ABSTRACT

WILLIAMS, JESSICA RANDI. Steady State and Transient Neutronics Model of the Pulsed Plasma Rocket Reactor. (Under the direction of Dr. Jason Hou).

The overall objective of this research project is to develop a high-fidelity model of the Pulsed Plasma Rocket (PPR) reactor to evaluate the reactor's neutron performance during normal operations using the high-fidelity MOOSE-based Rattlesnake reactor physics code. The PPR is a spacecraft propulsion concept that aims to achieve both high thrust and high specific impulse by using nuclear pulse power (NPP). The reactor operates by shooting a bullet of nuclear fuel through a barrel of nuclear fuel and introducing a large flux of neutrons to rapidly increase the criticality of the fuel bullet such that it changes phases to a plasma and expands to produce thrust to propel a spacecraft. Efforts have been made by various institutions and universities to study and develop initial design elements. However, the PPR reactor requires further study in the area of neutronics to improve the accuracy and fidelity of models of the propulsion system.

Ensuring that the Rattlesnake models developed in this study were accurate and high-fidelity required the generation of high-quality cross sections and a sufficiently refined mesh to provide the Rattlesnake model. The cross sections were generated by the Serpent 2 Monte Carlo code. A parametric study was carried out to determine an energy group structure for cross section generation that adequately covered the reactor's hard spectrum and provided accurate Rattlesnake results. A study was also carried out to determine spatial dependence for some of the material-based cross sections in the reactor, which helped to improve the accuracy of the cross sections used in the Rattlesnake model. The complexity of the geometry of the conceptual reactor design used in this study necessitated the use of Cubit, which is an external geometry and mesh generation toolkit that simplifies the generation of complex geometries and meshes. The effect of mesh refinement on the Rattlesnake model's accuracy was examined.

The generated cross sections and mesh were utilized in the steady state Rattlesnake model and the eigenvalues of the system were evaluated at different bullet positions and control drum rotations. As part of the verification process, the results from the Rattlesnake steady state model without control drum rotation were compared to the Serpent 2 results. The difference in values between the two models was determined to be acceptable within the scope of this study.

Once the Rattlesnake steady state model was verified, the effects of bullet movement through the reactor and control drum rotation were examined. It was determined that the bullet position did affect the overall criticality of the system and that the control drums could be used to control the reactivity in the reactor.

Preliminary work was begun on developing a transient model for the PPR in Rattlesnake. The preliminary transient model uses a simplified reactor geometry that only takes into consideration the bullet and barrel and ignores the control drums to focus on the development of a time-dependent model for the movement of the bullet through the barrel. Future work on this project will incorporate the control drums into the transient model.

Steady State and Transient Neutronics Model of the Pulsed Plasma Rocket Reactor

by
Jessica Randi Williams

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Nuclear Engineering

Raleigh, North Carolina
2022

APPROVED BY:

_____	_____
Jia (Jason) Hou Committee Chair	Maria Avramova
_____	
Agustin Abarca	

DEDICATION

To my parents Karen and Wayne Williams, and my brother Devin Williams: thank you for loving and supporting me every step of the way.

BIOGRAPHY

The author was born to Karen and Wayne Williams in November 1992 in Fort Carson, Colorado and grew up in Knightdale, North Carolina. After graduating as valedictorian from East Wake High School of Arts, Education, and Global Studies in 2011, the author attended North Carolina State University, where the author obtained a Bachelor of Science in Nuclear Engineering and graduated *magna cum laude* in 2015. Afterwards, the author worked for five years before returning to North Carolina State University in 2020 to pursue a Master of Science in Nuclear Engineering with a focus on advanced nuclear reactor design.

