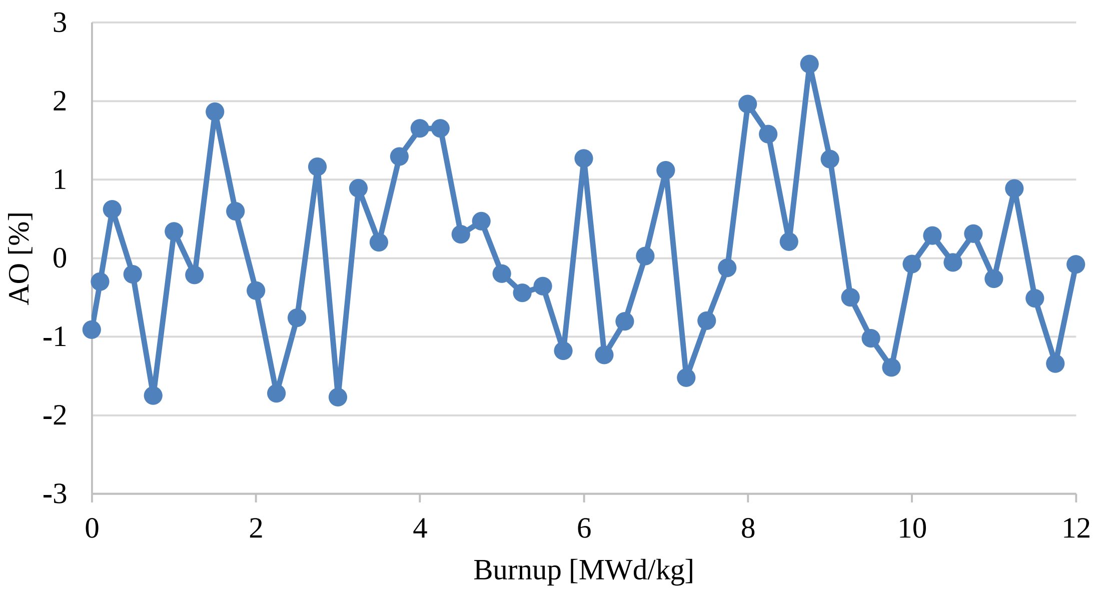


Figure 5.9 Axial offset of depletion using equilibrium xenon with predictor/corrector.

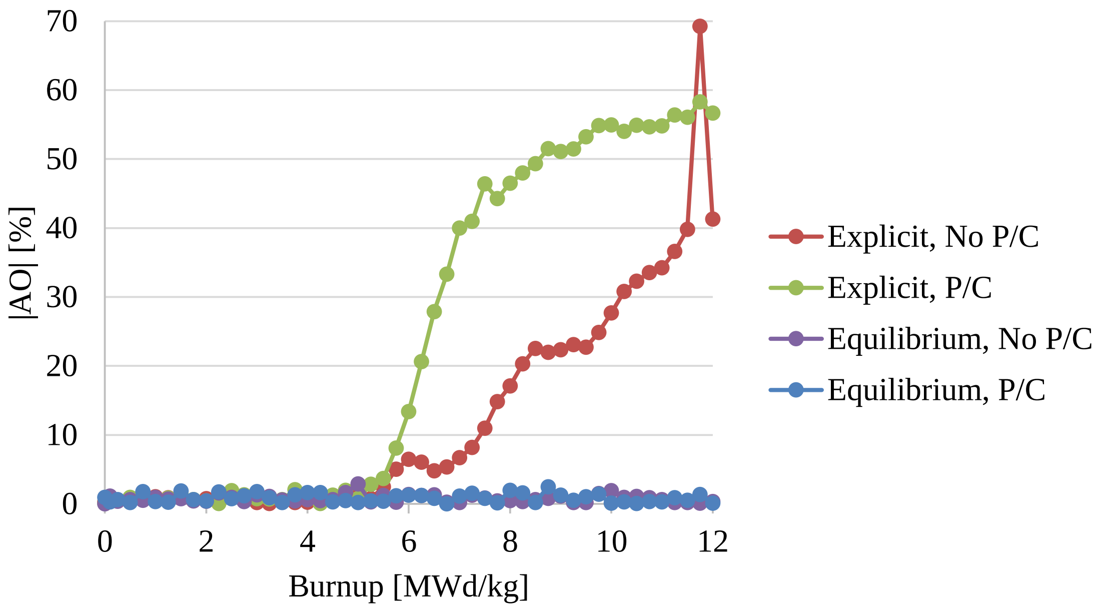


Figure 5.10 Axial offset magnitudes (absolute value) for various depletion simulations.

CHAPTER 6 3D FIRST CYCLE DEPLETION STUDY

6.1 Overview

This chapter discusses a fresh core depletion study of the 3D Serpent model described in Chapter 5. Several Serpent simulations were required to obtain the results presented. An example input file for physics parameters used can be seen in Appendix B.

6.1.1 Objectives

For the depletion of this or any other conventional core, the main objective is to burn the fuel in such a way to extract as much power out of it as possible (i.e. no excess reactivity left in the system). For a PWR system, this is when the CBC becomes very low and diluting the coolant further is infeasible or not practical.

During this depletion period from all fresh fuel to no excess reactivity, the neutronics behavior at specified intervals should be determined. This includes power profiles for both the axial and radial directions, CBC at each time step, and peaking factor data on F_q, F_z, and $F_{\Delta H}$ to evaluate the feasibility of the design.

At each of the specified burnup steps, the power distribution needs to feedback into the material properties (density and temperature for coolant and only temperature for fuel and IFBA) in order to account for thermal hydraulic effects. No iterations were used to converge on the power profile between burnup steps since it was assumed that with a fine enough step size, iterating would be unnecessary[33]. Future studies will iterate on the

[33] As will be addressed later this chapter, this was likely acceptable toward the BOC, but some of the results near EOC could have been greatly improved by iterating on the thermal hydraulics. Although tolerable for this study since it serves as proof of concept for an I²S-LWR core depletion analysis; more accurate results can be found by iterating on the thermal hydraulics between burn steps.

materials compositions and neutronics if the power profile changes dramatically between burnup steps.

6.1.2 Scope and Assumptions

While an actually operating PWR will have a CBC above zero at EOC, this study assumes that EOC occurs after a set period of time rather than achieving a minimal achievable soluble boron concentration. This is done for simplicity since the set burnup amount corresponding to the EOC has a CBC near zero. Further, knowledge was limited pertaining to the Chemical and Volume Control System (CVCS) dilution capabilities of the I²S-LWR, which is assumed to be no less than 50 ppm boric acid. The goal was to select a burnup amount such that the CBC did not go below this assumed dilution limit. This ultimately yielded an EOC core-average burnup value of 15 MWd/kg or 438 EFPD[34], which when combined with the 17 day assumed outage duration, results in 455 days or one year and 90 days (so about 15 months).

Since a significant amount of computational resources needs to go into obtaining the results of each burnup step, seventeen depletion points were used over the fuel cycle. Starting with BOC at 0 MWd/kgHM, these included: 0.15, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15 MWd/kgHM. The step size at the beginning of the cycle is shorter to allow for fission products such as ^{135}Xe to build in as well as allowing for the initially

[34] Although the target for the considered equilibrium I²S-LWR core is a 12-month cycle with 17 days assumed for an outage, this means that the core need only be designed to operate for 348 EFPD. As addressed in previous chapters, since this is a first core with some assemblies only being burned for a single fuel cycle, it may be economically advantageous to run at least the first cycle longer than the equilibrium design length to extract the most value out of these single-use assemblies. About 15 months is viewed as being a good period to both satisfying these considerations yet not diverging too far from the original design.

rapid IFBA depletion, which occurs at a faster rate than the fuel[35]. A larger step size is appropriate after the initial period because the depletion process becomes fairly linear once the major fission products have reached equilibrium and most of the ^{10}B in the IFBA has been burned.

6.2 Cycle Results

The results and figures discussed in this section mostly deal with the power distributions of the fresh core over the cycle. The data used to generate Figure 6.1 and Figure 6.2 can be found in Table 6.1.

Figure 6.1 shows values of three principal hot channel factors at each burnup step. $F_{\Delta H}$ is found based on a quarter assembly basis. F_z is found based on using 96 equally-sized axial regions (1.5 in each over the 144 in active fuel length). F_q is based on using 96 equally-sized (1.5 in) axial regions with quarter assembly radial discretization (11,616 total tally regions). The red data points for each plot corresponds to the results for the HZP case.

The radial ($F_{\Delta H}$) profile remains the lowest over the cycle. It remains close to 1.2 from BOC to EOC (max 1.221, min 1.162). Since the $F_{\Delta H}$ results are roughly constant, F_q performance mostly follows the trends of F_z. As will be addressed in its own section, F_z is initially high due to the classic chopped cosinelike shape of the power distribution. It is lowest during the middle of the cycle because the profile flattens out. F_z climbs again near EOC as the expected shape of a double-humped power distribution develops.

[35] Ideally, the burnable poison (IFBA for this reactor) in a core design should be almost entirely depleted by EOC. Any remaining poison introduces a reactivity penalty and shortens the cycle length; negatively impacting fuel cycle cost..

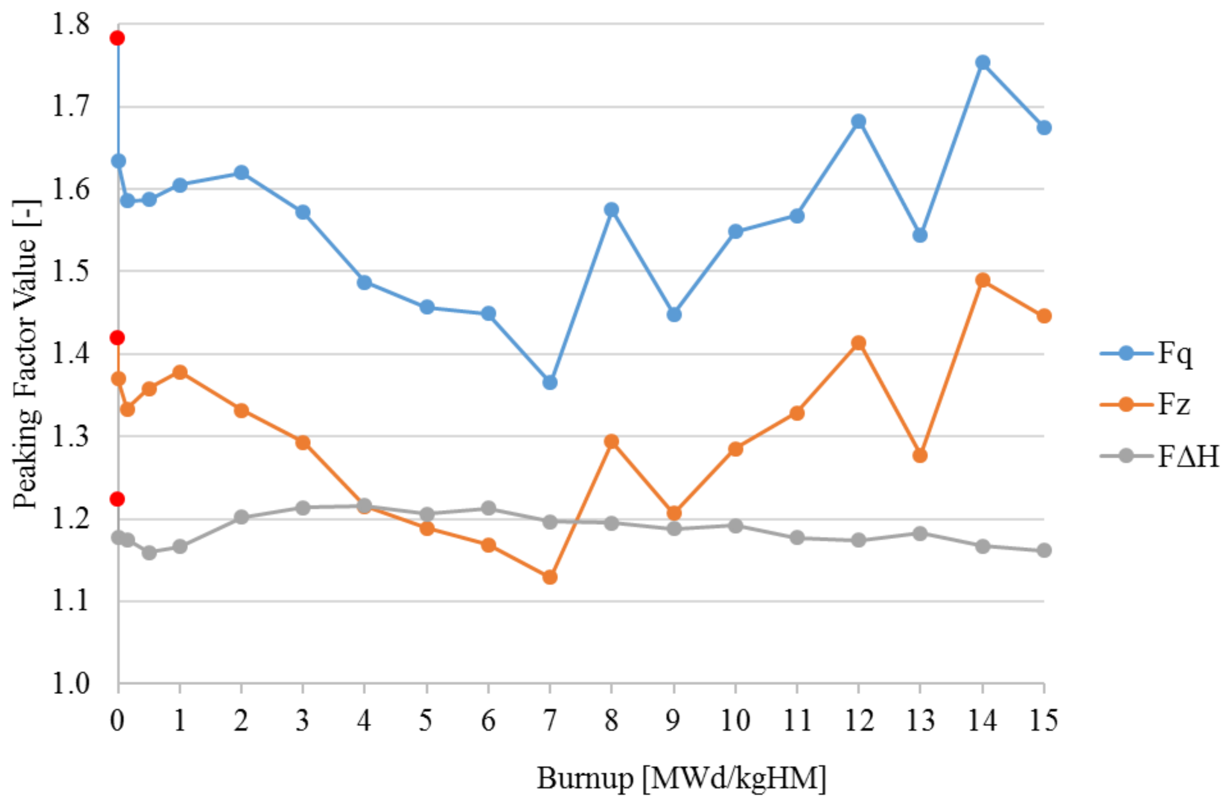


Figure 6.1 Hot channel factors at each burnup step of depletion cycle.

Table 6.1 presents a summary of the values found for key parameters during the course of the depletion cycle. Note that "Boron Used" refers to the concentration of boric acid used during each respective depletion step, found by an estimate from the corrector portion of the previous burnup step. "Actual Boron" refers to the actual soluble concentration which should have been used to achieve an eigenvalue of unity at the start of each burnup step. Observe that for most steps these two values are very close, as one would desire. The goal would have been to have the values be identical, but the level of accuracy achieved from the previous corrector estimate is acceptable for this study.

Table 6.1 Parameters obtained over the course of the depletion.

					Predictor				Corrector				
Step #	Burnup [MWd/kg]	Days	Boron used [ppm]	Actual Boron [ppm]	AO (%)	FΔH	Fz	Fq	AO (%)	FΔH	Fz	Fq	k Error [pcm]
HZP	0	-	1977	1977	12.71	1.224	1.419	1.782	-	-	-	-	-
0	0	0	1764	1764	-3.12	1.178	1.370	1.634	0.79	1.181	1.324	1.591	10
1	0.15	4.38	1695	1713	0.27	1.175	1.333	1.586	-0.15	1.172	1.354	1.581	10
2	0.5	14.6	1699	1695	-1.86	1.159	1.358	1.587	-0.31	1.202	1.365	1.680	12
3	1	29.2	1735	1736	0.15	1.166	1.378	1.605	-0.71	1.221	1.334	1.637	12
4	2	58.4	1739	1741	-0.99	1.202	1.332	1.620	1.28	1.216	1.252	1.524	12
5	3	87.6	1675	1674	-2.76	1.214	1.293	1.572	-0.80	1.205	1.224	1.484	13
6	4	117	1574	1581	-0.68	1.216	1.215	1.487	-0.14	1.210	1.184	1.433	14
7	5	146	1468	1466	-1.32	1.206	1.189	1.457	1.54	1.191	1.139	1.374	13
8	6	175	1339	1345	-1.21	1.213	1.169	1.449	-2.04	1.210	1.176	1.432	18
9	7	204	1220	1210	1.44	1.197	1.129	1.365	-0.93	1.185	1.185	1.441	18
10	8	234	1059	1071	-7.12	1.195	1.294	1.575	-3.22	1.187	1.242	1.490	19
11	9	263	933	931	-1.57	1.188	1.207	1.448	0.10	1.174	1.216	1.432	18
12	10	292	783	788	-4.46	1.192	1.285	1.548	-4.46	1.192	1.285	1.548	18
13	11	321	643	635	-5.60	1.177	1.329	1.568	-3.57	1.168	1.301	1.531	19
14	12	350	491	496	-9.62	1.174	1.414	1.683	-11.13	1.168	1.429	1.685	19
15	13	379.6	345	343	-1.27	1.183	1.277	1.544	2.80	1.162	1.327	1.602	19
16	14	408.8	190	214	-15.82	1.167	1.489	1.753	-11.60	1.162	1.446	1.675	13
17	15	438	63	63	-11.60	1.162	1.446	1.675	-	-	-	-	13

6.2.1 Axial Power Profiles

Figure 6.2 shows the AO at each burnup step. The first point near +13% is the HZP case (colored red). The first few steps at the beginning of the cycle from 0-6 MWd/kgHM are all rather close to zero. While most reactor designs maintain their AO just below the midplane (slightly negative), the slightly positive AO results observed at a few burnup steps originate from having the 7.5 in of cutback used only at the top of the core. If less cutback length was used, the power profile would have been shifted more to the bottom and AO wouldhave likely been negative at each point. This was not chosen for the design used because the power was expected to shift to bottom of the core during the cycle depletion in either case.

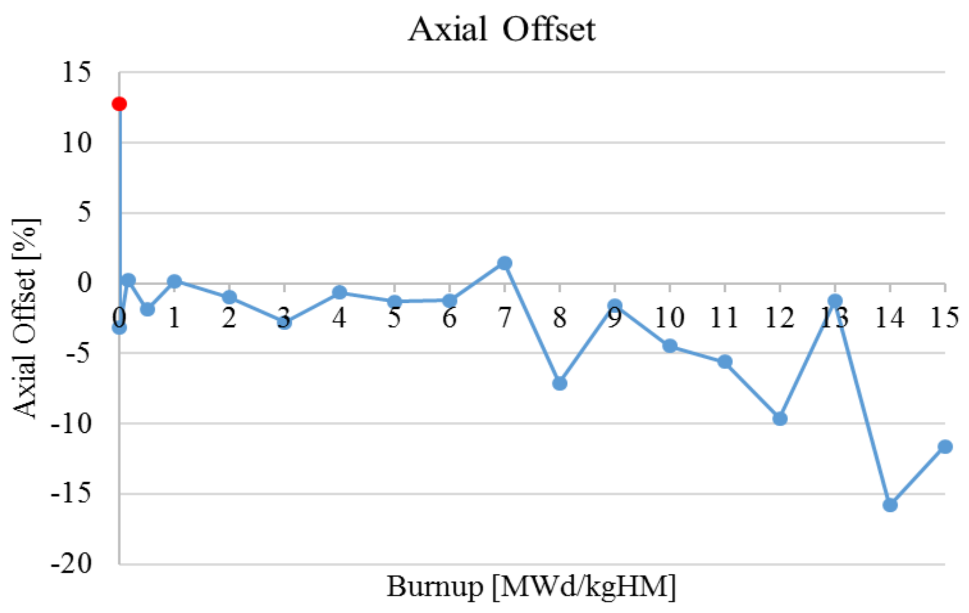


Figure 6.2 Axial offset at each burnup step of the depletion cycle.

Near the end of the cycle, the AO generally becomes more negative due to a power shift resulting from the IFBA depleting in the fuel. Near BOC, the fresh IFBA suppresses the power below the cutback region and the cutback is able to carry an otherwise larger share of the power despite more favorable coolant properties (denser water due to being at a lower temperature) below the cutback region. However, as the IFBA depletes, there is less integral reactivity control below the cutback region, reducing the difference between the IFBA-coated fuel and the cutback fuel. Thus, the bottom of the core is able to garner a larger contribution of the power. This process devolves to the power shifting down since the coolant heats and becomes less dense further down, so that by the time the coolant reaches the top, it is less favorable for producing power, shifting the power down even further. The individual axial power profile at each burnup step can be seen in

Figure 6.3.

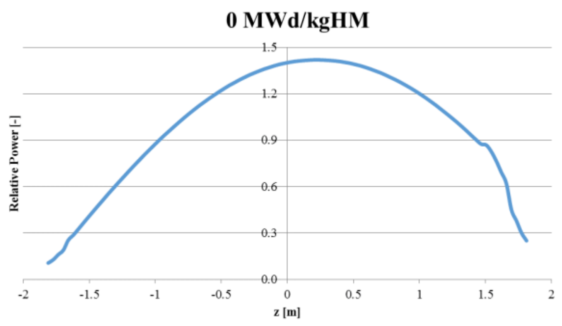
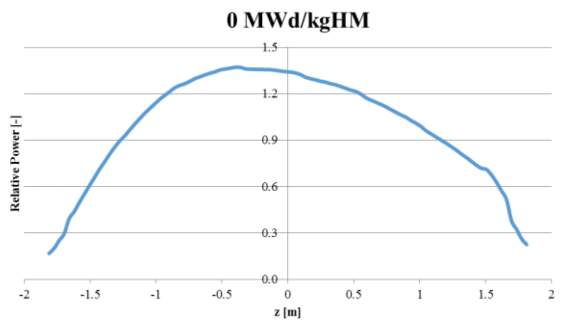
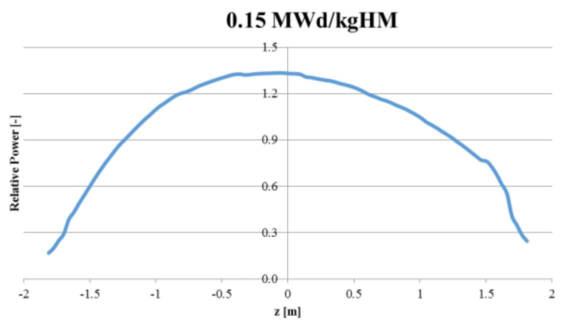
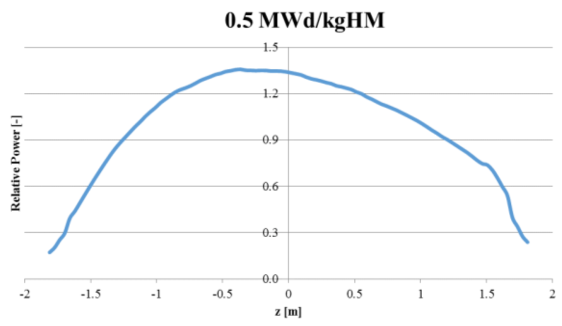
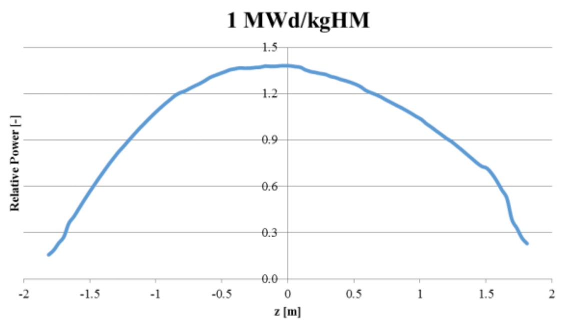
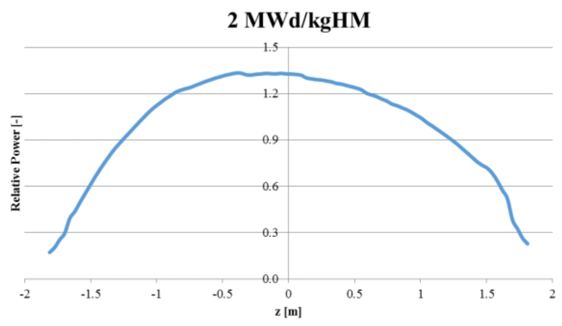
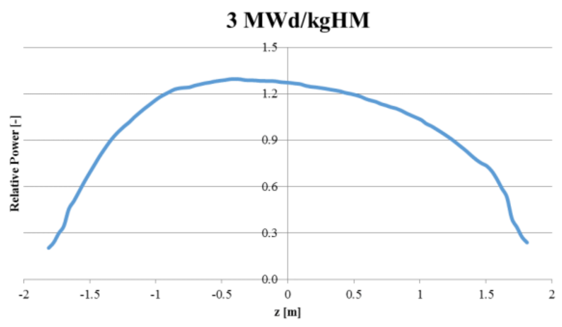
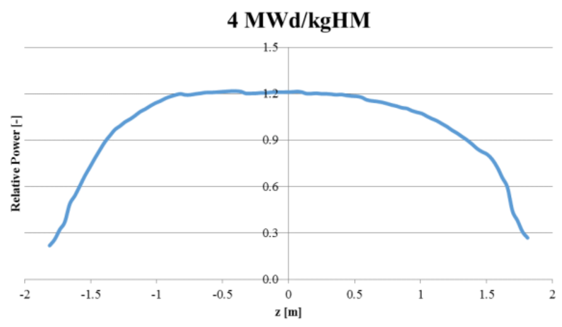
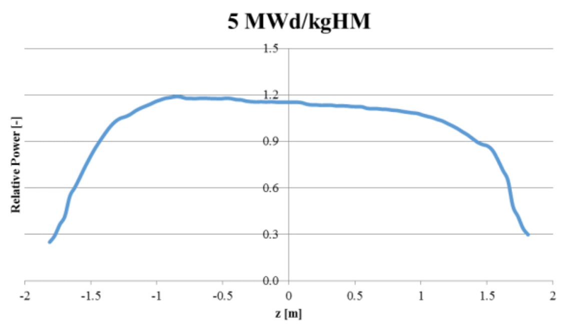
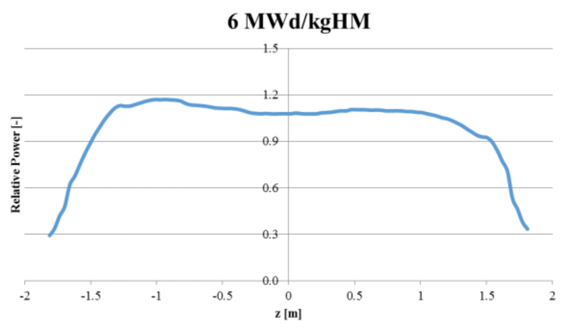
 HZP	 BOC
 0.15 MWd/kgHM	 0.5 MWd/kgHM
 1 MWd/kgHM	 2 MWd/kgHM
 3 MWd/kgHM	 4 MWd/kgHM

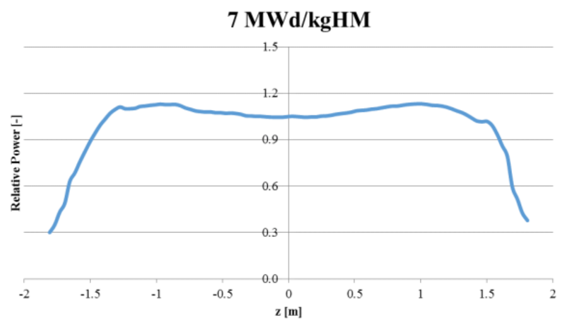
Figure 6.3 Axial power profiles over burnup cycle.