ACKNOWLEDGMENTS

This work is partially supported by the NASA Innovative Advanced Concepts (NIAC) Program (Grant number: 19-NIAC20B-0021). The author would like to thank their advisor, Dr. Jason Hou, for his guidance and support on this project. The author would also like to thank Dr. Ishita Trivedi from Idaho National Laboratory and Kan Ni from North Carolina State University for their technical assistance in cross section generation and conversion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. Pulsed Plasma Rocket Reactor Background

The pulsed plasma rocket (PPR) is a nuclear thermal propulsion concept for spacecraft that aims to satisfy the need for both high specific impulse and high thrust for long-distance space travel and shipping large payloads in space by using nuclear pulse power (NPP) [1]. There are two different types of NPP that have been considered: external NPP which uses a nuclear pulse or explosion outside of the spacecraft to induce propulsion and internal NPP which uses a nuclear pulse or explosion inside a pressure vessel from which heated propellant is expanded through a nozzle [2]. The PPR itself is an internal NPP concept.

The concept of using NPP was originally proposed by the National Aeronautics and Space Administration (NASA) in Project Orion to avoid the erosion problems in electric propulsion concepts and inefficiency problems in thermal thrusters [2]. Project Orion—a project sponsored by the US Government from 1958 to 1965—proposed using multiple nuclear weapon explosions to propel spacecraft [2]. In 1959, a 100-meter test flight using six impulse charges successfully demonstrated that impulsive flight could be stable [2]. Eventually the project was scrapped for multiple reasons including the enactment of the Partial Test Ban Treaty, which banned nuclear weapons explosions in space [2]. Figure 1.1 illustrates an early concept of Project Orion with shock absorbers which would protect the rocket payload from the effects of a nuclear weapon explosion.

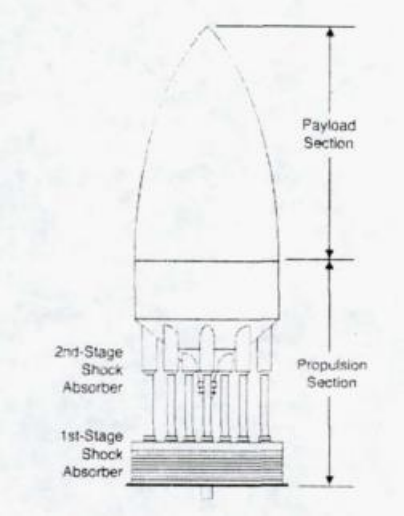

Figure 1.1: Early Project Orion concept

Although Project Orion was eventually scrapped, interest in NPP for spacecraft propulsion continued with the focus turned toward fusion concepts, which were believed to be more promising and more politically acceptable than using nuclear weapons explosions [2]. However, achieving nuclear fusion for commercial purposes has proven to be difficult, and achieving fusion on the small scale necessary for spacecraft propulsion is far off. In September 2013, NASA awarded a contract to develop the Pulsed Fission-Fusion Engine (PuFF), which is a propulsion concept that utilizes a combination of fission and fusion to achieve spacecraft propulsion [3]. The PuFF uses a z-pinch configuration to induce fusion which is enhanced by an additional fission component, requiring less energy for fission ignition and allowing for a smaller overall propulsion system than existing pure fusion concepts [3]. In the PuFF, a cylinder of deuterium-tritium (D-T) is surrounded by a depleted uranium (DU) sheath and compressed until fusion starts in the D-T mixture, generating fast neutrons that bombard the surrounding DU to induce fission which in turn increases the yield of the D-T, causing rapid expansion of the pulse unit which is expelled and produces thrust [3]. While the estimated performance of the PuFF

achieves the necessary thrust and specific impulse specifications, PuFF's need for large and durable capacitor banks that can operate between 1 to 10 hertz presents a major challenge to its implementation [4].