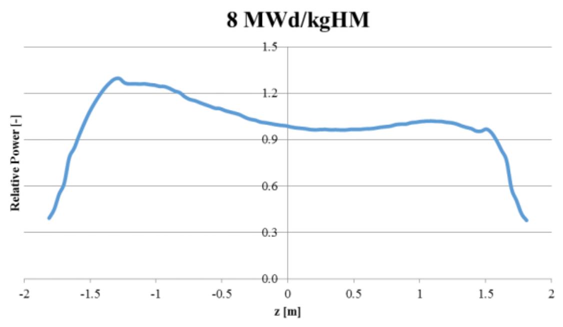
5 MWd/kgHM



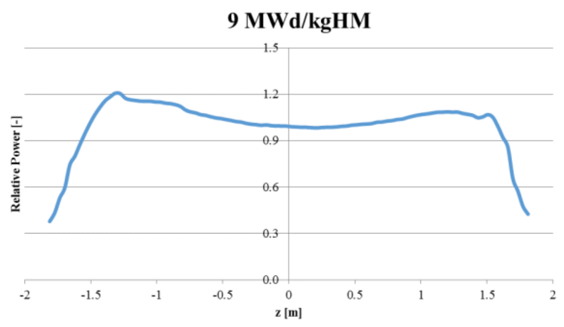
6 MWd/kgHM



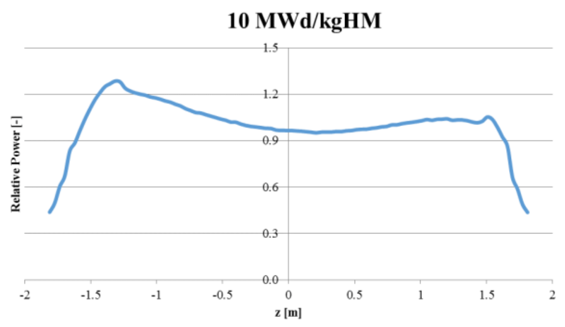
7 MWd/kgHM



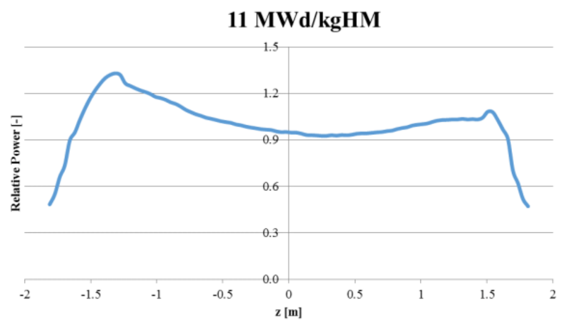
8 MWd/kgHM



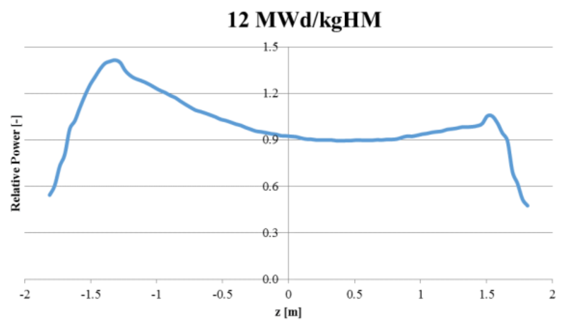
9 MWd/kgHM



10 MWd/kgHM

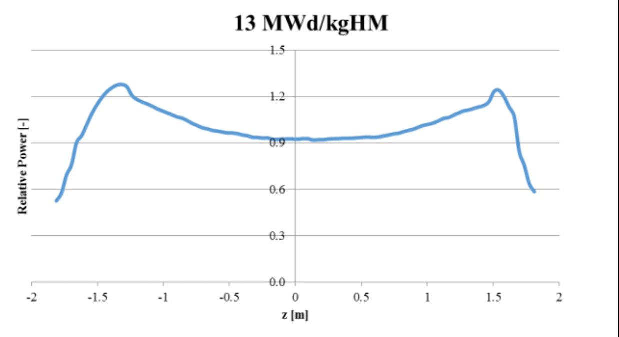


11 MWd/kgHM

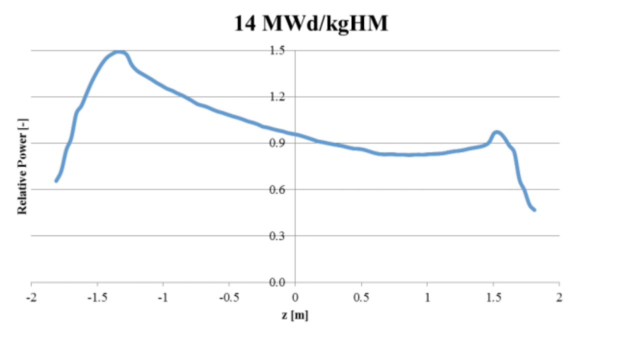


12 MWd/kgHM

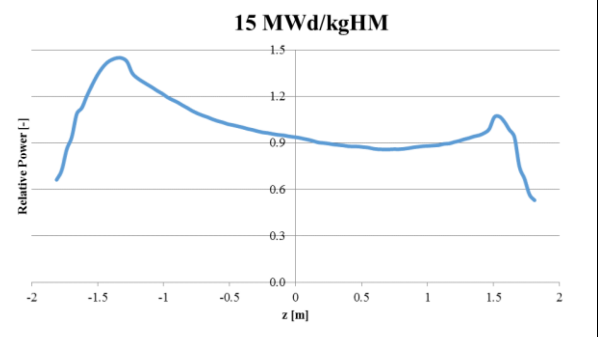
Figure 6.3 continued.



13 MWd/kgHM



14 MWd/kgHM



15 MWd/kgHM (EOC)

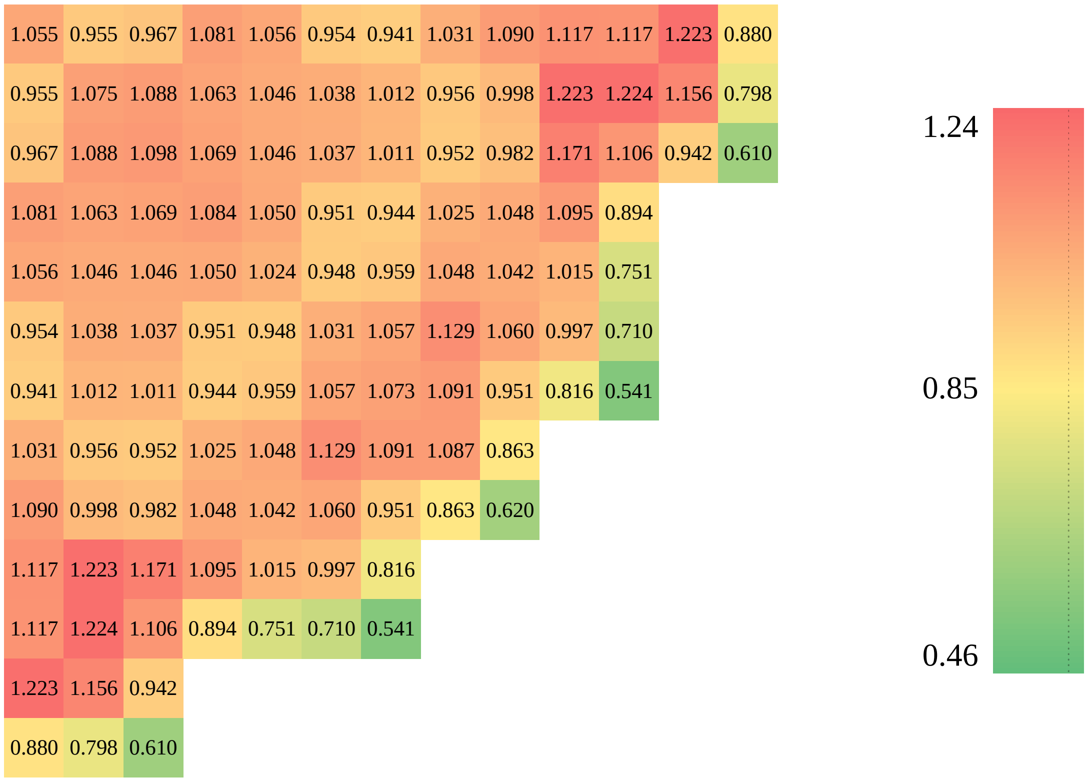
Figure 6.3 continued.

The slight peak near the right end (top of the core), visible in most of the axial profiles, is where the cutback region is located. One can observe that the cutback region by itself is barely noticeable near BOC. This is misleading since the cutback is solely responsible for the AO being close to zero and not being large and negative. As mentioned above, when the power profile transitions away from a cosine then flat shape, a double hump profile emerges toward the bottom of the core.

Although easier to be observed when the figures are larger, one can notice that there are locations in the plots where the power seems to suddenly drop just slightly. This behavior is entirely expected and the locations correspond to the axial segment boundaries. Below the locations (to the left in the figures), coolant is colder and denser (beneficial for reactivity). Above the locations (to the right in the figures), coolant is hotter and less dense (detrimental for reactivity). Although small, these perturbations illustrate how some accuracy of the results is lost due making assumptions and simplifications in the model. These effects would be less noticeable if more segments were used (finer axial discretization).

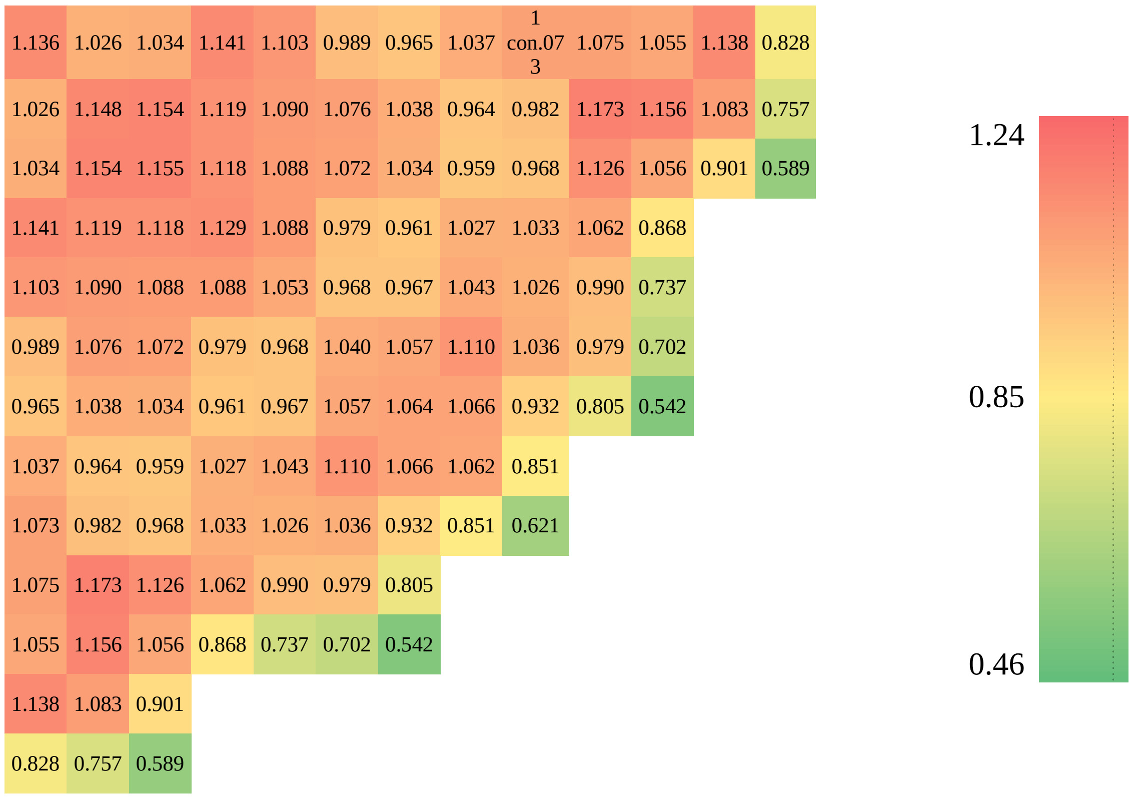
6.2.2 Radial Power Profiles

Figure 6.4 through Figure 6.22 presented below are the radial power profiles at each burnup step. At BOC, power is closer to the periphery of the core. As the core burns, this shifts more toward the center. It should be noted here that the Serpent outputs produced quarter core results, and after that, the data was eighth-core averaged (for symmetric plots) and redistributed to obtain the quarter core results below.



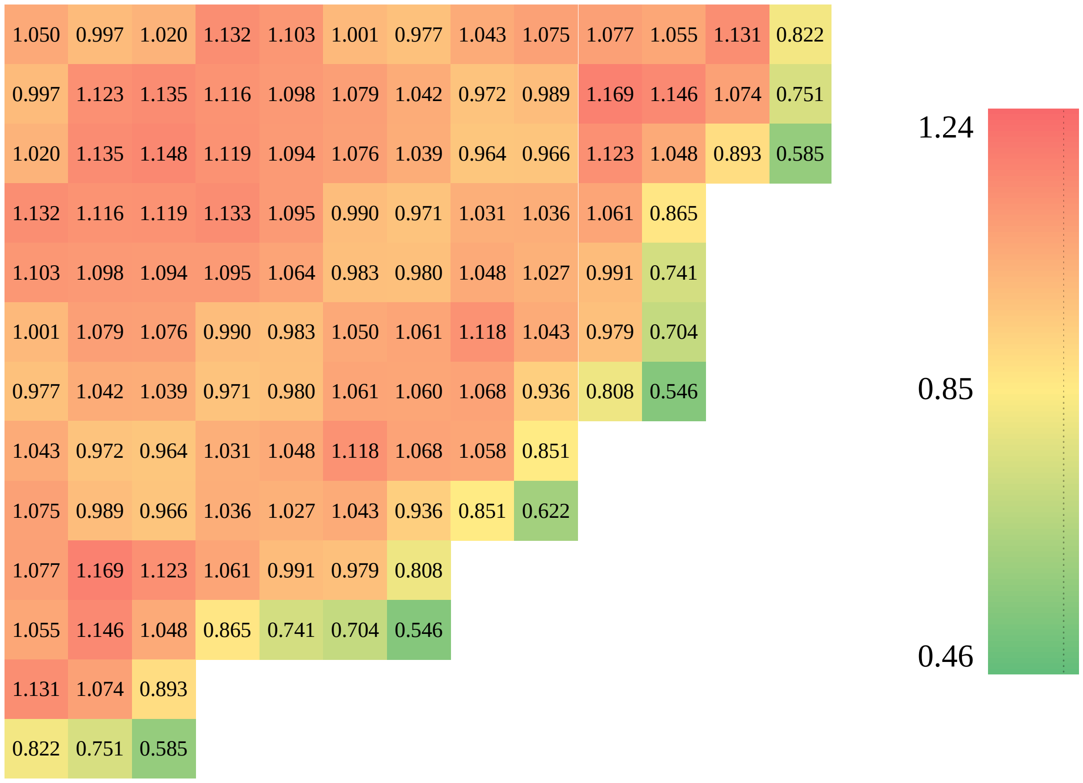
1.055	0.955	0.967	1.081	1.056	0.954	0.941	1.031	1.090	1.117	1.117	1.223	0.880
0.955	1.075	1.088	1.063	1.046	1.038	1.012	0.956	0.998	1.223	1.224	1.156	0.798
0.967	1.088	1.098	1.069	1.046	1.037	1.011	0.952	0.982	1.171	1.106	0.942	0.610
1.081	1.063	1.069	1.084	1.050	0.951	0.944	1.025	1.048	1.095	0.894		
1.056	1.046	1.046	1.050	1.024	0.948	0.959	1.048	1.042	1.015	0.751		
0.954	1.038	1.037	0.951	0.948	1.031	1.057	1.129	1.060	0.997	0.710		
0.941	1.012	1.011	0.944	0.959	1.057	1.073	1.091	0.951	0.816	0.541		
1.031	0.956	0.952	1.025	1.048	1.129	1.091	1.087	0.863				
1.090	0.998	0.982	1.048	1.042	1.060	0.951	0.863	0.620				
1.117	1.223	1.171	1.095	1.015	0.997	0.816						
1.117	1.224	1.106	0.894	0.751	0.710	0.541						
1.223	1.156	0.942										
0.880	0.798	0.610										

Figure 6.4 Radial power profile at HZP.



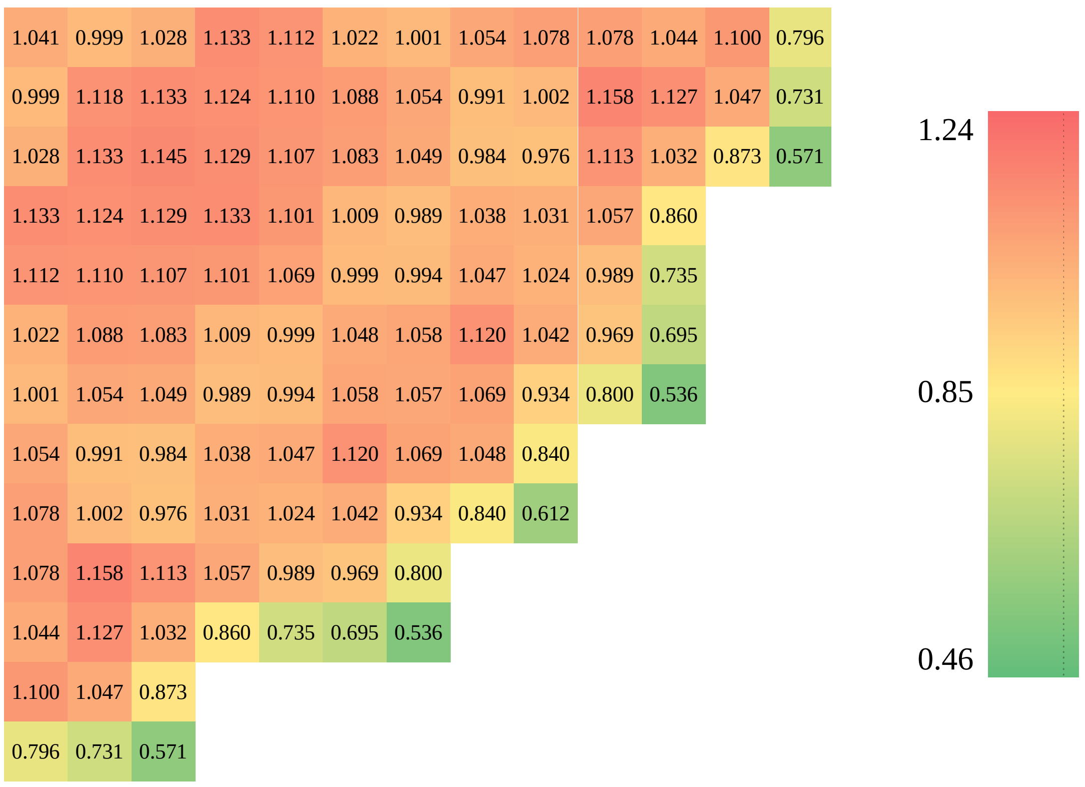
1.136	1.026	1.034	1.141	1.103	0.989	0.965	1.037	1 con.07 3	1.075	1.055	1.138	0.828
1.026	1.148	1.154	1.119	1.090	1.076	1.038	0.964	0.982	1.173	1.156	1.083	0.757
1.034	1.154	1.155	1.118	1.088	1.072	1.034	0.959	0.968	1.126	1.056	0.901	0.589
1.141	1.119	1.118	1.129	1.088	0.979	0.961	1.027	1.033	1.062	0.868		
1.103	1.090	1.088	1.088	1.053	0.968	0.967	1.043	1.026	0.990	0.737		
0.989	1.076	1.072	0.979	0.968	1.040	1.057	1.110	1.036	0.979	0.702		
0.965	1.038	1.034	0.961	0.967	1.057	1.064	1.066	0.932	0.805	0.542		
1.037	0.964	0.959	1.027	1.043	1.110	1.066	1.062	0.851				
1.073	0.982	0.968	1.033	1.026	1.036	0.932	0.851	0.621				
1.075	1.173	1.126	1.062	0.990	0.979	0.805						
1.055	1.156	1.056	0.868	0.737	0.702	0.542						
1.138	1.083	0.901										
0.828	0.757	0.589										

Figure 6.5 Radial power profile at BOC.



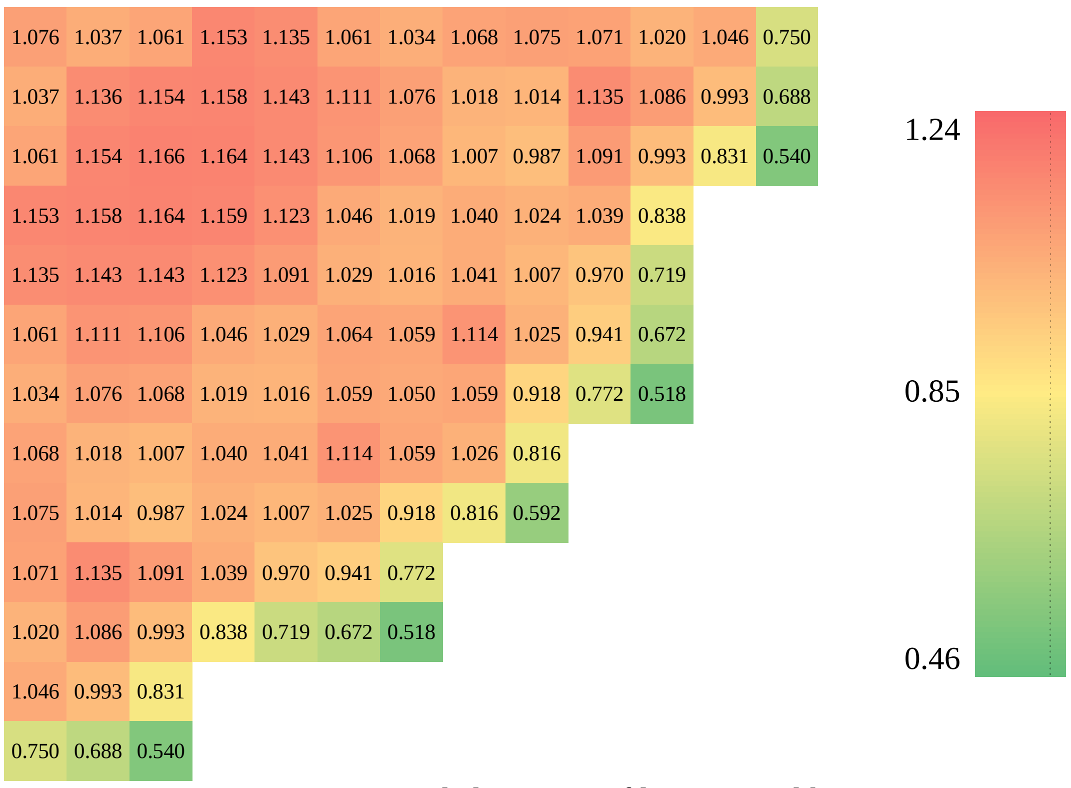
1.050	0.997	1.020	1.132	1.103	1.001	0.977	1.043	1.075	1.077	1.055	1.131	0.822
0.997	1.123	1.135	1.116	1.098	1.079	1.042	0.972	0.989	1.169	1.146	1.074	0.751
1.020	1.135	1.148	1.119	1.094	1.076	1.039	0.964	0.966	1.123	1.048	0.893	0.585
1.132	1.116	1.119	1.133	1.095	0.990	0.971	1.031	1.036	1.061	0.865		
1.103	1.098	1.094	1.095	1.064	0.983	0.980	1.048	1.027	0.991	0.741		
1.001	1.079	1.076	0.990	0.983	1.050	1.061	1.118	1.043	0.979	0.704		
0.977	1.042	1.039	0.971	0.980	1.061	1.060	1.068	0.936	0.808	0.546		
1.043	0.972	0.964	1.031	1.048	1.118	1.068	1.058	0.851				
1.075	0.989	0.966	1.036	1.027	1.043	0.936	0.851	0.622				
1.077	1.169	1.123	1.061	0.991	0.979	0.808						
1.055	1.146	1.048	0.865	0.741	0.704	0.546						
1.131	1.074	0.893										
0.822	0.751	0.585										

Figure 6.6 Radial power profile at 0.15 MWd/kgHM.



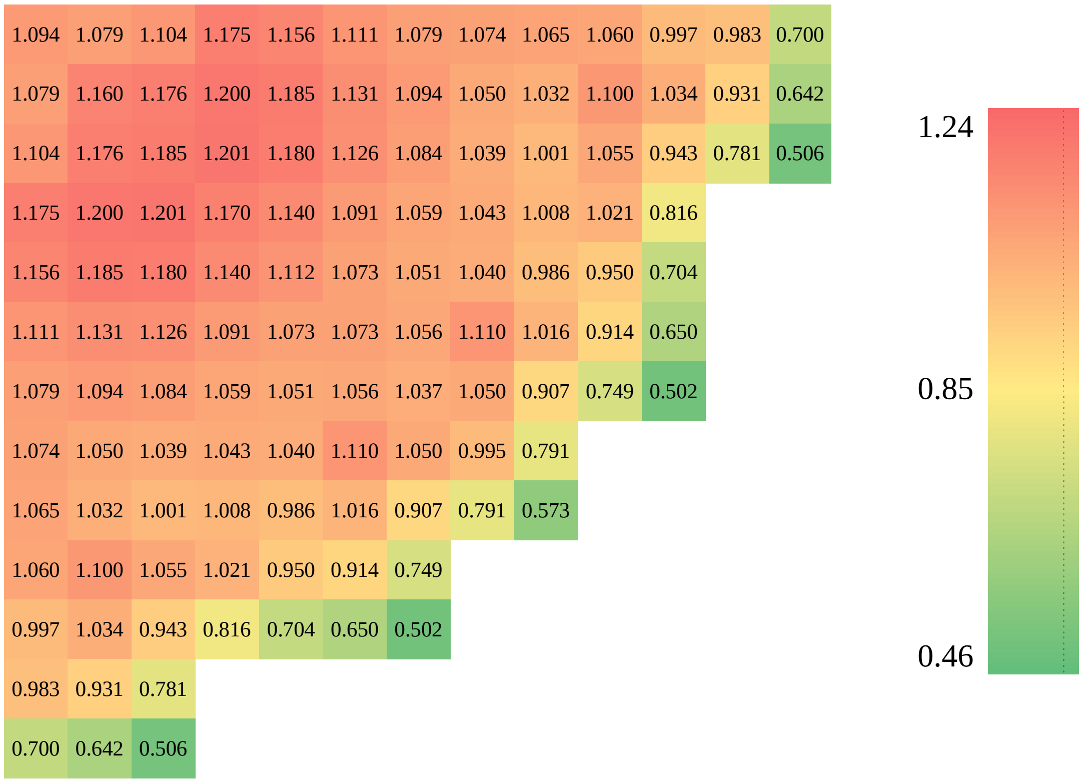
1.041	0.999	1.028	1.133	1.112	1.022	1.001	1.054	1.078	1.078	1.044	1.100	0.796
0.999	1.118	1.133	1.124	1.110	1.088	1.054	0.991	1.002	1.158	1.127	1.047	0.731
1.028	1.133	1.145	1.129	1.107	1.083	1.049	0.984	0.976	1.113	1.032	0.873	0.571
1.133	1.124	1.129	1.133	1.101	1.009	0.989	1.038	1.031	1.057	0.860		
1.112	1.110	1.107	1.101	1.069	0.999	0.994	1.047	1.024	0.989	0.735		
1.022	1.088	1.083	1.009	0.999	1.048	1.058	1.120	1.042	0.969	0.695		
1.001	1.054	1.049	0.989	0.994	1.058	1.057	1.069	0.934	0.800	0.536		
1.054	0.991	0.984	1.038	1.047	1.120	1.069	1.048	0.840				
1.078	1.002	0.976	1.031	1.024	1.042	0.934	0.840	0.612				
1.078	1.158	1.113	1.057	0.989	0.969	0.800						
1.044	1.127	1.032	0.860	0.735	0.695	0.536						
1.100	1.047	0.873										
0.796	0.731	0.571										

Figure 6.7 Radial power profile at 0.5 MWd/kgHM.



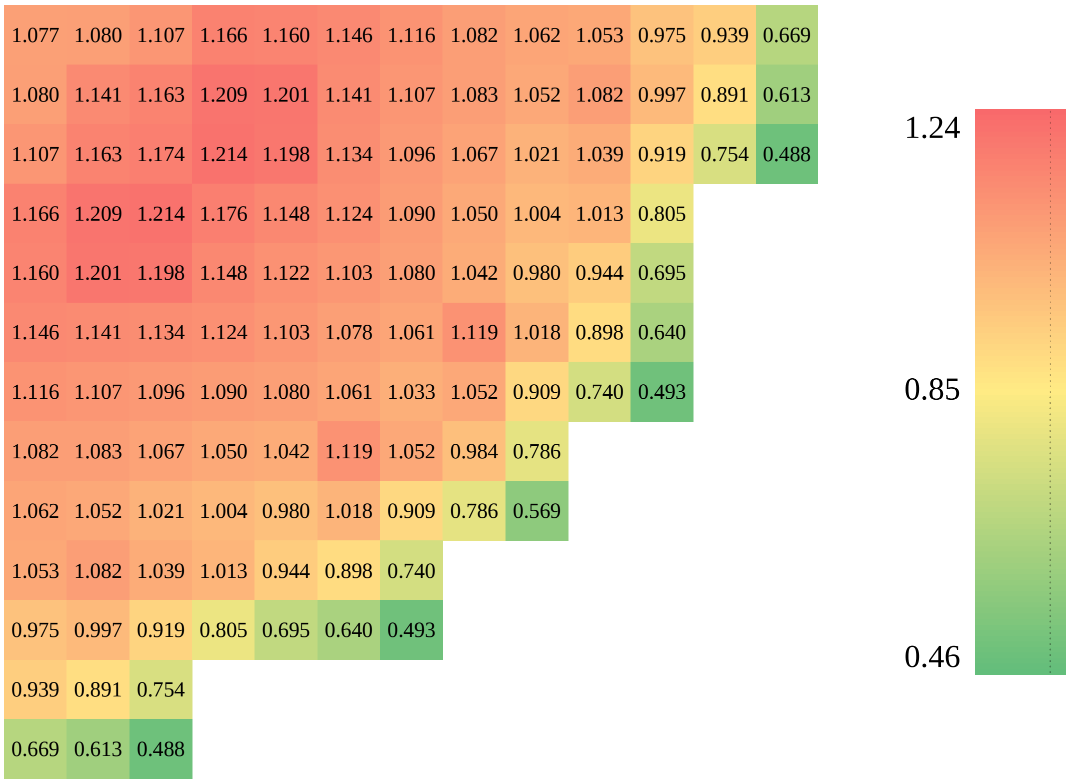
1.076	1.037	1.061	1.153	1.135	1.061	1.034	1.068	1.075	1.071	1.020	1.046	0.750
1.037	1.136	1.154	1.158	1.143	1.111	1.076	1.018	1.014	1.135	1.086	0.993	0.688
1.061	1.154	1.166	1.164	1.143	1.106	1.068	1.007	0.987	1.091	0.993	0.831	0.540
1.153	1.158	1.164	1.159	1.123	1.046	1.019	1.040	1.024	1.039	0.838		
1.135	1.143	1.143	1.123	1.091	1.029	1.016	1.041	1.007	0.970	0.719		
1.061	1.111	1.106	1.046	1.029	1.064	1.059	1.114	1.025	0.941	0.672		
1.034	1.076	1.068	1.019	1.016	1.059	1.050	1.059	0.918	0.772	0.518		
1.068	1.018	1.007	1.040	1.041	1.114	1.059	1.026	0.816				
1.075	1.014	0.987	1.024	1.007	1.025	0.918	0.816	0.592				
1.071	1.135	1.091	1.039	0.970	0.941	0.772						
1.020	1.086	0.993	0.838	0.719	0.672	0.518						
1.046	0.993	0.831										
0.750	0.688	0.540										

Figure 6.8 Radial power profile at 1 MWd/kgHM.



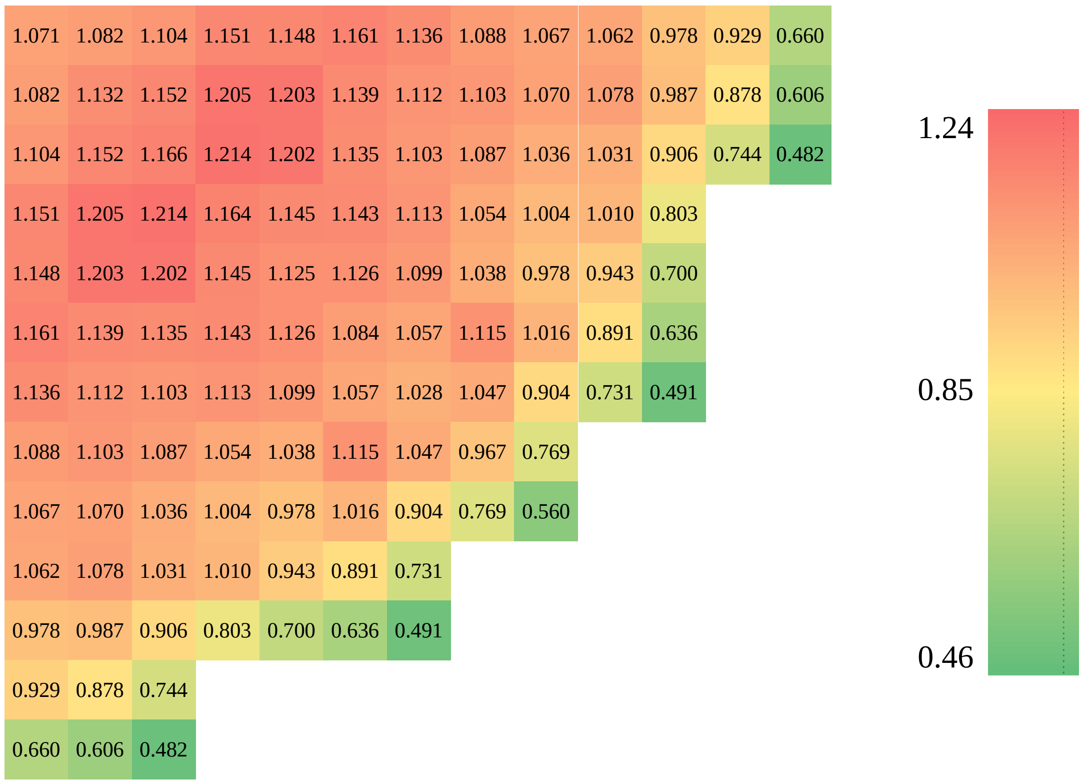
1.094	1.079	1.104	1.175	1.156	1.111	1.079	1.074	1.065	1.060	0.997	0.983	0.700
1.079	1.160	1.176	1.200	1.185	1.131	1.094	1.050	1.032	1.100	1.034	0.931	0.642
1.104	1.176	1.185	1.201	1.180	1.126	1.084	1.039	1.001	1.055	0.943	0.781	0.506
1.175	1.200	1.201	1.170	1.140	1.091	1.059	1.043	1.008	1.021	0.816		
1.156	1.185	1.180	1.140	1.112	1.073	1.051	1.040	0.986	0.950	0.704		
1.111	1.131	1.126	1.091	1.073	1.073	1.056	1.110	1.016	0.914	0.650		
1.079	1.094	1.084	1.059	1.051	1.056	1.037	1.050	0.907	0.749	0.502		
1.074	1.050	1.039	1.043	1.040	1.110	1.050	0.995	0.791				
1.065	1.032	1.001	1.008	0.986	1.016	0.907	0.791	0.573				
1.060	1.100	1.055	1.021	0.950	0.914	0.749						
0.997	1.034	0.943	0.816	0.704	0.650	0.502						
0.983	0.931	0.781										
0.700	0.642	0.506										

Figure 6.9 Radial power profile at 2 MWd/kgHM.



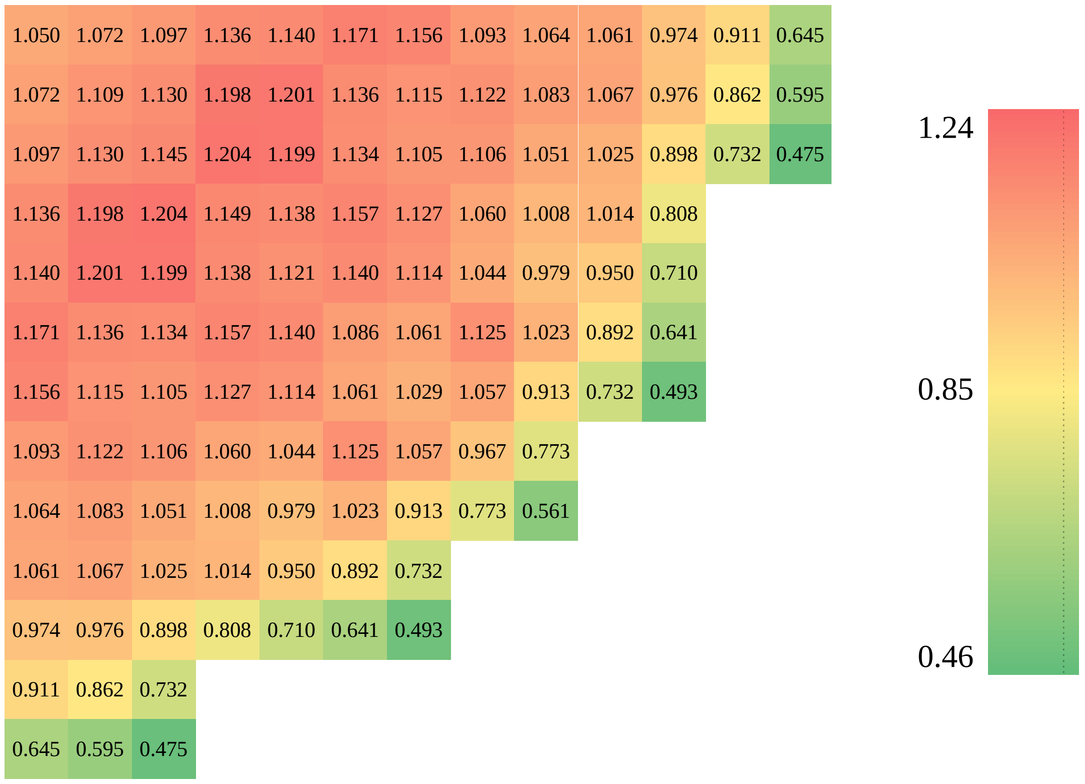
1.077	1.080	1.107	1.166	1.160	1.146	1.116	1.082	1.062	1.053	0.975	0.939	0.669
1.080	1.141	1.163	1.209	1.201	1.141	1.107	1.083	1.052	1.082	0.997	0.891	0.613
1.107	1.163	1.174	1.214	1.198	1.134	1.096	1.067	1.021	1.039	0.919	0.754	0.488
1.166	1.209	1.214	1.176	1.148	1.124	1.090	1.050	1.004	1.013	0.805		
1.160	1.201	1.198	1.148	1.122	1.103	1.080	1.042	0.980	0.944	0.695		
1.146	1.141	1.134	1.124	1.103	1.078	1.061	1.119	1.018	0.898	0.640		
1.116	1.107	1.096	1.090	1.080	1.061	1.033	1.052	0.909	0.740	0.493		
1.082	1.083	1.067	1.050	1.042	1.119	1.052	0.984	0.786				
1.062	1.052	1.021	1.004	0.980	1.018	0.909	0.786	0.569				
1.053	1.082	1.039	1.013	0.944	0.898	0.740						
0.975	0.997	0.919	0.805	0.695	0.640	0.493						
0.939	0.891	0.754										
0.669	0.613	0.488										

Figure 6.10 Radial power profile at 3 MWd/kgHM.



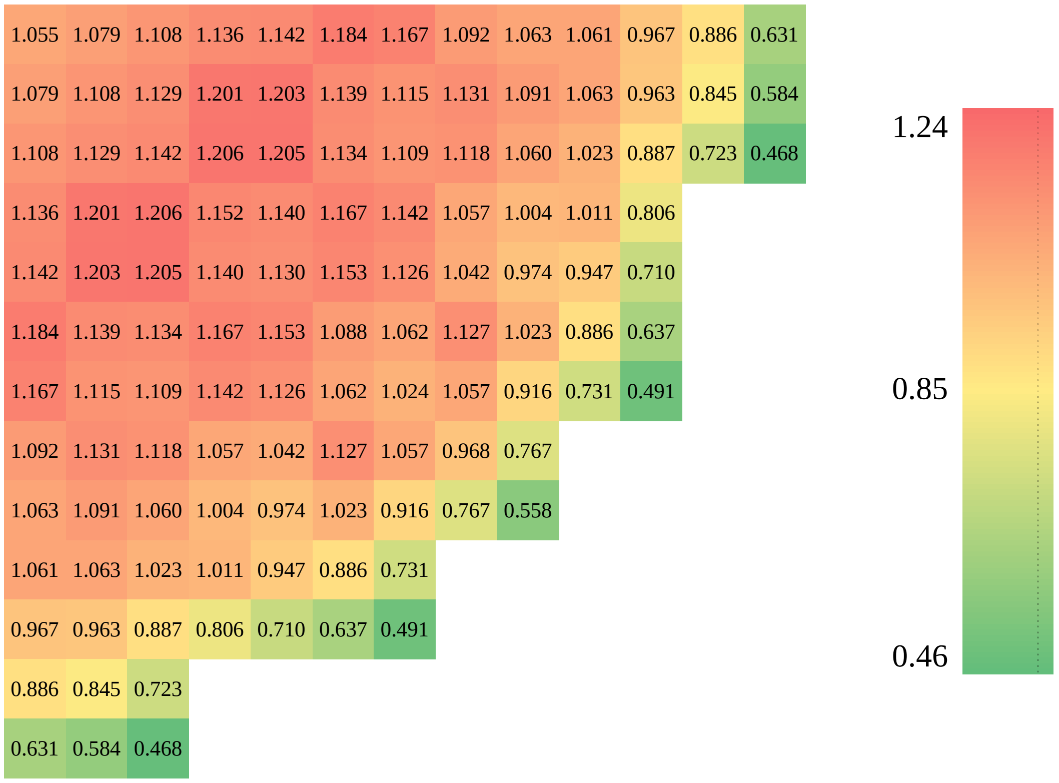
1.071	1.082	1.104	1.151	1.148	1.161	1.136	1.088	1.067	1.062	0.978	0.929	0.660
1.082	1.132	1.152	1.205	1.203	1.139	1.112	1.103	1.070	1.078	0.987	0.878	0.606
1.104	1.152	1.166	1.214	1.202	1.135	1.103	1.087	1.036	1.031	0.906	0.744	0.482
1.151	1.205	1.214	1.164	1.145	1.143	1.113	1.054	1.004	1.010	0.803		
1.148	1.203	1.202	1.145	1.125	1.126	1.099	1.038	0.978	0.943	0.700		
1.161	1.139	1.135	1.143	1.126	1.084	1.057	1.115	1.016	0.891	0.636		
1.136	1.112	1.103	1.113	1.099	1.057	1.028	1.047	0.904	0.731	0.491		
1.088	1.103	1.087	1.054	1.038	1.115	1.047	0.967	0.769				
1.067	1.070	1.036	1.004	0.978	1.016	0.904	0.769	0.560				
1.062	1.078	1.031	1.010	0.943	0.891	0.731						
0.978	0.987	0.906	0.803	0.700	0.636	0.491						
0.929	0.878	0.744										
0.660	0.606	0.482										

Figure 6.11 Radial power profile at 4 MWd/kgHM.



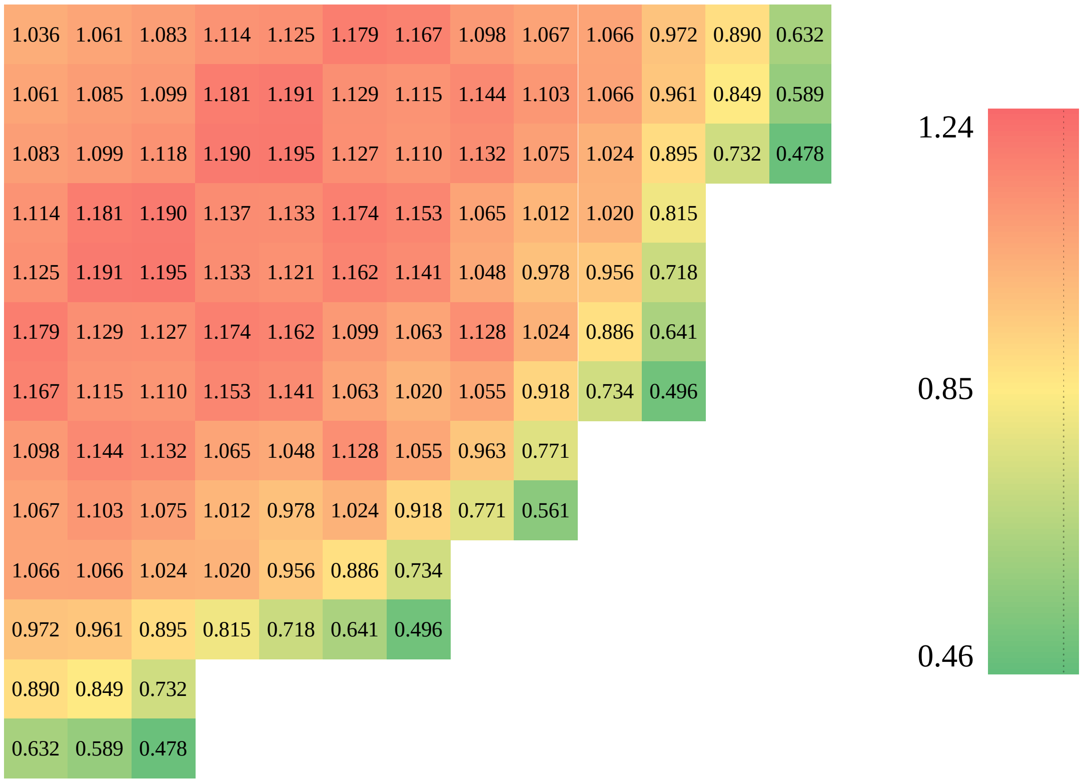
1.050	1.072	1.097	1.136	1.140	1.171	1.156	1.093	1.064	1.061	0.974	0.911	0.645
1.072	1.109	1.130	1.198	1.201	1.136	1.115	1.122	1.083	1.067	0.976	0.862	0.595
1.097	1.130	1.145	1.204	1.199	1.134	1.105	1.106	1.051	1.025	0.898	0.732	0.475
1.136	1.198	1.204	1.149	1.138	1.157	1.127	1.060	1.008	1.014	0.808		
1.140	1.201	1.199	1.138	1.121	1.140	1.114	1.044	0.979	0.950	0.710		
1.171	1.136	1.134	1.157	1.140	1.086	1.061	1.125	1.023	0.892	0.641		
1.156	1.115	1.105	1.127	1.114	1.061	1.029	1.057	0.913	0.732	0.493		
1.093	1.122	1.106	1.060	1.044	1.125	1.057	0.967	0.773				
1.064	1.083	1.051	1.008	0.979	1.023	0.913	0.773	0.561				
1.061	1.067	1.025	1.014	0.950	0.892	0.732						
0.974	0.976	0.898	0.808	0.710	0.641	0.493						
0.911	0.862	0.732										
0.645	0.595	0.475										

Figure 6.12 Radial power profile at 5 MWd/kgHM.



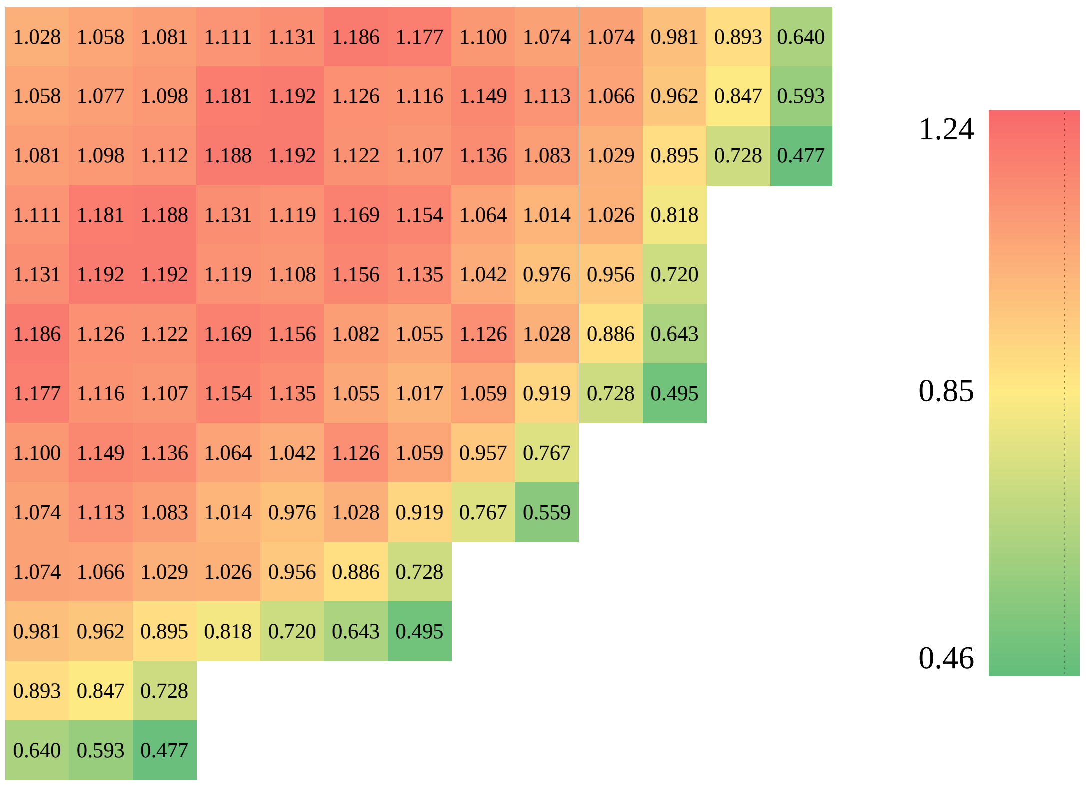
1.055	1.079	1.108	1.136	1.142	1.184	1.167	1.092	1.063	1.061	0.967	0.886	0.631
1.079	1.108	1.129	1.201	1.203	1.139	1.115	1.131	1.091	1.063	0.963	0.845	0.584
1.108	1.129	1.142	1.206	1.205	1.134	1.109	1.118	1.060	1.023	0.887	0.723	0.468
1.136	1.201	1.206	1.152	1.140	1.167	1.142	1.057	1.004	1.011	0.806		
1.142	1.203	1.205	1.140	1.130	1.153	1.126	1.042	0.974	0.947	0.710		
1.184	1.139	1.134	1.167	1.153	1.088	1.062	1.127	1.023	0.886	0.637		
1.167	1.115	1.109	1.142	1.126	1.062	1.024	1.057	0.916	0.731	0.491		
1.092	1.131	1.118	1.057	1.042	1.127	1.057	0.968	0.767				
1.063	1.091	1.060	1.004	0.974	1.023	0.916	0.767	0.558				
1.061	1.063	1.023	1.011	0.947	0.886	0.731						
0.967	0.963	0.887	0.806	0.710	0.637	0.491						
0.886	0.845	0.723										
0.631	0.584	0.468										

Figure 6.13 Radial power profile at 6 MWd/kgHM.



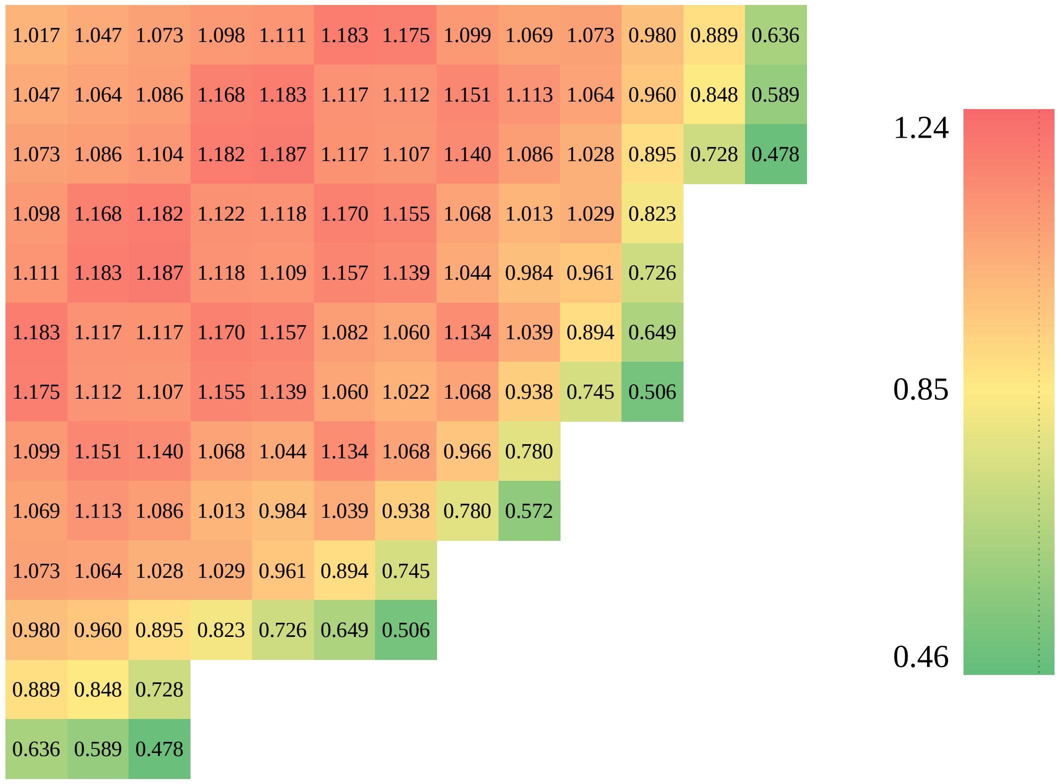
1.036	1.061	1.083	1.114	1.125	1.179	1.167	1.098	1.067	1.066	0.972	0.890	0.632
1.061	1.085	1.099	1.181	1.191	1.129	1.115	1.144	1.103	1.066	0.961	0.849	0.589
1.083	1.099	1.118	1.190	1.195	1.127	1.110	1.132	1.075	1.024	0.895	0.732	0.478
1.114	1.181	1.190	1.137	1.133	1.174	1.153	1.065	1.012	1.020	0.815		
1.125	1.191	1.195	1.133	1.121	1.162	1.141	1.048	0.978	0.956	0.718		
1.179	1.129	1.127	1.174	1.162	1.099	1.063	1.128	1.024	0.886	0.641		
1.167	1.115	1.110	1.153	1.141	1.063	1.020	1.055	0.918	0.734	0.496		
1.098	1.144	1.132	1.065	1.048	1.128	1.055	0.963	0.771				
1.067	1.103	1.075	1.012	0.978	1.024	0.918	0.771	0.561				
1.066	1.066	1.024	1.020	0.956	0.886	0.734						
0.972	0.961	0.895	0.815	0.718	0.641	0.496						
0.890	0.849	0.732										
0.632	0.589	0.478										

Figure 6.14 Radial power profile at 7 MWd/kgHM.