The PPR is derived from the PuFF concept but is smaller, lighter, and less complex [4]. In the PPR reactor, neutron-induced fission processes are utilized to implement a series of pulsed micro-explosions of very high power and eject plasma as a propellant. Using electromagnetic rings, projectiles of moderated uranium are sent through the chamber of an unmoderated uranium barrel where appropriate delivery of neutrons can induce rapid fission in the projectile, generating a plasma and producing around 200 gigajoules of energy per pulse [4]. The fully ionized plasma is expelled via an external magnetic field with a specific impulse of about 5,000 seconds with a thrust of 74,000 Newtons [4]. The system relies on maintaining the maximum power profile peak achievable with most of the power being delivered to the bullet [4].

1.2 Pulsed Plasma Rocket Reactor Conceptual Design

The conceptual PPR reactor operates by shooting a bullet of fuel through the center of a barrel of fuel. The control drums surrounding the reactor are rapidly turned to introduce a rapid and large increase of neutrons to the reactor. The large introduction of neutrons to the bullet rapidly increases the bullet's criticality, inducing a rapid phase change from a solid into a plasma, and also causes a rapid expansion. The rapidly expanding plasma is fully ionized so that the magnetic field of the nozzle can expel the plasma, creating thrust.

The conceptual PPR reactor design used for this study consists of a center barrel made of unmoderated high-assay low-enriched uranium (HALEU) metal fuel through which travels a bullet of HALEU metal fuel moderated by a lithium-hydride (LiH) center. The barrel is

surrounded by six control drums composed of half HALEU metal fuel and half beryllium absorber. For the purposes of this study, the modeled HALEU was enriched to 20% uranium-235. The LiH moderator was modeled as pure lithium-6 and pure hydrogen. Figures 1.2 and 1.3 illustrate the geometry of the reactor and Table 1.1 details the dimensions of each major component of the reactor.