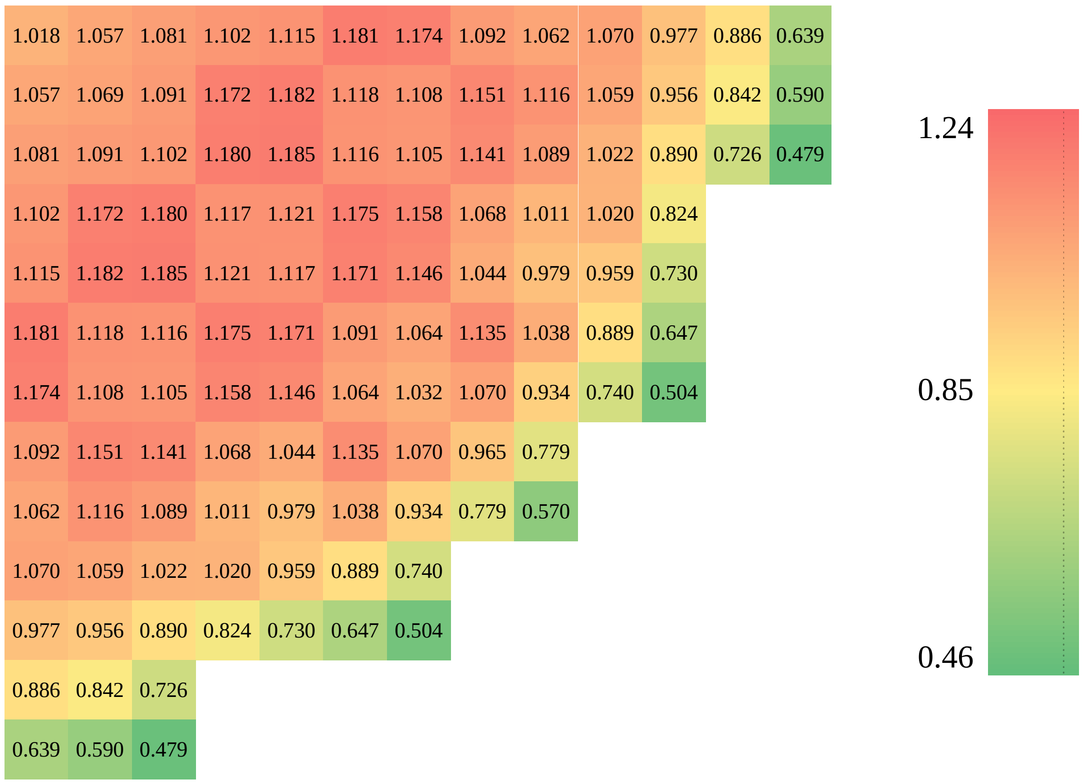
1.028	1.058	1.081	1.111	1.131	1.186	1.177	1.100	1.074	1.074	0.981	0.893	0.640
1.058	1.077	1.098	1.181	1.192	1.126	1.116	1.149	1.113	1.066	0.962	0.847	0.593
1.081	1.098	1.112	1.188	1.192	1.122	1.107	1.136	1.083	1.029	0.895	0.728	0.477
1.111	1.181	1.188	1.131	1.119	1.169	1.154	1.064	1.014	1.026	0.818		
1.131	1.192	1.192	1.119	1.108	1.156	1.135	1.042	0.976	0.956	0.720		
1.186	1.126	1.122	1.169	1.156	1.082	1.055	1.126	1.028	0.886	0.643		
1.177	1.116	1.107	1.154	1.135	1.055	1.017	1.059	0.919	0.728	0.495		
1.100	1.149	1.136	1.064	1.042	1.126	1.059	0.957	0.767				
1.074	1.113	1.083	1.014	0.976	1.028	0.919	0.767	0.559				
1.074	1.066	1.029	1.026	0.956	0.886	0.728						
0.981	0.962	0.895	0.818	0.720	0.643	0.495						
0.893	0.847	0.728										
0.640	0.593	0.477										

Figure 6.15 Radial power profile at 8 MWd/kgHM.



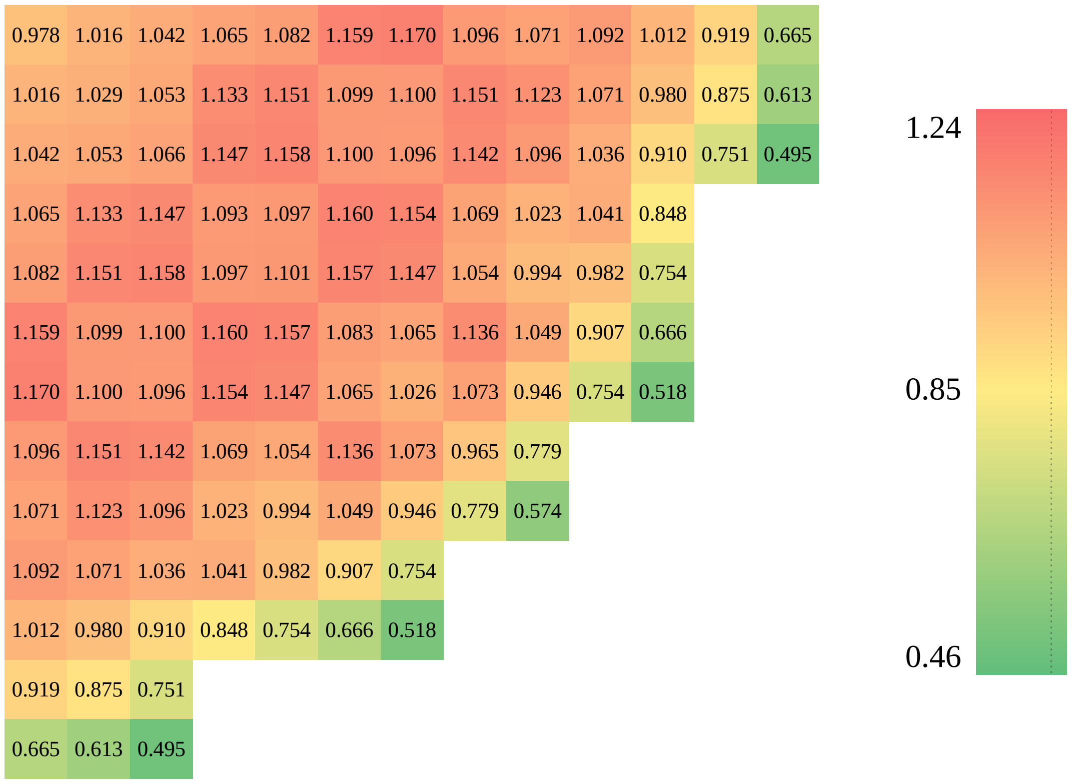
1.017	1.047	1.073	1.098	1.111	1.183	1.175	1.099	1.069	1.073	0.980	0.889	0.636
1.047	1.064	1.086	1.168	1.183	1.117	1.112	1.151	1.113	1.064	0.960	0.848	0.589
1.073	1.086	1.104	1.182	1.187	1.117	1.107	1.140	1.086	1.028	0.895	0.728	0.478
1.098	1.168	1.182	1.122	1.118	1.170	1.155	1.068	1.013	1.029	0.823		
1.111	1.183	1.187	1.118	1.109	1.157	1.139	1.044	0.984	0.961	0.726		
1.183	1.117	1.117	1.170	1.157	1.082	1.060	1.134	1.039	0.894	0.649		
1.175	1.112	1.107	1.155	1.139	1.060	1.022	1.068	0.938	0.745	0.506		
1.099	1.151	1.140	1.068	1.044	1.134	1.068	0.966	0.780				
1.069	1.113	1.086	1.013	0.984	1.039	0.938	0.780	0.572				
1.073	1.064	1.028	1.029	0.961	0.894	0.745						
0.980	0.960	0.895	0.823	0.726	0.649	0.506						
0.889	0.848	0.728										
0.636	0.589	0.478										

Figure 6.16 Radial power profile at 9 MWd/kgHM.



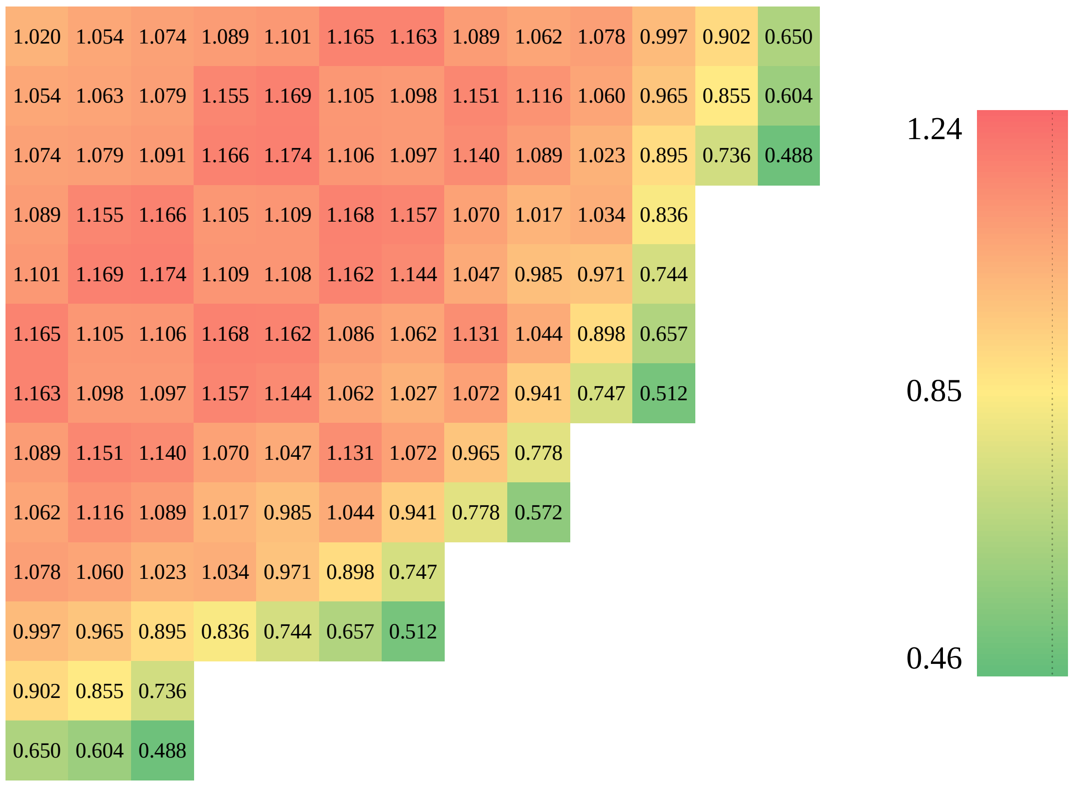
1.018	1.057	1.081	1.102	1.115	1.181	1.174	1.092	1.062	1.070	0.977	0.886	0.639
1.057	1.069	1.091	1.172	1.182	1.118	1.108	1.151	1.116	1.059	0.956	0.842	0.590
1.081	1.091	1.102	1.180	1.185	1.116	1.105	1.141	1.089	1.022	0.890	0.726	0.479
1.102	1.172	1.180	1.117	1.121	1.175	1.158	1.068	1.011	1.020	0.824		
1.115	1.182	1.185	1.121	1.117	1.171	1.146	1.044	0.979	0.959	0.730		
1.181	1.118	1.116	1.175	1.171	1.091	1.064	1.135	1.038	0.889	0.647		
1.174	1.108	1.105	1.158	1.146	1.064	1.032	1.070	0.934	0.740	0.504		
1.092	1.151	1.141	1.068	1.044	1.135	1.070	0.965	0.779				
1.062	1.116	1.089	1.011	0.979	1.038	0.934	0.779	0.570				
1.070	1.059	1.022	1.020	0.959	0.889	0.740						
0.977	0.956	0.890	0.824	0.730	0.647	0.504						
0.886	0.842	0.726										
0.639	0.590	0.479										

Figure 6.17 Radial power profile at 10 MWd/kgHM.



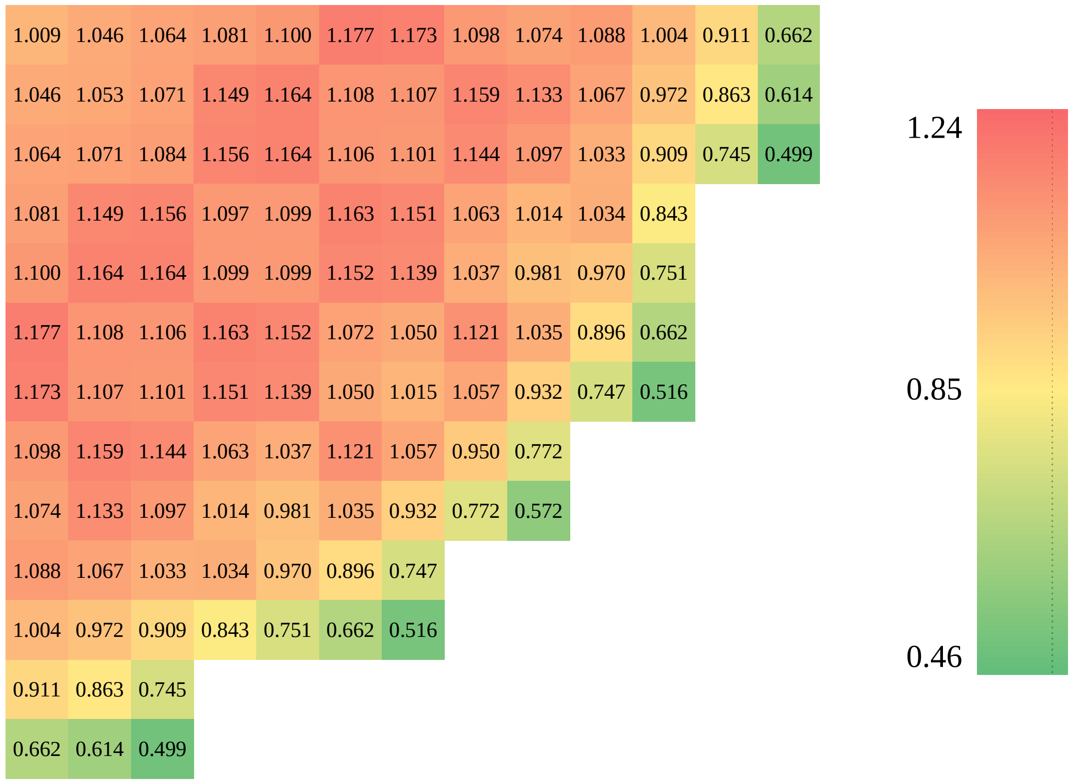
0.978	1.016	1.042	1.065	1.082	1.159	1.170	1.096	1.071	1.092	1.012	0.919	0.665
1.016	1.029	1.053	1.133	1.151	1.099	1.100	1.151	1.123	1.071	0.980	0.875	0.613
1.042	1.053	1.066	1.147	1.158	1.100	1.096	1.142	1.096	1.036	0.910	0.751	0.495
1.065	1.133	1.147	1.093	1.097	1.160	1.154	1.069	1.023	1.041	0.848		
1.082	1.151	1.158	1.097	1.101	1.157	1.147	1.054	0.994	0.982	0.754		
1.159	1.099	1.100	1.160	1.157	1.083	1.065	1.136	1.049	0.907	0.666		
1.170	1.100	1.096	1.154	1.147	1.065	1.026	1.073	0.946	0.754	0.518		
1.096	1.151	1.142	1.069	1.054	1.136	1.073	0.965	0.779				
1.071	1.123	1.096	1.023	0.994	1.049	0.946	0.779	0.574				
1.092	1.071	1.036	1.041	0.982	0.907	0.754						
1.012	0.980	0.910	0.848	0.754	0.666	0.518						
0.919	0.875	0.751										
0.665	0.613	0.495										

Figure 6.18 Radial power profile at 11 MWd/kgHM.



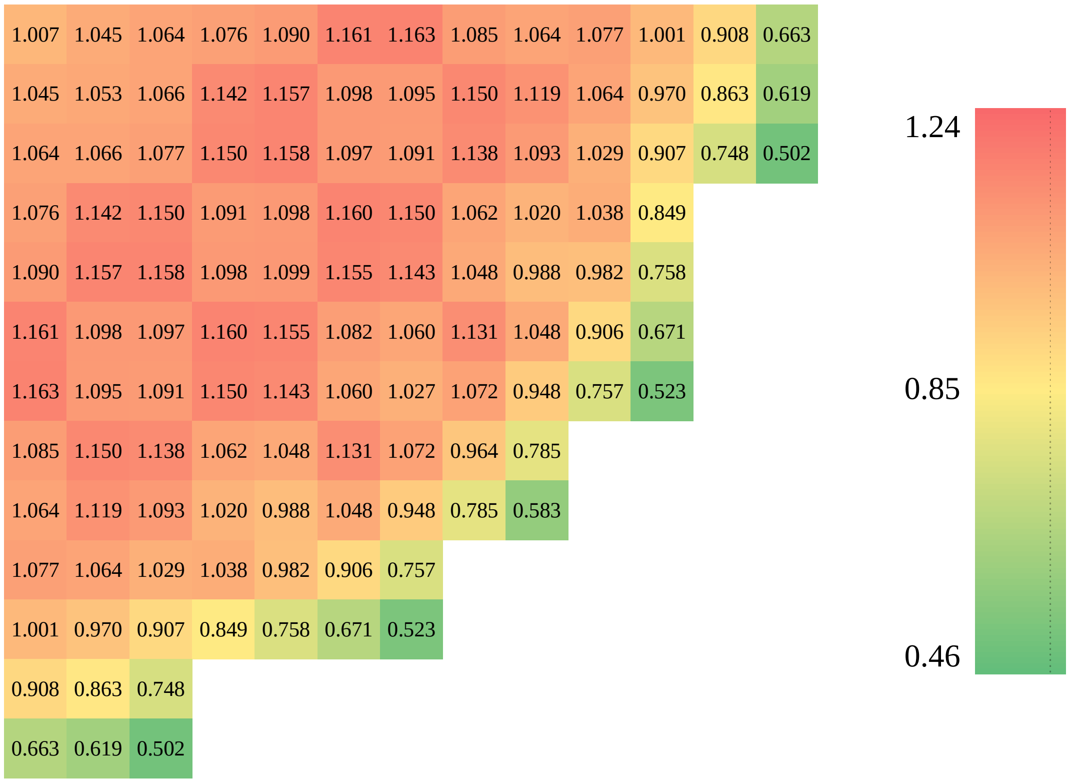
1.020	1.054	1.074	1.089	1.101	1.165	1.163	1.089	1.062	1.078	0.997	0.902	0.650
1.054	1.063	1.079	1.155	1.169	1.105	1.098	1.151	1.116	1.060	0.965	0.855	0.604
1.074	1.079	1.091	1.166	1.174	1.106	1.097	1.140	1.089	1.023	0.895	0.736	0.488
1.089	1.155	1.166	1.105	1.109	1.168	1.157	1.070	1.017	1.034	0.836		
1.101	1.169	1.174	1.109	1.108	1.162	1.144	1.047	0.985	0.971	0.744		
1.165	1.105	1.106	1.168	1.162	1.086	1.062	1.131	1.044	0.898	0.657		
1.163	1.098	1.097	1.157	1.144	1.062	1.027	1.072	0.941	0.747	0.512		
1.089	1.151	1.140	1.070	1.047	1.131	1.072	0.965	0.778				
1.062	1.116	1.089	1.017	0.985	1.044	0.941	0.778	0.572				
1.078	1.060	1.023	1.034	0.971	0.898	0.747						
0.997	0.965	0.895	0.836	0.744	0.657	0.512						
0.902	0.855	0.736										
0.650	0.604	0.488										

Figure 6.19 Radial power profile at 12 MWd/kgHM.



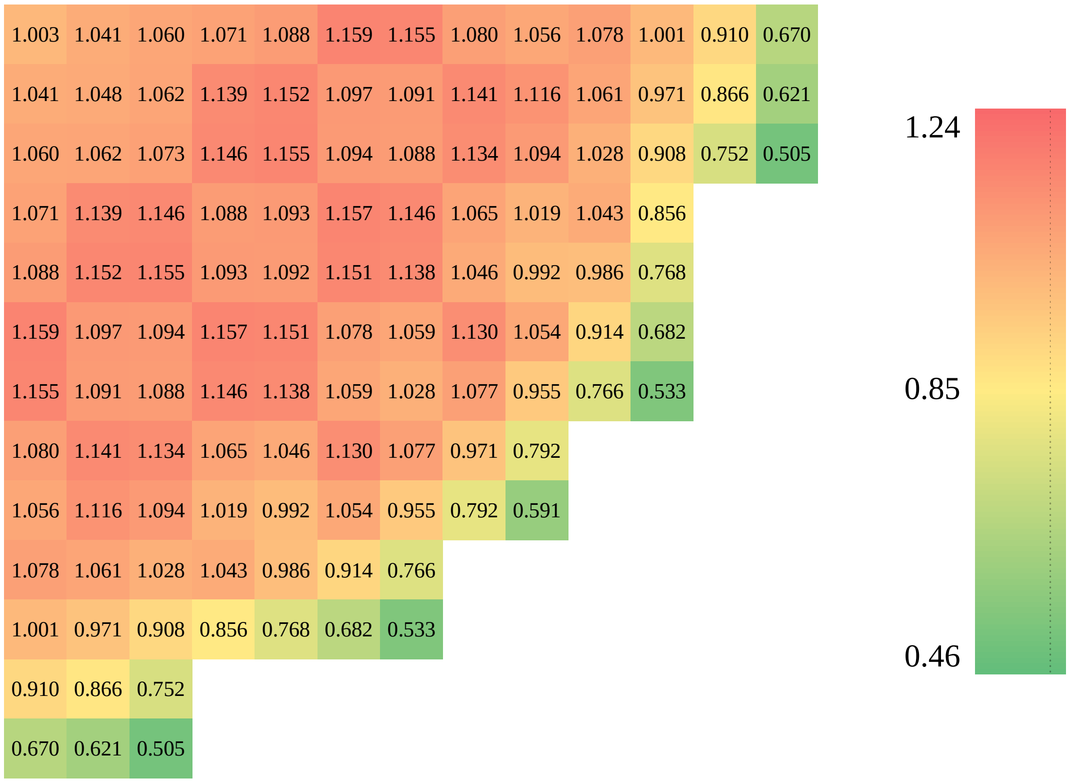
1.009	1.046	1.064	1.081	1.100	1.177	1.173	1.098	1.074	1.088	1.004	0.911	0.662
1.046	1.053	1.071	1.149	1.164	1.108	1.107	1.159	1.133	1.067	0.972	0.863	0.614
1.064	1.071	1.084	1.156	1.164	1.106	1.101	1.144	1.097	1.033	0.909	0.745	0.499
1.081	1.149	1.156	1.097	1.099	1.163	1.151	1.063	1.014	1.034	0.843		
1.100	1.164	1.164	1.099	1.099	1.152	1.139	1.037	0.981	0.970	0.751		
1.177	1.108	1.106	1.163	1.152	1.072	1.050	1.121	1.035	0.896	0.662		
1.173	1.107	1.101	1.151	1.139	1.050	1.015	1.057	0.932	0.747	0.516		
1.098	1.159	1.144	1.063	1.037	1.121	1.057	0.950	0.772				
1.074	1.133	1.097	1.014	0.981	1.035	0.932	0.772	0.572				
1.088	1.067	1.033	1.034	0.970	0.896	0.747						
1.004	0.972	0.909	0.843	0.751	0.662	0.516						
0.911	0.863	0.745										
0.662	0.614	0.499										

Figure 6.20 Radial power profile at 13 MWd/kgHM.



1.007	1.045	1.064	1.076	1.090	1.161	1.163	1.085	1.064	1.077	1.001	0.908	0.663
1.045	1.053	1.066	1.142	1.157	1.098	1.095	1.150	1.119	1.064	0.970	0.863	0.619
1.064	1.066	1.077	1.150	1.158	1.097	1.091	1.138	1.093	1.029	0.907	0.748	0.502
1.076	1.142	1.150	1.091	1.098	1.160	1.150	1.062	1.020	1.038	0.849		
1.090	1.157	1.158	1.098	1.099	1.155	1.143	1.048	0.988	0.982	0.758		
1.161	1.098	1.097	1.160	1.155	1.082	1.060	1.131	1.048	0.906	0.671		
1.163	1.095	1.091	1.150	1.143	1.060	1.027	1.072	0.948	0.757	0.523		
1.085	1.150	1.138	1.062	1.048	1.131	1.072	0.964	0.785				
1.064	1.119	1.093	1.020	0.988	1.048	0.948	0.785	0.583				
1.077	1.064	1.029	1.038	0.982	0.906	0.757						
1.001	0.970	0.907	0.849	0.758	0.671	0.523						
0.908	0.863	0.748										
0.663	0.619	0.502										

Figure 6.21 Radial power profile at 14 MWd/kgHM.



1.003	1.041	1.060	1.071	1.088	1.159	1.155	1.080	1.056	1.078	1.001	0.910	0.670
1.041	1.048	1.062	1.139	1.152	1.097	1.091	1.141	1.116	1.061	0.971	0.866	0.621
1.060	1.062	1.073	1.146	1.155	1.094	1.088	1.134	1.094	1.028	0.908	0.752	0.505
1.071	1.139	1.146	1.088	1.093	1.157	1.146	1.065	1.019	1.043	0.856		
1.088	1.152	1.155	1.093	1.092	1.151	1.138	1.046	0.992	0.986	0.768		
1.159	1.097	1.094	1.157	1.151	1.078	1.059	1.130	1.054	0.914	0.682		
1.155	1.091	1.088	1.146	1.138	1.059	1.028	1.077	0.955	0.766	0.533		
1.080	1.141	1.134	1.065	1.046	1.130	1.077	0.971	0.792				
1.056	1.116	1.094	1.019	0.992	1.054	0.955	0.792	0.591				
1.078	1.061	1.028	1.043	0.986	0.914	0.766						
1.001	0.971	0.908	0.856	0.768	0.682	0.533						
0.910	0.866	0.752										
0.670	0.621	0.505										

Figure 6.22 Radial power profile at 15 MWd/kgHM (EOC).

At the beginning portion of the cycle, the highest radial power factor occurs close to the periphery. After that, the power migrates more to the center of the core. Toward the end portion of the cycle, the highest value occurs in the middle region between the center and the periphery. Throughout this whole process, the magnitude of $F_{\Delta H}$ does not vary that much as observable from Figure 6.1. The only noticeable trend is that it slightly decreases during the depletion, but not by much.

6.3 Computational Summary

The runtime parameters used for the studies completed in this chapter are summarized in Table 6.2. It was believed that grouping together all the descriptions here would be more concise than listing them individually. All simulations summarized in Table 6.2 ran Serpent version 2.1.25, used processors with 8 GB of memory each, were run in parallel using OpenMP, and used the ENDF/B-VII cross section library.