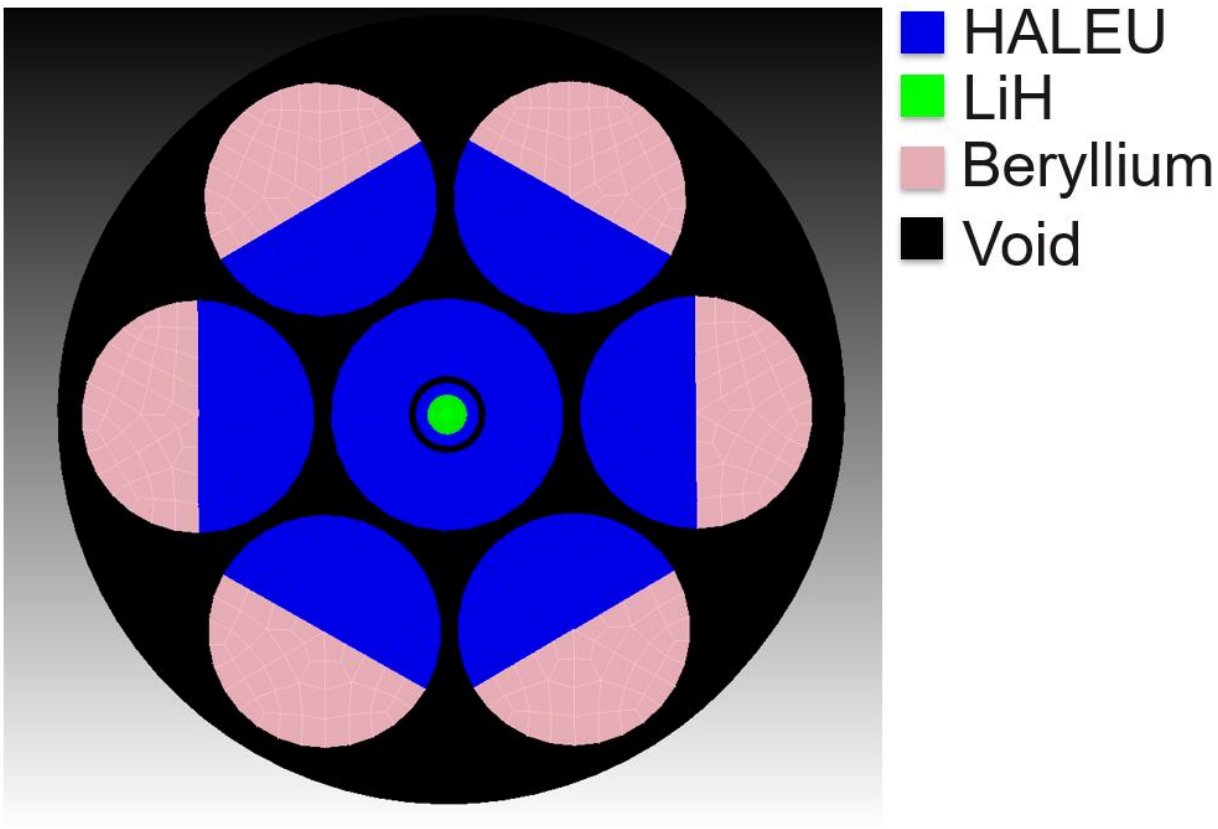

Figure 1.2: Conceptual design of the PPR (radial cross section)

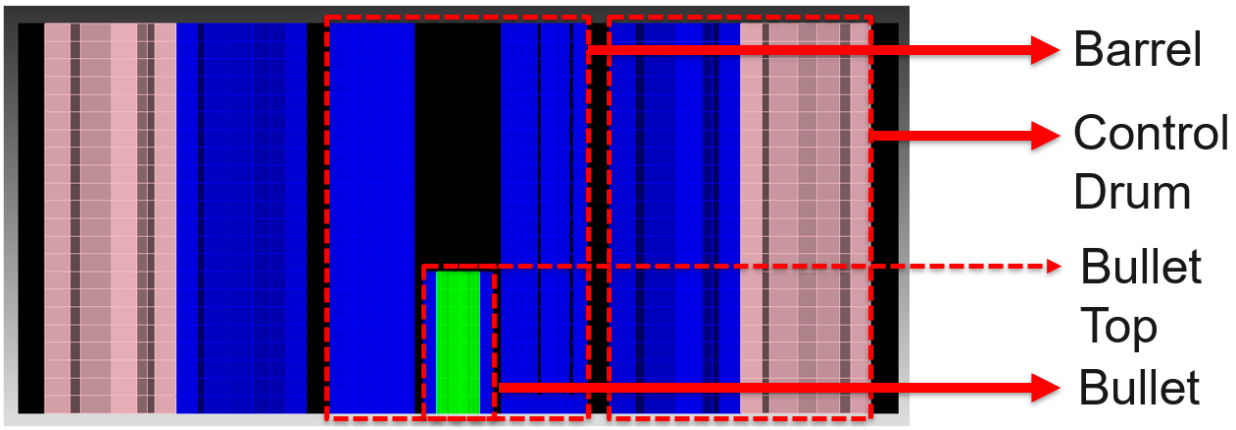

Figure 1.3: Conceptual design of the PPR (lateral cross section)

Table 1.1: Dimensions of the Conceptual PPR Reactor

Barrel	
Outer Radius	15 cm
Inner Radius	5 cm
Length	44 cm
Bullet	
Outer Radius	4 cm
Inner Radius	2.5 cm
Length	16 cm
Control Drums	
Radius	15 cm
Length	44 cm

There are additional conceptual designs that have been considered for the PPR reactor that are not evaluated in this study but may be integrated into future work as the reactor design

matures and is optimized. Such conceptual designs include features such as varying the geometry of the barrel, adding high enriched uranium (HEU), using a fast pulse laser to inject an intense pulse of neutrons into the bullet to increase the criticality spike, or using neutron reflectors instead of additional fuel in the control drums to increase the criticality spike in the bullet [4]. Figure 1.4 shows one such conceptual design in which the barrel of the reactor alternates radially between fuel and moderator and the end of the barrel includes HEU.