Table 6.2 Computational summary of first core depletion analysis.

Burnup [MWd/kg]	Particles/ Cycle	Active Cycles	Inactive Cycles	Number of Processors	Wall Time [h]	CPU Time [h]	k	k error [pcm]
0 (HZP)	225000	10000	250	8	47.0	376.4	0.99995	1
0	265000	500	100	30	23.3	699.4	1.00000	5
0.15	265000	500	100	30	27.5	823.5	1.00009	10
0.5	193000	500	100	32	19.8	633.7	0.99982	12
1	230000	500	100	32	24.3	779.1	1.00003	12
2	220000	500	100	32	24.7	790.6	1.00009	12
3	195000	500	100	32	23.0	689.5	0.99995	13
4	178000	500	100	32	19.6	626.4	1.00038	14
5	210000	500	100	32	22.6	723.7	0.99991	13
6	100000	500	100	32	11.3	361.8	1.00034	18
7	100000	500	100	32	11.2	357.3	0.9995	18
8	100000	500	100	32	11.2	359.5	1.0007	19
9	100000	500	100	32	11.4	365.1	0.99992	18
10	100000	500	100	32	11.1	356.6	1.00031	18
11	100000	500	100	32	11.2	359.5	0.99955	19

Table 6.2 continued.

Burnup [MWd/kg]	Particles/ Cycle	Active Cycles	Inactive Cycles	Number of Processors	Wall Time [h]	CPU Time [h]	k	k error [pcm]
12	100000	500	100	32	11.1	356.0	1.00029	19
13	100000	500	100	32	11.4	366.0	0.99989	19
14	200000	500	100	32	22.5	718.8	1.00134	13

6.4 Conclusions

From the outset of the work for this fuel cycle and its depletion, the main goal was to build off of the 2D Serpent model and take any lessons learned from collaboration with WEC to create a 3D model with a core loading pattern believed to be viable over the cycle. In some regards, this was achieved quite well. Whereas the 2D Serpent model had large radial peaking factors (Figure 4.1) especially along the periphery, the 3D model tried to remedy this by changing the enrichment scheme to load lower relative enrichment on the periphery and higher relative enrichment near the center (Figure 3.3 versus Figure 5.1). The success of this change can be observed in the $F_{\Delta H}$ results of Figure 6.1.

Early in the cycle depletion, the axial results are quite good and behave as expected. Hot channel factors for F_z and F_q behave notably well through 6 MWd/kgHM. The F_z values at 12, 14, and 15 MWd/kgHM are slightly higher than one would expect, especially given that the 13 MWd/kgHM result seems less tilted. This likely indicates that the thermal hydraulics toward the end of the cycle were not very accurate and some iteration might have been necessary to capture the correct behavior. It is likely that the F_z and F_q would be lower if the thermal hydraulics were converged. This would have led to better performance at the end of the cycle. The intent of this analysis was just to demonstrate applicability to the I^{2}S-LWR system, but implementing thermal hydraulic iteration between burnup steps in the future for more accurate studies would be the correct course of action.

As mentioned earlier, another design goal for this first cycle core was that it should aim to operate slightly longer than the equilibrium 12-month fuel cycle. 17 days were assumed by the I²S-LWR design team as being necessary to refuel the core and conduct plant maintenance during a planned outage. The core loading pattern used in this analysis was able to achieve an average burnup of 15 MWd/kgHM; producing 2850 MW_t over 438 EFPD with 63 ppm soluble boron at EOC. Considering the time allotted for an outage, the entire fuel cycle length would be about 455 days (15 months). First, the goal to keep the EOC CBC above 50 ppm was achieved. Next, the cycle duration of 438 EFPD or 15 months for the entire fuel cycle aligns well with the intention of operating slightly past 12 months. Third, all of the above were achieved with using only four different fuel enrichments. While trivial by itself, the simplicity of having four enrichments that yield promising results should be appreciated. Some other designs are fairly complicated to achieve their goals; however, one advantage these other designs have is that they are able to quickly approach their equilibrium cycle. While the approach to an equilibrium cycle from this first cycle is well beyond the scope of this thesis, the process should at least be simplified since there are less options to consider.

Moving forward, future core loading patterns should incorporate more axial reactivity control mechanisms. Specifically, the initial axial profile retains too much of the classic cosine shape. The easiest solution to remedy this would be to use more IFBA cutback at both the top and bottom of the central fuel section (between the blanket regions). As with this analysis, more cutback should still be used at the top of the core to keep AO near zero. This approach would flatten the initial axial profile— reducing F_z and F_q at the beginning of the cycle. Combining this improvement with better convergence on the thermal hydraulics, it is believed that F_z could be kept below 1.35 and F_q below 1.6 over the entire first fuel cycle.

Using finer steps is another way to make the results more accurate. More axial segments would make the neutronics-thermal hydraulic feedback more true to reality as well as reduce the magnitude of the reactivity jumps at the segment interfaces evident in Figure 6.3. Shorter depletion steps would also improve the accuracy and have the added benefit of likely requiring fewer thermal hydraulic iterations between burn steps since the power profiles between finer steps should be more similar. In the opinion of the researcher, these two approaches would likely be less impactful than the two changes discussed at the end of the previous paragraph. However, they are addressed for completeness since they would make the model closer to reality.

6.5 Future Work

The direction for immediate future work lies in the C++ script for thermal hydraulic iteration coupled with depletion. An element which will be key for achieving this, at least in LWR systems like I^2S-LWR, will be implementing a critical boron algorithm. Right now, CBC is adjusted manually based upon the results of previous simulations. If the script is able to estimate the worth of soluble boron at the current state of the cycle, the CBC could be adjusted independently from user input.

Right now, the script works entirely independently from Serpent and both differing material compositions and individual burnup steps must be adjusted and started manual after running the script. One improvement is to modify the script so that it can work in an automated fashion between different cases – both for thermal hydraulic iteration and then progressing among depletion steps. This benefit of converging on both the source distribution and materials definitions would greatly reduce the magnitude of oscillations such as those seen axially at near EOC for this thesis. These automated thermal hydraulic – neutronic iterations should continue until an acceptable level of convergence is achieved. Once a user-specfied tolerance is met, the script should move Serpent onto the next phase of the depletion process until completion.

Currently, the C++ script will only work with the I²S-LWR geometry. A major goal of longer-term future development of the script would be to generalize it to different geometries, pin configurations, and even coolant systems. While ambitious, this would make automated depletion in Serpent possible for not only square-lattice LWR systems, but possibly hexagonal geometries coolant by metals or salts. By being more versatile and inclusive of other reactor designs, the script's application scope broadens and the likelihood of it being used beyond its development also rises.

Future work will also extend beyond depletion studies like those covered in this chapter. With an established 3D Serpent model of the I²S-LWR core, additional analyses for I²S-LWR are possible. The first of interest would be to investigate the neutronics impact of heterogeneous radial reflector channels. This would be of great benefit to the I²S-LWR project at large since many assumptions currently surround the radial reflector design and having actual results might significantly impact the radial reflector's final geometry. Another possibility might be to investigate other first core or even equilibrium core designs of the I²S-LWR, which would be easily implementable with the C++ script for creating burned materials from Serpent output files.

APPENDIX A C++ SCRIPT USED TO CREATE COOLANT, FUEL, AND IFBA MATERIALS

The following script was used to create the coolant, fuel, and IFBA materials for the 65 radially unique elements for each of the 10 axial fuel regions, each of the 10 radial reflector regions, and both the top and bottom axial reflectors. The script reads in power distribution data output by Serpent as well as the desired boron centration and general core thermal hydraulic parameters to produce the materials to be used.

```
//Single Channel Analysis Code - Strictly 1 Phase
//Ramey

//Initializing libraries to be used
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <iomanip>
#include <unistd.h>

using namespace std;  //Indicates standard naming conventions within C++

//Now initializing functions to be used
void startup (void);  //Splash function
void callInput (void);  //Looks for input file
void readInput (void);  //Reads the input file
void callNeutronics (void); //Asks for neutronics input file name
void readNeutronics (void); //Reads from the neutronics input file
void invertNeutronics (void); //Puts the input data in arrays needed for computation
later
void readProp (void);  //Reads the thermodynamic property file
void callWrite (void);  //Asks user for output file name
void computeZ (void);
void interpolate (void);
void computeTm (void);
void computeTave (void); //Force average coolant temperature to be the prescribed value
void findAxialOffset (void);
void computeTco (void);
void computeTci (void);
void computeTfo (void);
void computeTmax (void);
void axialCollapse (void);
void coolantMaker (void);
void writeOut (void);  //Outputs the desired data to the user-specified file

// Initializing global variables to be used:
bool subcooled = false;  //Subcooled boiling flag
char filenameIn [] = "input.txt";  //Input file name
char neutronicsIn [] = "neutronics5.txt"; //Serpent tally output filename
char filenameOut [50]; //Output file name
char propfileIn [] = "inProp.txt";  //Property file name
char hotChannelOut [] = "";
```

```
double propArray [24][6];  //Thermodynamic properties array
double inArray [10]; //Input file array
double neutronicsArray [96][169];
double flippedNArray [96][169];
double reducedArray [96][121];
double powerArray [96][121];
double symmPowerArray [96][65];
double axialPower [96];
double radialPower[121];
double section121 [10][121];
double sectionNeutArray [10][65];
double axialUnits [10] = { 4, 11, 12, 12, 12, 12, 12, 12, 5, 4};
int IFBAAssemblies [26] =
{2,3,6,7,10,11,16,17,20,21,27,28,31,32,39,40,43,44,46,47,50,51,54,55,59,60}; // The 26
assembly groups that use IFBA
int HEnrich [21] = {12,13,24,25,35,36,43,44,50,51,54,55,56,57,59,60,61,62,63,64,65};
int MEnrich [20] = {6,7,10,11,16,17,20,21,22,23,27,28,31,32,33,34,39,40,46,47};
int LEnrich [36] = {1,2,3,4,5,8,9,14,15,18,19,26,29,30,37,38,41,42,45,48,49,52,53,58};
double z [96]; //Axial location [m]
double Tmax [96][65]; //Centerline temperature [C]
double Tfo [96][65]; //Outside fuel temperature [C]
double Tci [96][65]; //Inside clad temperature [C]
double Tco [96][65]; //Outside clad temperature [C]
double Tm [96][65]; //Mean temperature [C]
double hz [96][65]; //Enthalpy at z [J/kg]
double TmaxC [10][65]; //Centerline temperature [C]
double TfoC [10][65]; //Outside fuel temperature [C]
double TciC [10][65]; //Inside clad temperature [C]
double TcoC [10][65]; //Outside clad temperature [C]
double TmC [10][65]; //Mean temperature [C]
double hzC [10][65]; //Enthalpy at z [J/kg]
double denfC [10][65];
double denCoolantC [10][65];
double denRefC [10];
double TfaveC [10][65];
double TSliceAve[10];
double symmNeutArray [96][65];
double L = 0; //Flow channel length [m]
double Dco = 0; //Outer cladding diameter [m]
double S = 0; //Pitch [m]
double Tmin = 0; //Inlet mean fluid temperature [C]
double Texit = 0; // Core exit temperature
double md = 0; //Mass flow rate [kg/s]
double Power = 0;
double P = 0; //Nominal reactor pressure [Pa]
double Dci = 0; //Inner cladding diameter [m]
double Dfo = 0; //Outer fuel diameter [m]
double vf [96][65];  //Fluid specific volume [m^3/kg]
double denf [96][65];  //Fluid density [kg/m^3]
double hf [96][65];  //Fluid enthalpy [J/kg]
double uf [96][65];  //Fluid viscosity [kg/(m*s)]
double kf [96][65];  //Fluid thermal conductivity [W/(m*K)]
double Prf [96][65];  //Fluid Prandtl number
double g = 9.81; //Gravity constant [m/s^2]
double Cfit = 0; //Single-phase C
//double fit[73]; //Single-phase friction factor
double n = 0.18; //For use in the single-phase pressure drop
double Dh = 0;  //Hydraulic diameter [m]
double Rco = 0;  //Outer cladding radius [m]
double Rci = 0;  //Inner cladding radius [m]
double Rfo = 0; //Outer fuel radius [m]
double const pi = 4*atan(1);
double G = 0; //Mass flux [kg/(s*m^2)]
double qbot[96][65]; //Linear heat rate [W/m]
double qprime[96][65];  //Linear heat rate [W/m]
double qtop [96][65]; //Linear heat rate [W/m]
double henter = 0;  //Entrance enthalpy [J/kg]
double hexit = 0; //Exit enthalpy [J/kg]
double psi = 0;  //Psi used for flows through channels not modeled as circular tubes
double Relo[96][65];  //Liquid-only Reynolds number
double Area = 0;  //Flow Area [m^2]
double Ttable = 0; // Temperature from table
double axialOffset = 0;
double denfin = 0;
double hfin = 0;
double ufin = 0;
double kfin = 0;
double Prfin = 0;
double vfin = 0;
double Reloin = 0;
double maxOutletTemp = 0;
double CBC = 0;
double atomDHCoolant[10][65];
double atomDOCoolant [10][65];
double atomB10Coolant [10][65];
double atomB11Coolant [10][65];
double denSliceAve [10];
double atomDHRef [10];
double atomDORef [10];
double atomB10Ref [10];
double atomB11Ref [10];
double atomDHBARef = 0;
double atomDOBARef = 0;
double atomB10BARef = 0;
double atomB11BARef = 0;
double atomDHTARef = 0;
double atomDOTARef = 0;
double atomB10TARef = 0;
double atomB11TARef = 0;
double maxFuelTemp = 0;
double denInter[10];
double atomDHInter[10];
double atomDOInter[10];
double atomB10Inter[10];
double atomB11Inter[10];
double avgCladTemp = 0;
double avgGapTemp = 0;
double avgFuelTemp = 0;
double avgCenterline = 0;
double Fq = 0;
double Fz = 0;
double FdH = 0;
int i = 0;
int j = 0;
int k = 0;
int r = 0;
int countr = 0;
ifstream prop_file;  //Allows for thermodynamic properties to be read
ifstream in_file;  //Allows for the user-specified input file to be read
ofstream out_file;  //Allows program to write the results for the given case

int main (void) {  //The entire program runs in this function
    startup();
    callInput();
return 0;
}

void startup (void){ //This only appears the first time the program is run.
    cout << "Single Channel Analysis Code\n\n";
    cout << "This program gives physical values along 96 flow elements.\n\n";
    cout << "The thermodynamic properties must be given \n";
    cout << " in the following order in the specified units:\n";
    cout << "T[C] den[kg/m^3] h[J/kg] mu[kg/(m*s)] kf[W/(m*K)] Prf[-]\n\n";
}

void callInput (void) {  //The user inputs the name of the channel data input file.  Must
be in units and format expected.
    cout << "The channel data input file must be named 'input.txt' Trying to read
now...\n";
    /* cout << "The values must be given in the following order with the specified units:
\n \n";
    cout << "Length [m] \n";
    cout << "Diameter [m] \n";
    cout << "Pitch [m] \n";
    cout << "Tm,in [C] \n";
    cout << "Thermal Power [W] \n";
    cout << "Nominal Pressure [Pa] \n";
    cout << "Outlet Temperature [C] \n";
    cout << "Inner Cladding Diameter [m] \n";
    cout << "Outer Fuel Diameter [m] \n";
    cout << "Critical Boron Concentration [ppm]\n\n";*/
    in_file.open(filenameIn);
    if (in_file.fail()) {
        in_file.clear();
        cout << "Could not open 'input.txt' file. Please try program again.\n \n \n";}
        else {
            in_file.close();
            readInput();
            cout << "   Input File Accepted\n\n";
            readProp();}
}

void readInput (void){ //Reads in the user-specified data
    in_file.open(filenameIn);
    for (j=0;j<10;j++){in_file >> inArray[j];}
    in_file.close();
    //Assigns the input array values to variables
    L = inArray[0];
    Dco = inArray[1];
    S = inArray[2];
    Tmin = inArray[3];
    Power = inArray[4];
    P = inArray[5];
    Texit = inArray[6];
    Dci = inArray[7];
    Dfo = inArray[8];
    CBC = inArray[9];
    double pdr = S/Dco;
    Area = S*S - (pi*Dco*Dco/4);
    Dh = 4*(Area)/(pi*Dco);
    //This accounts for the two different expressions for Cfit
    //if (pdr <= 1.1){Cfit = 0.09435 + 0.5806*(pdr-1) - 1.359*(pdr-1)*(pdr-1);}
    //    else {Cfit = 0.1339+ 0.09059*(pdr-1) - 0.09926*(pdr-1)*(pdr-1);}
    Rco = Dco/2;
    Rci = Dci/2;
    Rfo = Dfo/2;
    md = Power / (121*336*6084*(Texit-Tmin)); //Initial guess at md
    G = md/Area;
    psi = 1.826*S/Dco - 1.0430;
}

void readProp (void){ //Reads in the thermo property data
    prop_file.open(propfileIn);
    for (j=0;j<24;j++){for (i=0;i < 6; i++){prop_file >> propArray[j][i];}}
    prop_file.close();
    denfin = propArray[0][1];
    hfin = propArray[0][2];
    ufin = propArray[0][3];
    kfin = propArray[0][4];
    Prfin = propArray[0][5];
    vfin = 1/denfin;
    Reloin = G*Dh/ufin; //Liquid only Reynolds number
    callNeutronics();
}

void callNeutronics (void){ //Try to open the neutronics input file
    cout << "The neutronics input file must be named 'neutronics5.txt'. Trying to read
now...\n";
    in_file.open(neutronicsIn);
    if (in_file.fail()) {
        in_file.clear();
        cout << "Could not open neutronics file " << neutronicsIn << ". Please address
and try program again.\n\n\n";}
```

```
        else {
            in_file.close();
            cout << "   Neutronics File Accepted\n\n";
            readNeutronics();}
}

void readNeutronics (void){ //Read data from neutronics input file
    bool doneReading = false;
    i = 0;
    j = 0;
    k = 0;
    int l = 0;
    int linnumber = 3; //Line Number
    string wholeline;
    string component;
    char charnumb [20];
    double value = 0;
    in_file.open(neutronicsIn);
    do {
        getline(in_file, wholeline );
        k++;
        if (k == linnumber){
            component = wholeline.substr (52,11);
            for (l = 0; l < 11; l++){charnumb[l] = component[l];}
            value = atof(charnumb);
            neutronicsArray[i][j] = value;
            j++;
            linnumber++;
            if (j == 169) {
                i++;
                j = 0;}
            if (i == 96) {doneReading = true;}}
    }while (!doneReading);
    in_file.close();
    invertNeutronics();
}

void invertNeutronics (void){ //Manipulate data from neutronics input file
    double maxValue = 0;
    for (i = 0; i < 96; i++){ // Need to flip about horizontal axis
        for (j = 0; j < 13; j++) {flippedNArray[i][j] = neutronicsArray[i][j+156];}
        for (j = 13; j < 26; j++) {flippedNArray[i][j] = neutronicsArray[i][j+130];}
        for (j = 26; j < 39; j++) {flippedNArray[i][j] = neutronicsArray[i][j+104];}
        for (j = 39; j < 52; j++) {flippedNArray[i][j] = neutronicsArray[i][j+78];}
        for (j = 52; j < 65; j++) {flippedNArray[i][j] = neutronicsArray[i][j+52];}
        for (j = 65; j < 78; j++) {flippedNArray[i][j] = neutronicsArray[i][j+26];}
        for (j = 78; j < 91; j++) {flippedNArray[i][j] = neutronicsArray[i][j];}
        for (j = 91; j < 104; j++) {flippedNArray[i][j] = neutronicsArray[i][j-26];}
        for (j = 104; j < 117; j++) {flippedNArray[i][j] = neutronicsArray[i][j-52];}
        for (j = 117; j < 130; j++) {flippedNArray[i][j] = neutronicsArray[i][j-78];}
        for (j = 130; j < 143; j++) {flippedNArray[i][j] = neutronicsArray[i][j-104];}
        for (j = 143; j < 156; j++) {flippedNArray[i][j] = neutronicsArray[i][j-130];}
        for (j = 156; j < 169; j++) {flippedNArray[i][j] = neutronicsArray[i][j-156];}}
    for (i=0;i<96;i++){ // Collapse the 169 13x13 array into the 121 actual assemblies
        for (j=0;j<50;j++){reducedArray[i][j]=flippedNArray[i][j];}
        for (j=50;j<61;j++){reducedArray[i][j]=flippedNArray[i][j+2];}
        for (j=61;j<72;j++){reducedArray[i][j]=flippedNArray[i][j+4];}
        for (j=72;j<83;j++){reducedArray[i][j]=flippedNArray[i][j+6];}
        for (j=83;j<92;j++){reducedArray[i][j]=flippedNArray[i][j+8];}
        for (j=92;j<101;j++){reducedArray[i][j]=flippedNArray[i][j+12];}
        for (j=101;j<108;j++){reducedArray[i][j]=flippedNArray[i][j+16];}
        for (j=108;j<115;j++){reducedArray[i][j]=flippedNArray[i][j+22];}
        for (j=115;j<118;j++){reducedArray[i][j]=flippedNArray[i][j+28];}
        for (j=118;j<121;j++){reducedArray[i][j]=flippedNArray[i][j+38];}}
    for (i=0;i<96;i++){ // Average across symmetry groups
        symmNeutArray[i][1-1] = reducedArray[i][1-1];
        symmNeutArray[i][2-1] = 0.5*reducedArray[i][2-1] + 0.5*reducedArray[i][14-1];
        symmNeutArray[i][3-1] = 0.5*reducedArray[i][3-1] + 0.5*reducedArray[i][27-1];
        symmNeutArray[i][4-1] = 0.5*reducedArray[i][4-1] + 0.5*reducedArray[i][40-1];
        symmNeutArray[i][5-1] = 0.5*reducedArray[i][5-1] + 0.5*reducedArray[i][51-1];
        symmNeutArray[i][6-1] = 0.5*reducedArray[i][6-1] + 0.5*reducedArray[i][62-1];
        symmNeutArray[i][7-1] = 0.5*reducedArray[i][7-1] + 0.5*reducedArray[i][73-1];
        symmNeutArray[i][8-1] = 0.5*reducedArray[i][8-1] + 0.5*reducedArray[i][84-1];
        symmNeutArray[i][9-1] = 0.5*reducedArray[i][9-1] + 0.5*reducedArray[i][93-1];
        symmNeutArray[i][10-1] = 0.5*reducedArray[i][10-1] + 0.5*reducedArray[i][102-1];
        symmNeutArray[i][11-1] = 0.5*reducedArray[i][11-1] + 0.5*reducedArray[i][109-1];
        symmNeutArray[i][12-1] = 0.5*reducedArray[i][12-1] + 0.5*reducedArray[i][116-1];
        symmNeutArray[i][13-1] = 0.5*reducedArray[i][13-1] + 0.5*reducedArray[i][119-1];
        symmNeutArray[i][14-1] = reducedArray[i][15-1];
        symmNeutArray[i][15-1] = 0.5*reducedArray[i][16-1] + 0.5*reducedArray[i][28-1];
        symmNeutArray[i][16-1] = 0.5*reducedArray[i][17-1] + 0.5*reducedArray[i][41-1];
        symmNeutArray[i][17-1] = 0.5*reducedArray[i][18-1] + 0.5*reducedArray[i][52-1];
        symmNeutArray[i][18-1] = 0.5*reducedArray[i][19-1] + 0.5*reducedArray[i][63-1];
        symmNeutArray[i][19-1] = 0.5*reducedArray[i][20-1] + 0.5*reducedArray[i][74-1];
        symmNeutArray[i][20-1] = 0.5*reducedArray[i][21-1] + 0.5*reducedArray[i][85-1];
        symmNeutArray[i][21-1] = 0.5*reducedArray[i][22-1] + 0.5*reducedArray[i][94-1];
        symmNeutArray[i][22-1] = 0.5*reducedArray[i][23-1] + 0.5*reducedArray[i][103-1];
        symmNeutArray[i][23-1] = 0.5*reducedArray[i][24-1] + 0.5*reducedArray[i][110-1];
        symmNeutArray[i][24-1] = 0.5*reducedArray[i][25-1] + 0.5*reducedArray[i][117-1];
        symmNeutArray[i][25-1] = 0.5*reducedArray[i][26-1] + 0.5*reducedArray[i][120-1];
        symmNeutArray[i][26-1] = reducedArray[i][29-1];
        symmNeutArray[i][27-1] = 0.5*reducedArray[i][30-1] + 0.5*reducedArray[i][42-1];
        symmNeutArray[i][28-1] = 0.5*reducedArray[i][31-1] + 0.5*reducedArray[i][53-1];
        symmNeutArray[i][29-1] = 0.5*reducedArray[i][32-1] + 0.5*reducedArray[i][64-1];
        symmNeutArray[i][30-1] = 0.5*reducedArray[i][33-1] + 0.5*reducedArray[i][75-1];
        symmNeutArray[i][31-1] = 0.5*reducedArray[i][34-1] + 0.5*reducedArray[i][86-1];
        symmNeutArray[i][32-1] = 0.5*reducedArray[i][35-1] + 0.5*reducedArray[i][95-1];
        symmNeutArray[i][33-1] = 0.5*reducedArray[i][36-1] + 0.5*reducedArray[i][104-1];
        symmNeutArray[i][34-1] = 0.5*reducedArray[i][37-1] + 0.5*reducedArray[i][111-1];
        symmNeutArray[i][35-1] = 0.5*reducedArray[i][38-1] + 0.5*reducedArray[i][118-1];
        symmNeutArray[i][36-1] = 0.5*reducedArray[i][39-1] + 0.5*reducedArray[i][121-1];
        symmNeutArray[i][37-1] = reducedArray[i][43-1];
        symmNeutArray[i][38-1] = 0.5*reducedArray[i][44-1] + 0.5*reducedArray[i][54-1];
        symmNeutArray[i][39-1] = 0.5*reducedArray[i][45-1] + 0.5*reducedArray[i][65-1];
        symmNeutArray[i][40-1] = 0.5*reducedArray[i][46-1] + 0.5*reducedArray[i][76-1];
        symmNeutArray[i][41-1] = 0.5*reducedArray[i][47-1] + 0.5*reducedArray[i][87-1];
        symmNeutArray[i][42-1] = 0.5*reducedArray[i][48-1] + 0.5*reducedArray[i][96-1];
        symmNeutArray[i][43-1] = 0.5*reducedArray[i][49-1] + 0.5*reducedArray[i][105-1];
        symmNeutArray[i][44-1] = 0.5*reducedArray[i][50-1] + 0.5*reducedArray[i][112-1];
        symmNeutArray[i][45-1] = reducedArray[i][55-1];
        symmNeutArray[i][46-1] = 0.5*reducedArray[i][56-1] + 0.5*reducedArray[i][66-1];
        symmNeutArray[i][47-1] = 0.5*reducedArray[i][57-1] + 0.5*reducedArray[i][77-1];
        symmNeutArray[i][48-1] = 0.5*reducedArray[i][58-1] + 0.5*reducedArray[i][88-1];
        symmNeutArray[i][49-1] = 0.5*reducedArray[i][59-1] + 0.5*reducedArray[i][97-1];
        symmNeutArray[i][50-1] = 0.5*reducedArray[i][60-1] + 0.5*reducedArray[i][106-1];
        symmNeutArray[i][51-1] = 0.5*reducedArray[i][61-1] + 0.5*reducedArray[i][113-1];
        symmNeutArray[i][52-1] = reducedArray[i][67-1];
        symmNeutArray[i][53-1] = 0.5*reducedArray[i][68-1] + 0.5*reducedArray[i][78-1];
        symmNeutArray[i][54-1] = 0.5*reducedArray[i][69-1] + 0.5*reducedArray[i][89-1];
        symmNeutArray[i][55-1] = 0.5*reducedArray[i][70-1] + 0.5*reducedArray[i][98-1];
        symmNeutArray[i][56-1] = 0.5*reducedArray[i][71-1] + 0.5*reducedArray[i][72-1];
        symmNeutArray[i][57-1] = 0.5*reducedArray[i][72-1] + 0.5*reducedArray[i][114-1];
        symmNeutArray[i][58-1] = reducedArray[i][79-1];
        symmNeutArray[i][59-1] = 0.5*reducedArray[i][80-1] + 0.5*reducedArray[i][90-1];
        symmNeutArray[i][60-1] = 0.5*reducedArray[i][81-1] + 0.5*reducedArray[i][99-1];
        symmNeutArray[i][61-1] = 0.5*reducedArray[i][82-1] + 0.5*reducedArray[i][108-1];
        symmNeutArray[i][62-1] = 0.5*reducedArray[i][83-1] + 0.5*reducedArray[i][115-1];
        symmNeutArray[i][63-1] = reducedArray[i][91-1];
        symmNeutArray[i][64-1] = 0.5*reducedArray[i][92-1] + 0.5*reducedArray[i][100-1];
        symmNeutArray[i][65-1] = reducedArray[i][101-1];}
    for (i=0;i<96;i++){for (j=0;j<65;j++){if (symmNeutArray[i][j] > maxValue) {maxValue =
symmNeutArray[i][j];}}}
    double total = 0;
    for (i=0;i<96;i++){for (j=0;j<121;j++){total += reducedArray[i][j];}}
    double averageNeutronics = 0;
    averageNeutronics = total/11616; // 11616=121*96
    Fq = maxValue / averageNeutronics;
    double averagePower = 0;
    averagePower = Power / (4*11616); // 4 for quarter core
    double averagePinPower = 0;
    averagePinPower = averagePower * 100 / (1.5*2.54*84); // W/m value
    for (i=0;i<96;i++){for(j=0;j<121;j++){powerArray[i][j] = reducedArray[i][j] *
averagePinPower / averageNeutronics;}} //W/m
    for (i=0;i<96;i++){for(j=0;j<65;j++){symmPowerArray[i][j] = symmNeutArray[i][j] *
averagePinPower / averageNeutronics;}} //W/m
    callWrite();
}

void callWrite (void){ //User specifies the name of the results file
    cout << "Please give the file name where the results will be written. \n";
    cout << "*Warning: Program will overwrite data of existing files.* \n";
    cout << "Please choose your file name wisely or write to a new file. \n \n";
    cout << "File name: ";
    cin >> filenameOut;
    cout << "\n\n";
    computeZ();
}

void computeZ (void){  //Axial position and linear heat rate at that position
    double zmid = 0;
    z[96] = L/2;
    for (i = 0 ; i < 96 ; i++){
        z[i] = -1*L/2 + L*i/96;
        for (j=0;j<65;j++) {
            if (i==0){qbot[0][j] = 0.5*symmPowerArray[0][j];}
                else{qbot[i][j] = 0.5*(symmPowerArray[i][j]+symmPowerArray[i-1][j]);}
            qprime[i][j] = symmPowerArray[i][j];
            if (i==95){qtop[95][j] = 0.5*symmPowerArray[95][j];}
                else{qtop[i][j] = 0.5*(symmPowerArray[i][j]+symmPowerArray[i+1][j]);}}}
    computeTm();
}

void computeTm (void){  //Mean coolant temperature
    countr = 0;
    int next = 1;
    do{
        for (j=0;j<65;j++){
            if (countr == 0){ // For the bottom elements only
                for(i=0;i<65;i++){
                    hz[0][i] = hfin + qbot[0][i]*L/(96 * md);
                    Tm[0][i] = ((propArray[1][0]-propArray[0][0])/(propArray[1][2] -
propArray[0][2]))*(hz[0][i] - propArray[0][2]) + propArray[0][0];}} //Assume dT is less
than 2C at bottom
            hz[next][j] = hz[countr][j] + qbot[countr][j]*L/(96 * md);
            if (hz[next][j] <= propArray[1][2]){r = 0;}
            else if (hz[next][j] <= propArray[2][2]) {r = 1;}
            else if (hz[next][j] <= propArray[3][2]) {r = 2;}
            else if (hz[next][j] <= propArray[4][2]) {r = 3;}
            else if (hz[next][j] <= propArray[5][2]) {r = 4;}
            else if (hz[next][j] <= propArray[6][2]) {r = 5;}
            else if (hz[next][j] <= propArray[7][2]) {r = 6;}
            else if (hz[next][j] <= propArray[8][2]) {r = 7;}
            else if (hz[next][j] <= propArray[9][2]) {r = 8;}
            else if (hz[next][j] <= propArray[10][2]) {r = 9;}
            else if (hz[next][j] <= propArray[11][2]) {r = 10;}
            else if (hz[next][j] <= propArray[12][2]) {r = 11;}
            else if (hz[next][j] <= propArray[13][2]) {r = 12;}
            else if (hz[next][j] <= propArray[14][2]) {r = 13;}
            else if (hz[next][j] <= propArray[15][2]) {r = 14;}
            else if (hz[next][j] <= propArray[16][2]) {r = 15;}
            else if (hz[next][j] <= propArray[17][2]) {r = 16;}
            else if (hz[next][j] <= propArray[18][2]) {r = 17;}
            else if (hz[next][j] <= propArray[19][2]) {r = 18;}
            else if (hz[next][j] <= propArray[20][2]) {r = 19;}
            else if (hz[next][j] <= propArray[21][2]) {r = 20;}
            else if (hz[next][j] <= propArray[22][2]) {r = 21;}
            else if (hz[next][j] <= propArray[23][2]) {r = 22;}
            else {r = 23;}
            Tm[next][j] = (((propArray[r+1][0] - propArray[r][0])/(propArray[r+1][2] -
propArray[r][2]))*(hz[next][j] - propArray[r][2])) + propArray[r][0];}
        countr++;
        next++;
    } while (countr < 96);
    for (i=0;i<96;i++){for(j=0;j<65;j++){interpolate();}} // Gets properties for all
these things
    computeTave();
}

void interpolate (void) {  //Finds thermodynamic properties at each element
    r = 0; //Thermodynamic property table row counter
    if (Tm[i][j] <= 300){r = 0; Ttable = 298;}
    else if (Tm[i][j] <= 302) {r = 1; Ttable = 300;}
    else if (Tm[i][j] <= 304) {r = 2; Ttable = 302;}
    else if (Tm[i][j] <= 306) {r = 3; Ttable = 304;}
    else if (Tm[i][j] <= 308) {r = 4; Ttable = 306;}
    else if (Tm[i][j] <= 310) {r = 5; Ttable = 308;}
    else if (Tm[i][j] <= 312) {r = 6; Ttable = 310;}
    else if (Tm[i][j] <= 314) {r = 7; Ttable = 312;}
    else if (Tm[i][j] <= 316) {r = 8; Ttable = 314;}
    else if (Tm[i][j] <= 318) {r = 9; Ttable = 316;}
    else if (Tm[i][j] <= 320) {r = 10; Ttable = 318;}
    else if (Tm[i][j] <= 322) {r = 11; Ttable = 320;}
    else if (Tm[i][j] <= 324) {r = 12; Ttable = 322;}
    else if (Tm[i][j] <= 326) {r = 13; Ttable = 324;}
    else if (Tm[i][j] <= 328) {r = 14; Ttable = 326;}
    else if (Tm[i][j] <= 330) {r = 15; Ttable = 328;}
    else if (Tm[i][j] <= 332) {r = 16; Ttable = 330;}
    else if (Tm[i][j] <= 334) {r = 17; Ttable = 332;}
    else if (Tm[i][j] <= 336) {r = 18; Ttable = 334;}
    else if (Tm[i][j] <= 338) {r = 19; Ttable = 336;}
    else if (Tm[i][j] <= 340) {r = 20; Ttable = 338;}
    else if (Tm[i][j] <= 342) {r = 21; Ttable = 340;}
    else if (Tm[i][j] <= 344) {r = 22; Ttable = 342;}
    //If a temperature ever exceeds those in the thermodynamic property table
    //Because exceeding means bulk 2 phase flow
    else {r = 23; Ttable = 344;}//Artificially force the temperature to be the saturation
temperature
    //While this high temperature forcing is incorrect; it is the best option right now
with strictly one phase
    //Interpolate thermodynamic properties based upon the temperature
    denf[i][j] = ((propArray[r+1][1]-propArray[r][1])/(2))*(Tm[i][j] -
Ttable)+propArray[r][1];
    hf[i][j] = ((propArray[r+1][2]-propArray[r][2])/(2))*(Tm[i][j] -
Ttable)+propArray[r][2];
    uf[i][j] = ((propArray[r+1][3]-propArray[r][3])/(2))*(Tm[i][j] -
Ttable)+propArray[r][3];
    kf[i][j] = ((propArray[r+1][4]-propArray[r][4])/(2))*(Tm[i][j] -
Ttable)+propArray[r][4];
    Prf[i][j] = ((propArray[r+1][5]-propArray[r][5])/(2))*(Tm[i][j] -
Ttable)+propArray[r][5];
    vf[i][j] = 1/denf[i][j];
    Relo[i][j] = G*Dh/uf[i][j]; //Liquid only Reynolds number
}

void computeTave (void) {
    double hout[65];
    double Tout[65];
    double averageOutlet = 0;
    double deltaT = 0;
    double epsilon = 0.000001;
    maxOutletTemp = 0;
    for (j=0;j<65;j++){
        hout[j] = hz[95][j] + qtop[95][j]*L/(96 * md);
        if (hout[j] <= propArray[1][2]){r = 0;}
        else if (hout[j] <= propArray[2][2]) {r = 1;}
        else if (hout[j] <= propArray[3][2]) {r = 2;}
        else if (hout[j] <= propArray[4][2]) {r = 3;}
        else if (hout[j] <= propArray[5][2]) {r = 4;}
        else if (hout[j] <= propArray[6][2]) {r = 5;}
        else if (hout[j] <= propArray[7][2]) {r = 6;}
        else if (hout[j] <= propArray[8][2]) {r = 7;}
        else if (hout[j] <= propArray[9][2]) {r = 8;}
        else if (hout[j] <= propArray[10][2]) {r = 9;}
        else if (hout[j] <= propArray[11][2]) {r = 10;}
        else if (hout[j] <= propArray[12][2]) {r = 11;}
        else if (hout[j] <= propArray[13][2]) {r = 12;}
        else if (hout[j] <= propArray[14][2]) {r = 13;}
        else if (hout[j] <= propArray[15][2]) {r = 14;}
        else if (hout[j] <= propArray[16][2]) {r = 15;}
        else if (hout[j] <= propArray[17][2]) {r = 16;}
        else if (hout[j] <= propArray[18][2]) {r = 17;}
        else if (hout[j] <= propArray[19][2]) {r = 18;}
        else if (hout[j] <= propArray[20][2]) {r = 19;}
        else if (hout[j] <= propArray[21][2]) {r = 20;}
        else if (hout[j] <= propArray[22][2]) {r = 21;}
        else if (hout[j] <= propArray[23][2]) {r = 22;}
        Tout[j] = ((propArray[r+1][0]-propArray[r][0])/(propArray[r+1][2] - 
propArray[r][2]))*(hz[95][j] - propArray[r][2]) + propArray[r][0];
        if(Tout[j]>maxOutletTemp){maxOutletTemp=Tout[j];}}
    for(i=0;i<65;i++){averageOutlet += 2*Tout[i]/121;}
    averageOutlet -= (Tout[1] + Tout[14] + Tout[26] + Tout[37] + Tout[45] + Tout[52] + 
Tout[58] + Tout[63] + Tout[65])/121; //Want diagonal elements to be not doubled
    deltaT = abs(averageOutlet - Texit);
    if (deltaT < epsilon) {findAxialOffset();}
        else{
            md *= (averageOutlet-Tmin)/(Texit-Tmin);
            G = md/Area;
            computeTm();} //Restart obtaining the temperatures again if too hot or cold
}

void findAxialOffset (void) {
    double botPower = 0;
    double topPower = 0;
    double maxValue = 0;
    double avgValue = 0;
    for (i=0;i<96;i++){
        axialPower[i]=0;
        for (j=0;j<121;j++){axialPower[i] += powerArray[i][j];}}
    for (i=0;i<48;i++){botPower += axialPower[i];}
    for (i=48;i<96;i++){topPower += axialPower[i];}
    axialOffset = 100*(topPower-botPower)/(topPower+botPower); // As a %
    for (i=0;i<96;i++){
        avgValue += axialPower[i]/96;
        if (axialPower[i] > maxValue){maxValue = axialPower[i];}}
    Fz = maxValue / avgValue;
    computeTco();
}

void computeTco (void){  //Outer cladding temperature (no subcooled boiling)
    double htc = 0;
    for (i=0;i<96;i++){
        for (j=0;j<65;j++){
            htc = 0.023*psi*pow(Relo[i][j],0.8)*pow(Prf[i][j],0.333)*kf[i][j]/Dh;
            Tco[i][j] = Tm[i][j] + qprime[i][j]/(pi*Dco*htc);}}
    computeTci();
}

void computeTci (void){  //Inner cladding temperature
    double kc = 0; // Cladding thermal conductivity
    for (i=0;i<96;i++){
        for (j=0;j<65;j++){
            kc = 10.318 + 0.016003*Tco[i][j]; //Assume cladding is at Tco temp - 
conservative for APMT
            Tci[i][j] = Tco[i][j] + qprime[i][j]*log(Rco/Rci)/(2*pi*kc);}}
    computeTfo();
}

void computeTfo (void){  //Outer fuel temperature
    double htcg = 0;
    double Tgap = 0;
    double kgas = 0;
    double delta = 0;
    double epsilon = 0;
    double sigma = 0;
    double Tfok = 0;
    double Tcik = 0;
    double Tfo2 = 0;
    sigma = 5.67037321*pow(10,-8); //Stefan-Boltzmann constant
    delta = Rci - Rfo;
    for (j=0;j<65;j++){Tfo[0][j]=500;} //Initial guess on Tfo at bottom
    for (i=0;i<96;i++){
        for (j=0;j<65;j++){
            if (i > 0) {Tfo[i][j] = Tfo[i-1][j];}  //The initial guess at Tfo - the
element below it
            do{
                Tfok = Tfo[i][j] + 273.15; //Tfo in Kelvin
                Tcik = Tci[i][j] + 273.15; //Tci in Kelvin
                Tgap = (Tfok + Tcik)/2; //Simple average of the two temperatures
                kgas = 0.00158*pow(Tgap,0.79); // Conductivity of the gas gap
                htcg = kgas/delta + sigma*(pow(Tfok,4) - pow(Tcik,4))/(Tfok-Tcik);
                Tfo2 = Tci[i][j] + qprime[i][j]/(pi*(Rci+Rfo)*htcg);
                epsilon = abs(Tfo2 - Tfo[i][j]);
                Tfo[i][j] = Tfo2;
                } while (epsilon > 0.000001);}}
    computeTmax();
}

void computeTmax (void){  //Maximum temperature (middle of the fuel)
    double Tmax1 = 0;
    double Tmax2 = 0;
    double kdTmin = 0;
    double kdTmax1 = 0;
    double kdTmax2 = 0;
    int rf = 100;  //Relaxation factor
    double epsilon = 1;
    //Using the conservative lower estimate from:
    //Integral Inherently Safe LWR (I2S-LWR) project Material property database
    //k(T in C) = 7.98 + 0.0051T
    for (i=0;i<96;i++){
        for (j=0;j<65;j++){
            kdTmin = 7.98*Tfo[i][j] + 0.00255*(Tfo[i][j])*(Tfo[i][j]);
            kdTmax1 = kdTmin + qprime[i][j]/(4*pi);
            Tmax1 = Tfo[i][j] + qbot[i][j]/(4*pi*8); //An initial guess at Tmax using
constant fuel conductivity of 8 W/(m*K)
            epsilon = 1; //Needs to be reset above the threshold each time
            do{
                kdTmax2 = 7.98*Tmax1 + 0.00255*(Tmax1)*(Tmax1);
                Tmax2 = Tmax1 + (kdTmax1 - kdTmax2)/rf;
                epsilon = abs(Tmax2 - Tmax1);
                Tmax1 = Tmax2;
            } while (epsilon > 0.000001);
            Tmax[i][j] = Tmax2;
            if (Tmax[i][j]>maxFuelTemp){maxFuelTemp = Tmax[i][j];}}}
    axialCollapse();
}

void axialCollapse (void) {
    avgCladTemp = 0;
    avgGapTemp = 0;
    avgFuelTemp = 0;
    avgCenterline = 0;
    for (i=0;i<96;i++){
        for (j=0;j<65;j++){
            avgCladTemp += 2*0.5*(Tci[i][j] + Tco[i][j])/(121*96);
            avgGapTemp += 2*0.5*(Tci[i][j] + Tfo[i][j])/(121*96);
            avgFuelTemp += 2*0.5*(Tfo[i][j] + Tmax[i][j])/(121*96);
            avgCenterline += 2*Tmax[i][j]/(121*96);}
        avgCladTemp -= 0.5*(Tci[i][1] + Tci[i][14] + Tci[i][26] + Tci[i][37] + Tci[i][45]
+ Tci[i][52] + Tci[i][58] + Tci[i][63] + Tci[i][65] + Tco[i][1] + Tco[i][14] + Tco[i][26]
+ Tco[i][37] + Tco[i][45] + Tco[i][52] + Tco[i][58] + Tco[i][63] + Tco[i][65])/(121*96);
        avgGapTemp -= 0.5*(Tci[i][1] + Tci[i][14] + Tci[i][26] + Tci[i][37] + Tci[i][45]
+ Tci[i][52] + Tci[i][58] + Tci[i][63] + Tci[i][65] + Tfo[i][1] + Tfo[i][14] + Tfo[i][26]
+ Tfo[i][37] + Tfo[i][45] + Tfo[i][52] + Tfo[i][58] + Tfo[i][63] + Tfo[i][65])/(121*96);
        avgFuelTemp -= 0.5*(Tmax[i][1] + Tmax[i][14] + Tmax[i][26] + Tmax[i][37] +
Tmax[i][45] + Tmax[i][52] + Tmax[i][58] + Tmax[i][63] + Tmax[i][65] + Tfo[i][1] +
Tfo[i][14] + Tfo[i][26] + Tfo[i][37] + Tfo[i][45] + Tfo[i][52] + Tfo[i][58] + Tfo[i][63]
+ Tfo[i][65])/(121*96);
        avgCenterline -= 0.5*(Tmax[i][1] + Tmax[i][14] + Tmax[i][26] + Tmax[i][37] +
Tmax[i][45] + Tmax[i][52] + Tmax[i][58] + Tmax[i][63] + Tmax[i][65])/(121*96);}
    for (i=0;i<10;i++){
        for (j=0;j<65;j++){
            TmaxC[i][j] = 0;
            TfoC[i][j] = 0;
            TciC[i][j] = 0;
            TcoC[i][j] = 0;
            TmC[i][j] = 0;
            TfaveC[i][j] = 0;
            denfC[i][j] = 0;}}
    for (j=0;j<65;j++){
        countr = 0;
        r = 0;
        for (i=0;i<10;i++){
            r += axialUnits[i];
            for (k=countr;k<r;k++){
                TmaxC[i][j] += Tmax[k][j]/axialUnits[i];
                TfoC[i][j] += Tfo[k][j]/axialUnits[i];
                TciC[i][j] += Tci[k][j]/axialUnits[i];
                TcoC[i][j] += TcoC[k][j]/axialUnits[i];
                TmC[i][j] += Tm[k][j]/axialUnits[i];
                denfC[i][j] += denf[k][j]/(1000*axialUnits[i]);} // convert from kg/m3 to
g/cc
            TfaveC[i][j] = 0.5*(TmaxC[i][j] + TfoC[i][j]);
            countr = r;}}
    //Assume radial reflector coolant temp and density is equal to the average in each
axial section
    for (i=0;i<10;i++){
        TSliceAve[i] = 0;
        denSliceAve[i] = 0;
        for (j=0;j<65;j++){
            TSliceAve[i] += 2*TmC[i][j]/121;
            denSliceAve[i] += 2*denfC[i][j]/121;} //convert from kg/m3 to g/cc
        TSliceAve[i] -= (TmC[i][1] + TmC[i][14] + TmC[i][26] + TmC[i][37] + TmC[i][45] +
TmC[i][52] + TmC[i][58] + TmC[i][63] + TmC[i][65])/121;
        denSliceAve[i] -= (denfC[i][1] + denfC[i][14] + denfC[i][26] + denfC[i][37] +
denfC[i][45] + denfC[i][52] + denfC[i][58] + denfC[i][63] + denfC[i][65])/121;}
    //Convert all temps to K
    for (i=0;i<10;i++){
        for(j=0;j<65;j++){
            TmaxC[i][j] += 273.15;
            TfoC[i][j] += 273.15;
            TciC[i][j] += 273.15;
            TcoC[i][j] += 273.15;
            TmC[i][j] += 273.15;
            TfaveC[i][j] += 273.15;}
        TSliceAve[i] += 273.15;}
    double maxValue = 0;
    double avgValue = 0;
    for (i=0;i<121;i++){
        radialPower[i] = 0;
        for (j=0;j<96;j++){radialPower[i] += reducedArray[j][i]/96;}}
    for (i=0;i<121;i++){
        avgValue += radialPower[i]/121;
        if (radialPower[i] > maxValue){maxValue = radialPower[i];}}
    FdH = maxValue / avgValue;
    coolantMaker();
}

void coolantMaker (void){
    for (i = 0; i < 10; i++){
        for (j=0;j<65;j++){
            denCoolantC[i][j] = 0.006*7.25 + denfC[i][j]*0.994;
            atomDHCoolant[i][j] = ((1000000-
CBC)/1000000)*denfC[i][j]*2*(6.022*pow(10,23))*0.994/18.016;
            atomDOCoolant[i][j] = atomDHCoolant[i][j]/2;
            atomB10Coolant[i][j] =
denfC[i][j]*(CBC/1000000)*0.199*(6.022*pow(10,23))*0.994/10.811;
```

```
            atomB11Coolant[i][j] =
denfC[i][j]*(CBC/1000000)*0.801*(6.022*pow(10,23))*0.994/10.811;}
        denRefC[i] = 0.9*7.25 + 0.1*denSliceAve[i];
        atomDHRef[i] = ((1000000-
CBC)/1000000)*denSliceAve[i]*2*(6.022*pow(10,23))*0.1/18.016;
        atomDORef[i] = atomDHRef[i]/2;
        atomB10Ref[i] = denSliceAve[i]*(CBC/1000000)*0.199*(6.022*pow(10,23))*0.1/10.811;
        atomB11Ref[i] = denSliceAve[i]*(CBC/1000000)*0.801*(6.022*pow(10,23))*0.1/10.811;
        denInter[i] = 0.006*7.25 + denSliceAve[i]*0.994;
        atomDHInter[i] = ((1000000-
CBC)/1000000)*denSliceAve[i]*2*(6.022*pow(10,23))*0.994/18.016;
        atomDOInter[i] = atomDHInter[i]/2;
        atomB10Inter[i] =
denSliceAve[i]*(CBC/1000000)*0.199*(6.022*pow(10,23))*0.994/10.811;
        atomB11Inter[i] =
denSliceAve[i]*(CBC/1000000)*0.801*(6.022*pow(10,23))*0.994/10.811;}
    atomDHBARef = ((1000000-CBC)/1000000)*0.73085*2*(6.022*pow(10,23))*0.7/18.016;
    atomDOBARef = atomDHBARef/2;
    atomB10BARef = 0.73085*(CBC/1000000)*0.199*(6.022*pow(10,23))*0.7/10.811;
    atomB11BARef = 0.73085*(CBC/1000000)*0.801*(6.022*pow(10,23))*0.7/10.811;
    atomDHTARef = ((1000000-CBC)/1000000)*0.64965*2*(6.022*pow(10,23))*0.7/18.016;
    atomDOTARef = atomDHTARef/2;
    atomB10TARef = 0.64965*(CBC/1000000)*0.199*(6.022*pow(10,23))*0.7/10.811;
    atomB11TARef = 0.64965*(CBC/1000000)*0.801*(6.022*pow(10,23))*0.7/10.811;
    writeOut();
}

void writeOut (void) {  //Writes all the calculated values to the user-specified file
    double totalMFR = 0;
    cout << "Axial Offset:\t\t\t\t" << setprecision(4) << axialOffset << " %\n";
    cout << "Fq (Global Peaking Factor):\t\t" << setprecision(4) << Fq << "\n";
    cout << "Fz (Axial Peaking Factor):\t\t" << setprecision(4) << Fz << "\n";
    cout << "FdH (Radial Peaking Factor):\t\t" << setprecision(4) << FdH << "\n";
    totalMFR = md*121*336;
    cout << "Mass Flow Rate:\t\t\t\t" << setprecision(5) << totalMFR << " kg/s\n";
    cout << "Maximum Channel Outlet Temperature:\t" << setprecision(4) << maxOutletTemp
<< " C\n";
    cout << "Maximum Fuel Temperature:\t\t" << setprecision(4) << maxFuelTemp << " C\n";
    cout << "Average Centerline Fuel Temperature:\t" << setprecision(4) << avgCenterline
<< " C\n";
    cout << "Average Fuel Temperature:\t\t" << setprecision(4) << avgFuelTemp << " C\n";
    cout << "Average Cladding Temperature:\t\t" << setprecision(4) << avgCladTemp << "
C\n";
    avgCladTemp += 273.15;
    cout << "Average Gas Gap Temperature:\t\t" << setprecision(4) << avgGapTemp << "
C\n";
    avgGapTemp += 273.15;
    out_file.open(filenameOut);
    out_file << endl;
    out_file << "/*\n";
    out_file << "Axial Offset:\t" << setprecision(4) << axialOffset << " %\n";
    out_file << "Fq (Global Peaking Factor):\t" << setprecision(4) << Fq << "\n";
    out_file << "Fz (Axial Peaking Factor):\t" << setprecision(4) << Fz << "\n";
    out_file << "FdH (Radial Peaking Factor):\t" << setprecision(4) << FdH <<"\n";
    out_file << "Mass Flow Rate:\t" << setprecision(5) << totalMFR << "\tkg/s\n";
    out_file << "Maximum Channel Outlet Temperature:\t" << setprecision(4) <<
maxOutletTemp << "\tC\n";
    out_file << "Maximum Fuel Temperature:\t" << setprecision(4) << maxFuelTemp <<
"\tC\n";
    out_file << "Average Centerline Fuel Temperature:\t" << setprecision(4) <<
avgCenterline << "\tC\n";
    out_file << "Average Fuel Temperature:\t" << setprecision(4) << avgFuelTemp <<
"\tC\n";
    out_file << "Average Cladding Temperature:\t" << setprecision(4) << avgCladTemp <<
"\tC\n";
    out_file << "Average Gas Gap Temperature:\t" << setprecision(4) << avgGapTemp <<
"\tC\n";
    out_file << "*/\n\n";
    //IFBA materials
    for (i=1;i<8;i++){ //Since IFBA only in seven slices
        k = 0;
        for (j=1;j<60;j++){
            if (j == (IFBAAssemblies[k]-1)){
                out_file << "mat IFBA" << i;
                if (IFBAAssemblies[k] < 10) {out_file << "0";}
                out_file << IFBAAssemblies[k] << " -6.08 tms " << setprecision(3) <<
TfoC[i][j] << " burn 1\n";
                out_file << "5010.06c  1.2\n";
                out_file << "5011.06c  0.8\n";
                out_file << "40000.06c 1\n\n";
                k++;}}}
    //High Fuel materials
    for (i=1;i<9;i++){
        k=0;
        for (j=11;j<65;j++){
            if (j == (HEnrich[k]-1)){
                out_file << "mat Fuel1" << i;
                if (HEnrich[k] < 10) {out_file << "0";}
                out_file << HEnrich[k]<< " -11.7161 tms " << setprecision(3) <<
TfaveC[i][j] << " burn 1\n";
                out_file << "14000.06c -0.864303\n";
                out_file << "92234.06c -0.004341\n";
                out_file << "92235.06c -0.434072 % Enrichment: 4.0 w%\n";
                out_file << "92238.06c -10.413384\n\n";
                for (r=0;r<26;r++){
                    if ((j == (IFBAAssemblies[r]-1)) && (i<8)){
                        out_file << "mat Fuel4" << i;
                        if (IFBAAssemblies[r] < 10) {out_file << "0";}
                        out_file << IFBAAssemblies[r]<< " -11.7161 tms "<<
setprecision(3) << TfaveC[i][j] << " burn 1\n";
                        out_file << "14000.06c -0.864303\n";
                        out_file << "92234.06c -0.004341\n";
                        out_file << "92235.06c -0.434072 % Enrichment: 4.0 w%\n";
                        out_file << "92238.06c -10.413384\n\n";}}
                k++;}}}
    //Medium Fuel materials
    for (i=1;i<9;i++){
        k=0;
        for (j=5;j<47;j++){
            if (j == (MEnrich[k]-1)){
                out_file << "mat Fuel2" << i;
                if (MEnrich[k] < 10) {out_file << "0";}
                out_file << MEnrich[k]<< " -11.7161 tms " << setprecision(3) <<
TfaveC[i][j] << " burn 1\n";
                out_file << "14000.06c -0.864303\n";
                out_file << "92234.06c -0.004341\n";
                out_file << "92235.06c -0.368961 % Enrichment: 3.4 w%\n";
                out_file << "92238.06c -10.478495\n\n";
                for (r=0;r<26;r++){
                    if ((j == (IFBAAssemblies[r]-1)) && (i<8)){
                        out_file << "mat Fuel5" << i;
                        if (IFBAAssemblies[r] < 10) {out_file << "0";}
                        out_file << IFBAAssemblies[r]<< " -11.7161 tms " <<
setprecision(3) << TfaveC[i][j] << " burn 1\n";
                        out_file << "14000.06c -0.864303\n";
                        out_file << "92234.06c -0.004341\n";
                        out_file << "92235.06c -0.368961 % Enrichment: 3.4 w%\n";
                        out_file << "92238.06c -10.478495\n\n";}}
                k++;}}}
    //Low Fuel materials
    for (i=1;i<9;i++){
        k=0;
        for (j=0;j<58;j++){
            if (j == (LEnrich[k]-1)){
                out_file << "mat Fuel3" << i;
                if (LEnrich[k] < 10) {out_file << "0";}
                out_file << LEnrich[k] << " -11.7161 tms " << setprecision(3) <<
TfaveC[i][j] << " burn 1\n";
                out_file << "14000.06c -0.864303\n";
                out_file << "92234.06c -0.004341\n";
                out_file << "92235.06c -0.309276 % Enrichment: 2.85 w%\n";
                out_file << "92238.06c -10.538180\n\n";
                for (r=0;r<26;r++){
                    if ((j == (IFBAAssemblies[r]-1)) && (i<8)){
                        out_file << "mat Fuel6" << i;
                        if (IFBAAssemblies[r] < 10) {out_file << "0";}
                        out_file << IFBAAssemblies[r] << " -11.7161 tms " <<
setprecision(3) << TfaveC[i][j] << " burn 1\n";
                        out_file << "14000.06c -0.864303\n";
                        out_file << "92234.06c -0.004341\n";
                        out_file << "92235.06c -0.309276 % Enrichment: 2.85 w%\n";
                        out_file << "92238.06c -10.538180\n\n";}}
                k++;}}}
    //Blanket Fuel materials
    for (j=0;j<65;j++){
            out_file << "mat Blanket0";
            if (j < 9) {out_file << "0";}
            out_file << (j+1) << " -11.7161 tms " << setprecision(3) << TfaveC[0][j] << "
burn 1\n";
            out_file << "14000.06c -0.864303\n";
            out_file << "92234.06c -0.004341\n";
            out_file << "92235.06c -0.271295 % Enrichment: 2.5 w%\n";
            out_file << "92238.06c -10.576161\n\n";}
    for (j=0;j<65;j++){
            out_file << "mat Blanket9";
            if (j < 9) {out_file << "0";}
            out_file << (j+1) << " -11.7161 tms " << setprecision(3) << TfaveC[9][j] << "
burn 1\n";
            out_file << "14000.06c -0.864303\n";
            out_file << "92234.06c -0.004341\n";
            out_file << "92235.06c -0.271295 % Enrichment: 2.5 w%\n";
            out_file << "92238.06c -10.576161\n\n";}
    //Channel Coolant materials
    for (i=0;i<10;i++){
        for (j=0;j<65;j++){
            out_file << "% Density of water: " << setprecision(5) << denfC[i][j] << "
g/cc\n";
            out_file << "mat water" << i;
            if (j<9) {out_file << "0";}
            out_file << (j+1) << " -" << setprecision(5) << denCoolantC[i][j] << " moder
lwtr 1001 tmp " << setprecision(3) << TmC[i][j] << endl;
            out_file << "24000.03c 1.05798e+020 % Cr\n";
            out_file << "13027.03c 4.85437e+019 % Al\n";
            out_file << "42000.03c 8.19042e+018 % Mo\n";
            out_file << "6000.03c  1.74478e+018 % C\n";
            out_file << "14000.03c 6.52910e+018 % Si\n";
            out_file << "25055.03c 1.90729e+018 % Mn\n";
            out_file << "26000.03c 3.27511e+020 % Fe\n";
            out_file << "1001.03c  " << setprecision(6) << atomDHCoolant[i][j] << " %
H\n";
            out_file << "8016.03c  " << setprecision(6) << atomDOCoolant[i][j] << " %
O\n";
            out_file << "5010.03c  " << setprecision(6) << atomB10Coolant[i][j] << " %
B10\n";
            out_file << "5011.03c  " << setprecision(6) << atomB11Coolant[i][j] << " %
B11\n\n";}}
    //Interassembly Coolant materials
    for (i=0;i<10;i++){
        out_file << "% Density of water: " << setprecision(5) << denSliceAve[i] << "
g/cc\n";
        out_file << "mat water" << i << " -" << setprecision(5) << denInter[i] << " moder
lwtr 1001 tmp " << setprecision(3) << TSliceAve[i] << endl;
        out_file << "24000.03c 1.05798e+020 % Cr\n";
        out_file << "13027.03c 4.85437e+019 % Al\n";
        out_file << "42000.03c 8.19042e+018 % Mo\n";
        out_file << "6000.03c  1.74478e+018 % C\n";
        out_file << "14000.03c 6.52910e+018 % Si\n";
        out_file << "25055.03c 1.90729e+018 % Mn\n";
        out_file << "26000.03c 3.27511e+020 % Fe\n";
        out_file << "1001.03c  " << setprecision(6) << atomDHInter[i] << " % H\n";
        out_file << "8016.03c  " << setprecision(6) << atomDOInter[i] << " % O\n";
        out_file << "5010.03c  " << setprecision(6) << atomB10Inter[i] << " % B10\n";
        out_file << "5011.03c  " << setprecision(6) << atomB11Inter[i] << " % B11\n\n";}
    //Radial Reflector materials
    for (i=0;i<10;i++){
        out_file << "% Density of water: " << setprecision(5) << denSliceAve[i] << "
g/cc\n";
        out_file << "mat RadRef" << i;
        out_file << " -" << setprecision(5) << denRefC[i] << " moder lwtr 1001 tmp " <<
setprecision(3) << TSliceAve[i] << endl;
        out_file << "24000.03c 1.58697e+022 % Cr\n";
        out_file << "13027.03c 7.28156e+021 % Al\n";
        out_file << "42000.03c 1.22856e+021 % Mo\n";
        out_file << "6000.03c  2.61717e+020 % C\n";
        out_file << "14000.03c 9.79366e+020 % Si\n";
        out_file << "25055.03c 2.86094e+020 % Mn\n";
        out_file << "26000.03c 4.91266e+022 % Fe\n";
        out_file << "1001.03c  " << setprecision(6) << atomDHRef[i] << " % H\n";
        out_file << "8016.03c  " << setprecision(6) << atomDORef[i] << " % O\n";
        out_file << "5010.03c  " << setprecision(6) << atomB10Ref[i] << " % B10\n";
        out_file << "5011.03c  " << setprecision(6) << atomB11Ref[i] << " % B11\n\n";}
    //Bottom Axial Reflector material
    out_file << "mat AxReflB -2.6866 moder lwtr 1001 tmp 571.15\n";
    out_file << "24000.03c 5.28991e+021 % Cr\n";
    out_file << "13027.03c 2.42719e+021 % Al\n";
    out_file << "42000.03c 4.09521e+020 % Mo\n";
    out_file << "6000.03c  8.72390e+019 % C\n";
    out_file << "14000.03c 3.26455e+020 % Si\n";
    out_file << "25055.03c 9.53645e+019 % Mn\n";
    out_file << "26000.03c 1.63755e+022 % Fe\n";
    out_file << "1001.03c  " << setprecision(6) << atomDHBARef << " % H\n";
    out_file << "8016.03c  " << setprecision(6) << atomDOBARef << " % O\n";
    out_file << "5010.03c  " << setprecision(6) << atomB10BARef << " % B10\n";
    out_file << "5011.03c  " << setprecision(6) << atomB11BARef  << " % B11\n\n";
    //Top Axial Reflector material
    out_file << "mat AxReflT -2.62976 moder lwtr 1001 tmp 603.65\n";
    out_file << "24000.03c 5.28991e+021 % Cr\n";
    out_file << "13027.03c 2.42719e+021 % Al\n";
    out_file << "42000.03c 4.09521e+020 % Mo\n";
    out_file << "6000.03c  8.72390e+019 % C\n";
    out_file << "14000.03c 3.26455e+020 % Si\n";
    out_file << "25055.03c 9.53645e+019 % Mn\n";
    out_file << "26000.03c 1.63755e+022 % Fe\n";
    out_file << "1001.03c  " << setprecision(6) << atomDHTARef << " % H\n";
    out_file << "8016.03c  " << setprecision(6) << atomDOTARef << " % O\n";
    out_file << "5010.03c  " << setprecision(6) << atomB10TARef << " % B10\n";
    out_file << "5011.03c  " << setprecision(6) << atomB11TARef  << " % B11\n\n";
    //End of output file
    out_file.close();
    cout << "\nThe file " << filenameOut << " has been written with the calculated
values.\n";
    cout << "You can rerun the code for another set of files.\n";
    cout << "Click the X at the upper right to exit when finished.\n\n";
}
```

APPENDIX B SAMPLE SERPENT CORE PHYSICS INPUT PARAMETERS

A sample of the core physics parameters used for all the depletion simulations is included. Geometry and materials definitions used in the input files are excluded from this thesis because they are very lengthy and have rather periodic definitions since each axial slice is defined explicitly for depletion tracking purposes. Note the overview of geometry and materials in Chapter 5 for listings of geometric and materials parameters, or see the note at the end of this appendix if further information is requested.

This example case was specifically used to deplete from 6 to 7 MWd/kg, but all inputs followed this same format. The nuclides listed in the "Nuclide inventory" are not all the nuclides tracked by Serpent during the depletion. The ones listed are just ones requested by the user to be included with the output. Serpent actually tracked 2210 nuclides (1356 for transport calculations and an additional 854 for decaying only) for this study for all materials used, fission products, decay chain members, and all the relevant excited states.

```
% --- Thermal scattering data for light water:
therm lwtr lwj3.11t

% --- Cross section data library file path:
set acelib "/home/ramey/xslibs/endfb7/acedata/endfb7_b3.xsdata"

% --- Optimization Mode (4-Best Optimization 1-Worst)
set opti 4

% --- Decay and fission yield libraries:
set declib "/home/ramey/xslibs/endfb7/acedata/sss_endfb7.dec"
set nfylib "/home/ramey/xslibs/endfb7/acedata/sss_endfb7.nfy"

% universe = 0 (homogenization over all space)
set gcu  0

% --- Set Boundary Condition
set bc 2       % 1 Black, 2 Reflect, 3 Periodic

% --- Neutron population and criticality cycles:
set pop 210000 500 100               % Paricles per generation (Active / Inactive)
```

```
% --- Geometry and mesh plots:
%plot 3 8000 8000 [40 -122.5 122.5 -122.5 122.5]
%mesh 3 5000 5000 %Reaction Rate Plotter

% --- Reduce energy grid size:
set egrid 5E-5 1E-9 15.0

% --- Total fission energy deposition in 3D mesh
det 1  % Radial profile
dr -8 void
dx -122.5 27.65845 13
dy -27.65845 122.5 13
dz 30.48 396.24 1

det 2  % Axial Profile
dr -8 void
dx -122.5 27.65845 1
dy -27.65845 122.5 1
dz 30.48 396.24 96

det 3 % For more detailed studies - each quarter-assembly location in 3in axial slices
dr -8 void
dx -122.5 27.65845 13
dy -27.65845 122.5 13
dz 30.48 396.24 96

% --- Total power for normalization:
set powdens 0.03425182        %  [kW/g]  (2850  MW  /  4  -  since  quarter  core  model)  /
20,801,813.54 g U

% --- Cut-offs:
set fpcut   1E-6
set stabcut 1E-12

% --- Options for burnup calculation:
set bumode  2  % CRAM (Chebyshev Rational Approximation Method) method
set pcc     1  % Predictor-corrector calculation on
set xscalc  2  % Cross sections from spectrum
set ures       0      %(1-Use 0-Omit) unresolved resonance table

set xenon 1 [Blanket001 Blanket002 Blanket003 Blanket004 Blanket005 Blanket006 Blanket007
Blanket008 Blanket009 Blanket010 Blanket011 Blanket012 Blanket013 Blanket014 Blanket015
Blanket016 Blanket017 Blanket018 Blanket019 Blanket020 Blanket021 Blanket022 Blanket023
Blanket024 Blanket025 Blanket026 Blanket027 Blanket028 Blanket029 Blanket030 Blanket031
Blanket032 Blanket033 Blanket034 Blanket035 Blanket036 Blanket037 Blanket038 Blanket039
Blanket040 Blanket041 Blanket042 Blanket043 Blanket044 Blanket045 Blanket046 Blanket047
Blanket048 Blanket049 Blanket050 Blanket051 Blanket052 Blanket053 Blanket054 Blanket055
Blanket056 Blanket057 Blanket058 Blanket059 Blanket060 Blanket061 Blanket062 Blanket063
Blanket064 Blanket065 Blanket901 Blanket902 Blanket903 Blanket904 Blanket905 Blanket906
Blanket907 Blanket908 Blanket909 Blanket910 Blanket911 Blanket912 Blanket913 Blanket914
Blanket915 Blanket916 Blanket917 Blanket918 Blanket919 Blanket920 Blanket921 Blanket922
Blanket923 Blanket924 Blanket925 Blanket926 Blanket927 Blanket928 Blanket929 Blanket930
Blanket931 Blanket932 Blanket933 Blanket934 Blanket935 Blanket936 Blanket937 Blanket938
Blanket939 Blanket940 Blanket941 Blanket942 Blanket943 Blanket944 Blanket945 Blanket946
Blanket947 Blanket948 Blanket949 Blanket950 Blanket951 Blanket952 Blanket953 Blanket954
Blanket955 Blanket956 Blanket957 Blanket958 Blanket959 Blanket960 Blanket961 Blanket962
Blanket963 Blanket964 Blanket965 Fuel1112 Fuel1113 Fuel1124 Fuel1125 Fuel1135 Fuel1136
Fuel1143 Fuel1144 Fuel1150 Fuel1151 Fuel1154 Fuel1155 Fuel1156 Fuel1157 Fuel1159 Fuel1160
Fuel1161 Fuel1162 Fuel1163 Fuel1164 Fuel1165 Fuel4143 Fuel4144 Fuel4150 Fuel4151 Fuel4154
Fuel4155 Fuel4159 Fuel4160 Fuel1212 Fuel1213 Fuel1224 Fuel1225 Fuel1235 Fuel1236 Fuel1243
Fuel1244 Fuel1250 Fuel1251 Fuel1254 Fuel1255 Fuel1256 Fuel1257 Fuel1259 Fuel1260 Fuel1261
Fuel1262 Fuel1263 Fuel1264 Fuel1265 Fuel4243 Fuel4244 Fuel4250 Fuel4251 Fuel4254 Fuel4255
Fuel4259 Fuel4260 Fuel1312 Fuel1313 Fuel1324 Fuel1325 Fuel1335 Fuel1336 Fuel1343 Fuel1344
Fuel1350 Fuel1351 Fuel1354 Fuel1355 Fuel1356 Fuel1357 Fuel1359 Fuel1360 Fuel1361 Fuel1362
Fuel1363 Fuel1364 Fuel1365 Fuel4343 Fuel4344 Fuel4350 Fuel4351 Fuel4354 Fuel4355 Fuel4359
Fuel4360 Fuel1412 Fuel1413 Fuel1424 Fuel1425 Fuel1435 Fuel1436 Fuel1443 Fuel1444 Fuel1450
Fuel1451 Fuel1454 Fuel1455 Fuel1456 Fuel1457 Fuel1459 Fuel1460 Fuel1461 Fuel1462 Fuel1463
Fuel1464 Fuel1465 Fuel4443 Fuel4444 Fuel4450 Fuel4451 Fuel4454 Fuel4455 Fuel4459 Fuel4460
Fuel1512 Fuel1513 Fuel1524 Fuel1525 Fuel1535 Fuel1536 Fuel1543 Fuel1544 Fuel1550 Fuel1551
Fuel1554 Fuel1555 Fuel1556 Fuel1557 Fuel1559 Fuel1560 Fuel1561 Fuel1562 Fuel1563 Fuel1564
Fuel1565 Fuel4543 Fuel4544 Fuel4550 Fuel4551 Fuel4554 Fuel4555 Fuel4559 Fuel4560 Fuel1612
Fuel1613 Fuel1624 Fuel1625 Fuel1635 Fuel1636 Fuel1643 Fuel1644 Fuel1650 Fuel1651 Fuel1654
Fuel1655 Fuel1656 Fuel1657 Fuel1659 Fuel1660 Fuel1661 Fuel1662 Fuel1663 Fuel1664 Fuel1665
Fuel4643 Fuel4644 Fuel4650 Fuel4651 Fuel4654 Fuel4655 Fuel4659 Fuel4660 Fuel1712 Fuel1713
Fuel1724 Fuel1725 Fuel1735 Fuel1736 Fuel1743 Fuel1744 Fuel1750 Fuel1751 Fuel1754 Fuel1755
Fuel1756 Fuel1757 Fuel1759 Fuel1760 Fuel1761 Fuel1762 Fuel1763 Fuel1764 Fuel1765 Fuel4743
Fuel4744 Fuel4750 Fuel4751 Fuel4754 Fuel4755 Fuel4759 Fuel4760 Fuel1812 Fuel1813 Fuel1824
Fuel1825 Fuel1835 Fuel1836 Fuel1843 Fuel1844 Fuel1850 Fuel1851 Fuel1854 Fuel1855 Fuel1856
Fuel1857 Fuel1859 Fuel1860 Fuel1861 Fuel1862 Fuel1863 Fuel1864 Fuel1865 Fuel2106 Fuel2107
Fuel2110 Fuel2111 Fuel2116 Fuel2117 Fuel2120 Fuel2121 Fuel2122 Fuel2123 Fuel2127 Fuel2128
Fuel2131 Fuel2132 Fuel2133 Fuel2134 Fuel2139 Fuel2140 Fuel2146 Fuel2147 Fuel5106 Fuel5107
Fuel5110 Fuel5111 Fuel5116 Fuel5117 Fuel5120 Fuel5121 Fuel5127 Fuel5128 Fuel5131 Fuel513
Fuel5139 Fuel5140 Fuel5146 Fuel5147 Fuel2206 Fuel2207 Fuel2210 Fuel2211 Fuel2216 Fuel2217
Fuel2220 Fuel2221 Fuel2222 Fuel2223 Fuel2227 Fuel2228 Fuel2231 Fuel2232 Fuel2233 Fuel2234
Fuel2239 Fuel2240 Fuel2246 Fuel2247 Fuel5206 Fuel5207 Fuel5210 Fuel5211 Fuel5216 Fuel5217
Fuel5220 Fuel5221 Fuel5227 Fuel5228 Fuel5231 Fuel523 Fuel5239 Fuel5240 Fuel5246 Fuel5247
Fuel2306 Fuel2307 Fuel2310 Fuel2311 Fuel2316 Fuel2317 Fuel2320 Fuel2321 Fuel2322 Fuel2323
Fuel2327 Fuel2328 Fuel2331 Fuel2332 Fuel2333 Fuel2334 Fuel2339 Fuel2340 Fuel2346 Fuel2347
Fuel5306 Fuel5307 Fuel5310 Fuel5311 Fuel5316 Fuel5317 Fuel5320 Fuel5321 Fuel5327 Fuel5328
Fuel5331 Fuel533 Fuel5339 Fuel5340 Fuel5346 Fuel5347 Fuel2406 Fuel2407 Fuel2410 Fuel2411
Fuel2416 Fuel2417 Fuel2420 Fuel2421 Fuel2422 Fuel2423 Fuel2427 Fuel2428 Fuel2431 Fuel2432
Fuel2433 Fuel2434 Fuel2439 Fuel2440 Fuel2446 Fuel2447 Fuel5406 Fuel5407 Fuel5410 Fuel5411
Fuel5416 Fuel5417 Fuel5420 Fuel5421 Fuel5427 Fuel5428 Fuel5431 Fuel543 Fuel5439 Fuel5440
Fuel5446 Fuel5447 Fuel2506 Fuel2507 Fuel2510 Fuel2511 Fuel2516 Fuel2517 Fuel2520 Fuel2521
Fuel2522 Fuel2523 Fuel2527 Fuel2528 Fuel2531 Fuel2532 Fuel2533 Fuel2534 Fuel2539 Fuel2540
Fuel2546 Fuel2547 Fuel5506 Fuel5507 Fuel5510 Fuel5511 Fuel5516 Fuel5517 Fuel5520 Fuel5521
Fuel5527 Fuel5528 Fuel5531 Fuel553 Fuel5539 Fuel5540 Fuel5546 Fuel5547 Fuel2606 Fuel2607
Fuel2610 Fuel2611 Fuel2616 Fuel2617 Fuel2620 Fuel2621 Fuel2622 Fuel2623 Fuel2627 Fuel2628
Fuel2631 Fuel2632 Fuel2633 Fuel2634 Fuel2639 Fuel2640 Fuel2646 Fuel2647 Fuel5606 Fuel5607
Fuel5610 Fuel5611 Fuel5616 Fuel5617 Fuel5620 Fuel5621 Fuel5627 Fuel5628 Fuel5631 Fuel563
Fuel5639 Fuel5640 Fuel5646 Fuel5647 Fuel2706 Fuel2707 Fuel2710 Fuel2711 Fuel2716 Fuel2717
Fuel2720 Fuel2721 Fuel2722 Fuel2723 Fuel2727 Fuel2728 Fuel2731 Fuel2732 Fuel2733 Fuel2734
Fuel2739 Fuel2740 Fuel2746 Fuel2747 Fuel5706 Fuel5707 Fuel5710 Fuel5711 Fuel5716 Fuel5717
Fuel5720 Fuel5721 Fuel5727 Fuel5728 Fuel5731 Fuel573 Fuel5739 Fuel5740 Fuel5746 Fuel5747
Fuel2806 Fuel2807 Fuel2810 Fuel2811 Fuel2816 Fuel2817 Fuel2820 Fuel2821 Fuel2822 Fuel2823
Fuel2827 Fuel2828 Fuel2831 Fuel2832 Fuel2833 Fuel2834 Fuel2839 Fuel2840 Fuel2846 Fuel2847
Fuel3102 Fuel3103 Fuel3104 Fuel3105 Fuel3108 Fuel3109 Fuel3114 Fuel3115 Fuel3118 Fuel3119
Fuel3126 Fuel3129 Fuel3130 Fuel3137 Fuel3138 Fuel3141 Fuel3142 Fuel3145 Fuel3148 Fuel3149
Fuel3152 Fuel3153 Fuel3158 Fuel6102 Fuel6103 Fuel3202 Fuel3203 Fuel3204 Fuel3205 Fuel3208
Fuel3209 Fuel3214 Fuel3215 Fuel3218 Fuel3219 Fuel3226 Fuel3229 Fuel3230 Fuel3237 Fuel3238
Fuel3241 Fuel3242 Fuel3245 Fuel3248 Fuel3249 Fuel3252 Fuel3253 Fuel3258 Fuel6202 Fuel6203
Fuel3302 Fuel3303 Fuel3304 Fuel3305 Fuel3308 Fuel3309 Fuel3314 Fuel3315 Fuel3318 Fuel3319
Fuel3326 Fuel3329 Fuel3330 Fuel3337 Fuel3338 Fuel3341 Fuel3342 Fuel3345 Fuel3348 Fuel3349
Fuel3352 Fuel3353 Fuel3358 Fuel6302 Fuel6303 Fuel3402 Fuel3403 Fuel3404 Fuel3405 Fuel3408
Fuel3409 Fuel3414 Fuel3415 Fuel3418 Fuel3419 Fuel3426 Fuel3429 Fuel3430 Fuel3437 Fuel3438
Fuel3441 Fuel3442 Fuel3445 Fuel3448 Fuel3449 Fuel3452 Fuel3453 Fuel3458 Fuel6402 Fuel6403
Fuel3502 Fuel3503 Fuel3504 Fuel3505 Fuel3508 Fuel3509 Fuel3514 Fuel3515 Fuel3518 Fuel3519
Fuel3526 Fuel3529 Fuel3530 Fuel3537 Fuel3538 Fuel3541 Fuel3542 Fuel3545 Fuel3548 Fuel3549
Fuel3552 Fuel3553 Fuel3558 Fuel6502 Fuel6503 Fuel3602 Fuel3603 Fuel3604 Fuel3605 Fuel3608
Fuel3609 Fuel3614 Fuel3615 Fuel3618 Fuel3619 Fuel3626 Fuel3629 Fuel3630 Fuel3637 Fuel3638
Fuel3641 Fuel3642 Fuel3645 Fuel3648 Fuel3649 Fuel3652 Fuel3653 Fuel3658 Fuel6602 Fuel6603
Fuel3702 Fuel3703 Fuel3704 Fuel3705 Fuel3708 Fuel3709 Fuel3714 Fuel3715 Fuel3718 Fuel3719
Fuel3726 Fuel3729 Fuel3730 Fuel3737 Fuel3738 Fuel3741 Fuel3742 Fuel3745 Fuel3748 Fuel3749
Fuel3752 Fuel3753 Fuel3758 Fuel6702 Fuel6703 Fuel3802 Fuel3803 Fuel3804 Fuel3805 Fuel3808
Fuel3809 Fuel3814 Fuel3815 Fuel3818 Fuel3819 Fuel3826 Fuel3829 Fuel3830 Fuel3837 Fuel3838
Fuel3841 Fuel3842 Fuel3845 Fuel3848 Fuel3849 Fuel3852 Fuel3853 Fuel3858]

dep butot
 7

set rfw 1 RestartBU8.txt

set rfr 6 RestartBU7.txt

% --- Nuclide inventory:
set inventory
30070
50100
50110
360830
400900
400910
```

```
400920
400940
400960
451030
451050
471090
531350
541310
541350
551330
551340
551350
551370
561400
571400
601430
601450
611470
611480
611490
611481
621470
621490
621500
621510
621520
631530
631540
631550
631560
641520
641540
641550
641560
641570
641600
922340
922350
922360
922370
922380
922390
932360
932370
932380
932390
942360
942380
942390
942400
942410
942420
942430
952410
952420
952430
952440
952421
962420
962430
962440
962450
```

For additional information or inputs from individual burnup steps detailing geometry and materials, it is available upon request by emailing: kmramey@gatech.edu

APPENDIX C C++ SCRIPT USED FOR RADIAL POWER PROFILES

Below, the following code reads in a detector (tally) output file from Serpent and is able to produce the normalized-averaged radial power profiles seen in Section 6.2 Cycle Results. The assembly numbers referenced in the code correspond to the ones shown in Figure C.1.

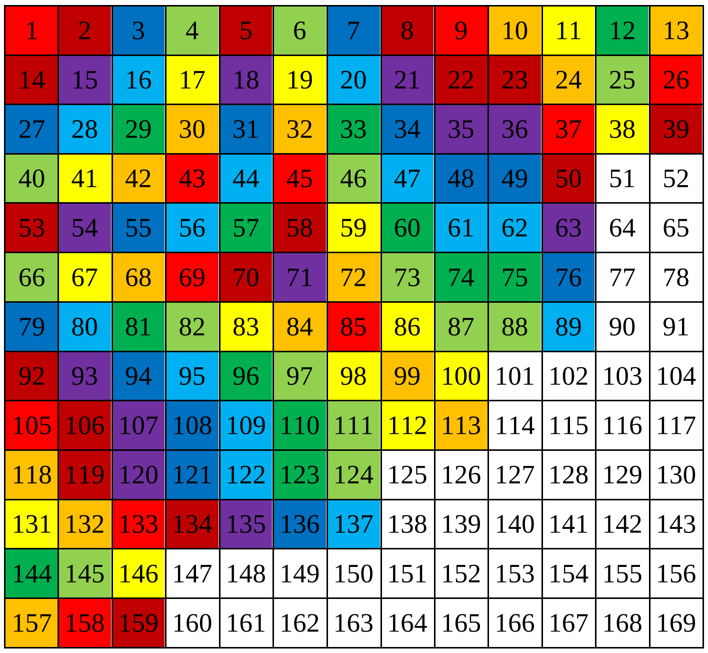


Figure C.1 Eighth-core symmetric groupings for the quarter core I²S-LWR model.

```
//Detect Reader - PPF extractor
//Made for I2S-LWR, 121 fuel bundle core
//Written by Kyle Ramey, Georgia Tech

//Initializing libraries to be used
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <iomanip>
#include <string>
#include <sstream>
#include <stdio.h>
#include <stdlib.h>

using namespace std;  //Indicates standard naming conventions within C++
//Functions to be used:
void startup (void);  //Splash function
void callInput (void);  //Asks user for input file name
void readInput (void);  //Reads the user-specified input file
void revValues (void); //Reverses values for inArray and associated errors
void average (void); // Averages over symmetric quarter-assembly locations in the core
void normalize (void); //Normalizes PPFs (from total fission rate)
void callWrite (void);  //Asks user for output file name
void writeOut (void);  //Outputs the desired data to the user-specified file

//Global variables to be used:
char filenameIn [50];  //Input file name
char filenameOut [50]; //Output file name
double revArray [169]; //Reversed desired data
double revError [169]; //
double inArray [169]; //Input file array
double inError [169];
double normQuarter [169]; // Normalized Quarter assembly
double avgQuarter [169]; // Averaged Quarter assembly
double normAvgQuarter [169]; //Normalized Averaged Quarter assembly
double maxNorm = 0;
double maxAvgNorm = 0;
double maxError = 0;
ifstream in_file;  //Allows for the user-specified input file to be read
ofstream out_file;  //Allows program to write the results for the given case

int main (void) { //The entire program runs in this function
    startup();
    callInput();
    return 0;
}

void startup (void){ //This only appears the first time the program runs
    cout << "DetReader for SERPENT - PPF extractor";
}

void callInput (void) { //The user inputs the name of the res file
    cout << "\n\nPlease give the name of the input file: ";
    cin >> filenameIn;
    cout << "\n\n";
    in_file.open(filenameIn);
    if (in_file.fail()) {
        in_file.clear();
        cout << "Could not open file. Please try again.\n \n \n";
        callInput();}
        else {
            in_file.close();
            readInput();}
}

void readInput (void){ //Reads in the user-specified data
    bool doneReading = false;
    int i = 0;
    int k = 0;
    int l = 0;
    int linnumber = 3; //Line Number
    string wholeline;
    string component;
    string error;
    char charnumb [20];
    char errornumb [10];
    double value = 0;
    maxError = 0;
    in_file.open(filenameIn);
    do {
        getline(in_file, wholeline );
        k++;
        if (k == linnumber){
            component = wholeline.substr (52,11);
            for (l = 0; l < 11; l++){
                charnumb[l] = component[l];}
            value = atof(charnumb);
            revArray[i] = value;
            error = wholeline.substr (65,7);
            for (l = 0; l < 7; l++){
                errornumb[l] = error[l];}
            value = atof(errornumb);
            revError[i] = value;
            if (revError[i] > maxError) {maxError = revError[i];}
            i++;
            linnumber++;
            if (i == 169) {doneReading = true;}}
    }while (!doneReading);
    in_file.close();
    revValues();
}

void revValues (void){
    int i = 0;
    int j = 156;
    int k  = 0;
    for (i = 0; i < 169; i++){
        inArray[i] = revArray[j+i];
        inError[i] = revError[j+i];
        k++;
        if (k == 13){
            k = 0;
            j -= 26;}}
    average();
}

void average (void){
    int i = 0;
    for (i = 0; i < 169; i++){
        avgQuarter[i] = 0;}
    double aver = 0;
    //Quarter assembly symmetry structure in PPF Excel doc "Quarter Core Layout"
    //Quarter Group 1
        avgQuarter[1-1] = inArray[1-1];
    //Quarter Group 2
        aver = inArray[2-1] + inArray[14-1];
        aver /= 2;
            avgQuarter[2-1] = aver;
            avgQuarter[14-1] = aver;
    //Quarter Group 3
        aver = inArray[3-1] + inArray[27-1];
        aver /= 2;
            avgQuarter[3-1] = aver;
            avgQuarter[27-1] = aver;
    //Quarter Group 4
        aver = inArray[4-1] + inArray[40-1];
        aver /= 2;
            avgQuarter[4-1] = aver;
            avgQuarter[40-1] = aver;
    //Quarter Group 5
        aver = inArray[5-1] + inArray[53-1];
        aver /= 2;
            avgQuarter[5-1] = aver;
            avgQuarter[53-1] = aver;
    //Quarter Group 6
        aver = inArray[6-1] + inArray[66-1];
        aver /= 2;
            avgQuarter[6-1] = aver;
            avgQuarter[66-1] = aver;
    //Quarter Group 7
        aver = inArray[7-1] + inArray[79-1];
        aver /= 2;
            avgQuarter[7-1] = aver;
            avgQuarter[79-1] = aver;
    //Quarter Group 8
        aver = inArray[8-1] + inArray[92-1];
        aver /= 2;
            avgQuarter[8-1] = aver;
            avgQuarter[92-1] = aver;
    //Quarter Group 9
        aver = inArray[9-1] + inArray[105-1];
        aver /= 2;
            avgQuarter[9-1] = aver;
            avgQuarter[105-1] = aver;
    //Quarter Group 10
        aver = inArray[10-1] + inArray[118-1];
        aver /= 2;
            avgQuarter[10-1] = aver;
            avgQuarter[118-1] = aver;
    //Quarter Group 11
        aver = inArray[11-1] + inArray[131-1];
        aver /= 2;
            avgQuarter[11-1] = aver;
            avgQuarter[131-1] = aver;
    //Quarter Group 12
        aver = inArray[12-1] + inArray[144-1];
        aver /= 2;
            avgQuarter[12-1] = aver;
            avgQuarter[144-1] = aver;
    //Quarter Group 13
        aver = inArray[13-1] + inArray[157-1];
        aver /= 2;
            avgQuarter[13-1] = aver;
            avgQuarter[157-1] = aver;
    //Quarter Group 14
        avgQuarter[15-1] = inArray[15-1];

    //Quarter Group 15
        aver = inArray[16-1] + inArray[28-1];
        aver /= 2;
            avgQuarter[16-1] = aver;
            avgQuarter[28-1] = aver;
    //Quarter Group 16
        aver = inArray[17-1] + inArray[41-1];
        aver /= 2;
            avgQuarter[17-1] = aver;
            avgQuarter[41-1] = aver;
    //Quarter Group 17
        aver = inArray[18-1] + inArray[54-1];
        aver /= 2;
            avgQuarter[18-1] = aver;
            avgQuarter[54-1] = aver;
    //Quarter Group 18
        aver = inArray[19-1] + inArray[67-1];
        aver /= 2;
            avgQuarter[19-1] = aver;
            avgQuarter[67-1] = aver;
    //Quarter Group 19
        aver = inArray[20-1] + inArray[80-1];
        aver /= 2;
            avgQuarter[20-1] = aver;
            avgQuarter[80-1] = aver;
    //Quarter Group 20
        aver = inArray[21-1] + inArray[93-1];
        aver /= 2;
            avgQuarter[21-1] = aver;
            avgQuarter[93-1] = aver;
    //Quarter Group 21
        aver = inArray[22-1] + inArray[106-1];
        aver /= 2;
            avgQuarter[22-1] = aver;
            avgQuarter[106-1] = aver;
    //Quarter Group 22
        aver = inArray[23-1] + inArray[119-1];
        aver /= 2;
            avgQuarter[23-1] = aver;
            avgQuarter[119-1] = aver;
    //Quarter Group 23
```

```
    aver = inArray[24-1] + inArray[132-1];
    aver /= 2;
        avgQuarter[24-1] = aver;
        avgQuarter[132-1] = aver;
//Quarter Group 24
    aver = inArray[25-1] + inArray[145-1];
    aver /= 2;
        avgQuarter[25-1] = aver;
        avgQuarter[145-1] = aver;
//Quarter Group 25
    aver = inArray[26-1] + inArray[158-1];
    aver /= 2;
        avgQuarter[26-1] = aver;
        avgQuarter[158-1] = aver;
//Quarter Group 26
    avgQuarter[29-1] = inArray[29-1];
//Quarter Group 27
    aver = inArray[30-1] + inArray[42-1];
    aver /= 2;
        avgQuarter[30-1] = aver;
        avgQuarter[42-1] = aver;
//Quarter Group 28
    aver = inArray[31-1] + inArray[55-1];
    aver /= 2;
        avgQuarter[31-1] = aver;
        avgQuarter[55-1] = aver;
//Quarter Group 29
    aver = inArray[32-1] + inArray[68-1];
    aver /= 2;
        avgQuarter[32-1] = aver;
        avgQuarter[68-1] = aver;
//Quarter Group 30
    aver = inArray[33-1] + inArray[81-1];
    aver /= 2;
        avgQuarter[33-1] = aver;
        avgQuarter[81-1] = aver;
//Quarter Group 31
    aver = inArray[34-1] + inArray[94-1];
    aver /= 2;
        avgQuarter[34-1] = aver;
        avgQuarter[94-1] = aver;
//Quarter Group 32
    aver = inArray[35-1] + inArray[107-1];
    aver /= 2;
        avgQuarter[35-1] = aver;
        avgQuarter[107-1] = aver;
//Quarter Group 33
    aver = inArray[36-1] + inArray[120-1];
    aver /= 2;
        avgQuarter[36-1] = aver;
        avgQuarter[120-1] = aver;
//Quarter Group 34
    aver = inArray[37-1] + inArray[133-1];
    aver /= 2;
        avgQuarter[37-1] = aver;
        avgQuarter[133-1] = aver;
//Quarter Group 35
    aver = inArray[38-1] + inArray[146-1];
    aver /= 2;
        avgQuarter[38-1] = aver;
        avgQuarter[146-1] = aver;
//Quarter Group 36
    aver = inArray[39-1] + inArray[159-1];
    aver /= 2;
        avgQuarter[39-1] = aver;
        avgQuarter[159-1] = aver;
//Quarter Group 37
    avgQuarter[43-1] = inArray[43-1];
//Quarter Group 38
    aver = inArray[44-1] + inArray[56-1];
    aver /= 2;
```

```
        avgQuarter[44-1] = aver;
        avgQuarter[56-1] = aver;
//Quarter Group 39
    aver = inArray[45-1] + inArray[69-1];
    aver /= 2;
        avgQuarter[45-1] = aver;
        avgQuarter[69-1] = aver;
//Quarter Group 40
    aver = inArray[46-1] + inArray[82-1];
    aver /= 2;
        avgQuarter[46-1] = aver;
        avgQuarter[82-1] = aver;
//Quarter Group 41
    aver = inArray[47-1] + inArray[95-1];
    aver /= 2;
        avgQuarter[47-1] = aver;
        avgQuarter[95-1] = aver;
//Quarter Group 42
    aver = inArray[48-1] + inArray[108-1];
    aver /= 2;
        avgQuarter[48-1] = aver;
        avgQuarter[108-1] = aver;
//Quarter Group 43
    aver = inArray[49-1] + inArray[121-1];
    aver /= 2;
        avgQuarter[49-1] = aver;
        avgQuarter[121-1] = aver;
//Quarter Group 44
    aver = inArray[50-1] + inArray[134-1];
    aver /= 2;
        avgQuarter[50-1] = aver;
        avgQuarter[134-1] = aver;
//Quarter Group 45
    avgQuarter[57-1] = inArray[57-1];
//Quarter Group 46
    aver = inArray[58-1] + inArray[70-1];
    aver /= 2;
        avgQuarter[58-1] = aver;
        avgQuarter[70-1] = aver;
//Quarter Group 47
    aver = inArray[59-1] + inArray[83-1];
    aver /= 2;
        avgQuarter[59-1] = aver;
        avgQuarter[83-1] = aver;
//Quarter Group 48
    aver = inArray[60-1] + inArray[96-1];
    aver /= 2;
        avgQuarter[60-1] = aver;
        avgQuarter[96-1] = aver;
//Quarter Group 49
    aver = inArray[61-1] + inArray[109-1];
    aver /= 2;
        avgQuarter[61-1] = aver;
        avgQuarter[109-1] = aver;
//Quarter Group 50
    aver = inArray[62-1] + inArray[122-1];
    aver /= 2;
        avgQuarter[62-1] = aver;
        avgQuarter[122-1] = aver;
//Quarter Group 51
    aver = inArray[63-1] + inArray[135-1];
    aver /= 2;
        avgQuarter[63-1] = aver;
        avgQuarter[135-1] = aver;
//Quarter Group 52
    avgQuarter[71-1] = inArray[71-1];
//Quarter Group 53
    aver = inArray[72-1] + inArray[84-1];
    aver /= 2;
        avgQuarter[72-1] = aver;
        avgQuarter[84-1] = aver;
    //Quarter Group 54
        aver = inArray[73-1] + inArray[97-1];
        aver /= 2;
            avgQuarter[73-1] = aver;
            avgQuarter[97-1] = aver;
    //Quarter Group 55
        aver = inArray[74-1] + inArray[110-1];
        aver /= 2;
            avgQuarter[74-1] = aver;
            avgQuarter[110-1] = aver;
    //Quarter Group 56
        aver = inArray[75-1] + inArray[123-1];
        aver /= 2;
            avgQuarter[75-1] = aver;
            avgQuarter[123-1] = aver;
    //Quarter Group 57
        aver = inArray[76-1] + inArray[136-1];
        aver /= 2;
            avgQuarter[76-1] = aver;
            avgQuarter[136-1] = aver;
    //Quarter Group 58
        avgQuarter[85-1] = inArray[85-1];
    //Quarter Group 59
        aver = inArray[86-1] + inArray[98-1];
        aver /= 2;
            avgQuarter[86-1] = aver;
            avgQuarter[98-1] = aver;
    //Quarter Group 60
        aver = inArray[87-1] + inArray[111-1];
        aver /= 2;
            avgQuarter[87-1] = aver;
            avgQuarter[111-1] = aver;
    //Quarter Group 61
        aver = inArray[88-1] + inArray[124-1];
        aver /= 2;
            avgQuarter[88-1] = aver;
            avgQuarter[124-1] = aver;
    //Quarter Group 62
        aver = inArray[89-1] + inArray[137-1];
        aver /= 2;
            avgQuarter[89-1] = aver;
            avgQuarter[137-1] = aver;
    //Quarter Group 63
        avgQuarter[99-1] = inArray[99-1];
    //Quarter Group 64
        aver = inArray[100-1] + inArray[112-1];
        aver /= 2;
            avgQuarter[100-1] = aver;
            avgQuarter[112-1] = aver;
    //Quarter Group 65
        avgQuarter[113-1] = inArray[113-1];
    normalize();
}

void normalize (void){
    double quartersum = 0;
    int j = 0;
    int k = 0;
    for (j = 0; j < 169; j++){quartersum += inArray[j];}
    maxNorm = 0;
    for (k = 0; k < 169; k++){
        normQuarter[k] = 121*inArray[k]/quartersum;
        if (normQuarter[k] > maxNorm) {maxNorm = normQuarter[k];}}
    maxAvgNorm = 0;
    for (k = 0; k < 169; k++){
        normAvgQuarter[k] = 121*avgQuarter[k]/quartersum;
        if (normAvgQuarter[k] > maxAvgNorm) {maxAvgNorm = normAvgQuarter[k];}}
    callWrite();
}

void callWrite (void){ //User specifies the name of the results file
```

```
    cout << "Please give the file name where the results will be written. \n";
    cout << "Warning: Program will overwrite data of existent files. \n";
    cout << "Please choose your file name wisely or write to a new file. \n \n";
    cout << "File name: ";
    cin >> filenameOut;
    cout << "\n \n";
    writeOut();
}

void writeOut (void) {  //Writes all the calculated values to the user-specified file
    out_file.open(filenameOut);
    int i = 0;
    int j = 0;
    out_file << "Total Fission Energy Deposition Profile (Quarter Assembly):\n\n" <<
scientific << setprecision(5);
    for (i = 0; i < 169; i++){
        for (j = 0; j < 13; j++){
            out_file << inArray[i] << "\t";
            i++;}
            i--;
            out_file << endl;}
    out_file << "\nNormalized Fission Energy Deposition Profile (Quarter Assembly):\n\n"
<< scientific << setprecision(5);
    for (i = 0; i < 169; i++){
        for (j = 0; j < 13; j++){
            out_file << normQuarter[i] << "\t";
            i++;}
            i--;
            out_file << endl;}
    out_file << "\nNormalized Averaged Fission Energy Deposition Profile (Quarter
Assembly):\n\n" << scientific << setprecision(5);
    for (i = 0; i < 169; i++){
        for (j = 0; j < 13; j++){
            out_file << normAvgQuarter[i] << "\t";
            i++;}
            i--;
            out_file << endl;}
    out_file << "\nAssociated Errors in Fission Energy Deposition:\nPosition\tTot
Fiss\tRel Error\tAbs Error\t\tMax Norm\t\tMax Avg Norm\t\tMax Rel Error\n" << scientific
<< setprecision(5);
    out_file << "1\t" << inArray[0] << "\t" << inError[0] << "\t" <<
inArray[0]*inError[0] << "\t\t" << maxNorm << "\t\t" << maxAvgNorm << "\t\t" << maxError
<< endl;
    for (i = 1; i < 169; i++){
        out_file << i+1 << "\t" << inArray[i] << "\t" << inError[i] << "\t" <<
inArray[i]*inError[i] << endl;}
    out_file.close();
    cout << "\nThe file " << filenameOut << " has been written with the calculated
values.\n";
    cout << "You will now be prompted to run the code again.\n";
    cout << "Click the X at the upper right to exit if finished.\n\n\n\n\n\n\n\n\n";
    callInput();  //Goes back to prompt the user for another input file.
}
```

REFERENCES

[1] J. J. Duderstadt and L. J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, Inc., 1976.

[2] J. R. Lamarsh and A. J. Baratta, "Intoduction to Nuclear Engineering Third Edition," Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2001.

[3] M. Nakagawa and T. Mori, "Whole Core Calculations of Power Reactors by Use of Monte Carlo Method," *Journal of Nuclear Science and Technology,* vol. 30, no. 7, pp. 692-701, 1993.

[4] G. E. Whitesides, "A Difficulty in Computing the k-effective of the World," *Transactions of the American Nuclear Society*, vol. 14, no. 2, p. 680, 1971.

[5] T. Ueki, T. Mori and M. Nakagawa, "Error Estimations and Their Biases in Monte Carlo Eigenvalue Calculations," *Nuclear Science and Engineering,* vol. 125, pp. 1-11, 1997.

[6] F. B. Brown, ""K-effective of the World" and Other Concerns for Monte Carlo Eigenvalue Calculations," *Progress in Nuclear Science and Technology,* vol. 2, pp. 738-742, 2011.

[7] B. T. Mervin, S. W. Mosher, J. C. Wagner and G. I. Maldonado, "Uncertainty Underprediction in Monte Carlo Eigenvalue Calculations," *Nuclear Science and Engineering* , vol. 173, pp. 276-292, 2013.

[8] H. J. Shim and C. H. Kim, "Real Variance Estimation Using an Intercycle Fission Source Correlation for Monte Carlo Eigenvalue Calculations," *Nuclear Science and Engineering,* vol. 162, pp. 98-108, 2009.

[9] J. Leppanen, "Serpent- a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code," VTT Technical Research Centre of Finland, 2013.

[10] V. Valtavirta, "Status and development of multi-physics capabilities in Serpent 2," in *Serpent User Group Meeting 2015*, Knoxville, TN, October 13-16, 2015.

[11] B. Petrovic, "Integral Inherently Safe Light Water Reactor (I2S-LWR) Concept: Extending SMR Safety Features to Large Power Output," in *ICAPP*, Charlotte, NC, April 6-9, 2014.

[12] American Nuclear Society, "Interim Report of the American Nuclear Society President's Special Committee on Small and Medium Sized Reactor (SMR) Generic Licensing Issues," La Grange Park, Illinois, July 2010.

[13] F. Franceschini and B. Petrovic, "I2S-LWR Core Design," I2S-LWR Second Team Meeting, Atlanta, GA, January 2014.

[14] P. Ferroni, "Integral Inherently Safe LWR project Material property database," Westinghouse Electric Company LLC, 2013.

[15] J. R. Secker and J. A. Brown, "Westinghouse PWR Burnable Absorber Evolution and Usage," in *Winter ANS Conference,* Las Vegas, NV, 2010.

[16] J.-P. A. Renier and M. L. Grossback, "Development of Improved Burnable Poisons for Commercial Nuclear Power Reactors," Oak Ridge National Laboratory, 2001.

[17] T. Flaspoehler and B. Petrovic, "Radiation Damage Assessment in the RPV of the Integral Inherently Safe Light Water Reactor (I^{2}S-LWR)," in *ISRD-15*, Aix-en-Provence, France, May 18-23, 2014.

[18] J. McKillop, "Reducing the activation of the IRIS reactor building using the SCALE/MAVRIC methodology".

[19] F. Ganda, "Innovative Neutronic Solutions for a Long Life Core of the International Reactor Innovative and Secure (IRIS)," Politecnico di Milano, Milan, IT, 2002.

[20] T. J. Wyant, Numerical Study of Error Propagation in Monte Carlo Depletion Simulations, Atlanta, GA: Georgia Institute of Technology, 2012.

[21] D. Kotlyar and E. Shwageraus, "Study of Predictor-corrector methods for Monte Carlo Burnup Codes," in *Serpent User Group Meeting 2012*, Madrid, Spain, 2012.

[22] A. Isotalo, J. Leppanen and J. Dufek, "Preventing xenon oscillations in Monte Carlo burnup calculations by enforcing equilibirum xenon distribution," *Annals of Nuclear Energy,* no. 60, pp. 78-85, 2013.

[23] J. Burns, "Reactivity Control for a PWR 19x19 Uranium Silicide Fuel Assembly (MS Thesis)," Georgia Institute of Technology, Atlanta, GA, 2015.

[24] K. Ramey and B. Petrovic, "2D Serpent Model for I2S-LWR Radial Reflector Studies," in *American Nuclear Society Winter Meeting*, Washington DC, 2015.

[25] K. Ramey and B. Petrovic, "Power Density Uncertainties in 2D Full Core Serpent Simulations," in *American Nuclear Society Winter Meeting*, Washington DC, 2015.

[26] M. J. Memmott, "Plant Parameter List for I2S-LWR," 2015.