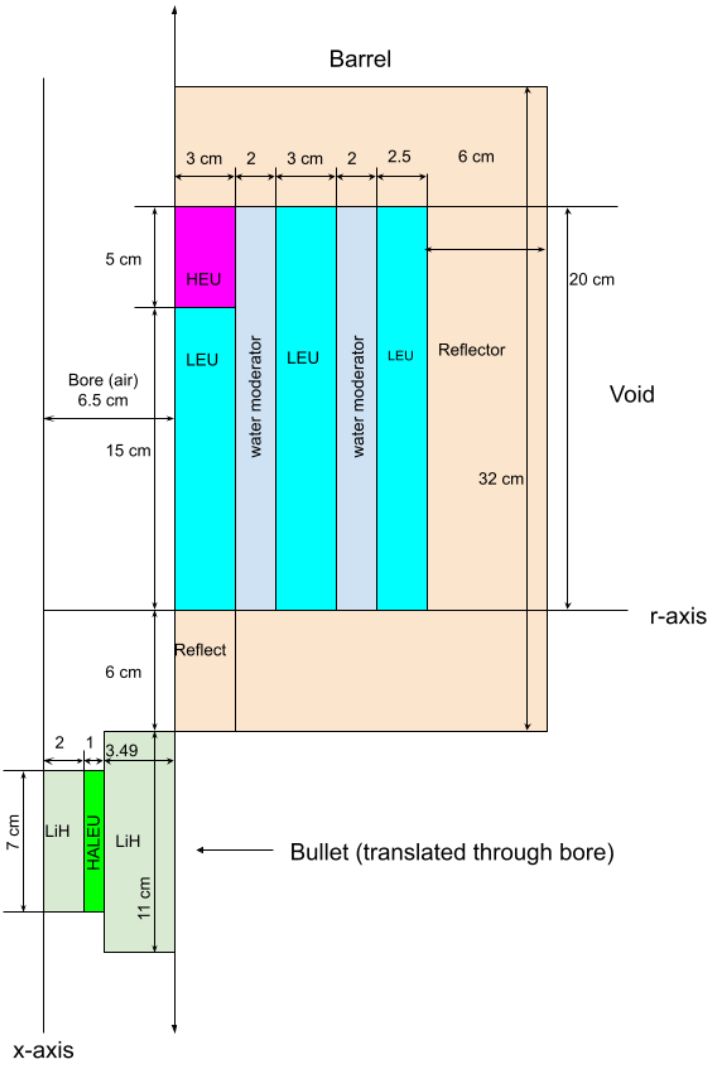

Figure 1.4: Alternative PPR conceptual reactor design

1.3. Motivation for Study

Over the last few years, efforts have been made by various institutions and universities on the initial propulsion system design, projectile speed, material properties design, etc. for the PPR [1]. Nevertheless, the PPR reactor requires further specified study in the area of neutronics in order to increase simulation accuracy and fidelity of the propulsion system. To address the requirement for a high-fidelity simulation, the reactor physics code Rattlesnake was employed in this study. Rattlesnake is a MOOSE-based reactor physics code which includes multiple functions to analyze a nuclear reactor core [5]. The objective of this study is to assess the neutronics performance of the PPR reactor during normal operations using Rattlesnake through the evaluation of the impact of the fuel projectile movement inside the reactor core and rotational position of the control drums on the criticality of the system at steady state and explore the transient modeling capabilities of Rattlesnake for the reactor.

2. COMPUTATIONAL MODELING OF THE PPR REACTOR

The goal of this study is to model the PPR reactor using the Rattlesnake deterministic code. Rattlesnake is a MOOSE-based reactor physics code that includes multiple functions to model a nuclear reactor core [5]. Rattlesnake is built on the MOOSE finite element method (FEM) framework and has a lot of common features with other MOOSE-based codes [5, 6]. The Rattlesnake model in this study requires two outside inputs to correctly simulate the performance of the PPR reactor: material-based multigroup cross sections and a meshed geometry.

2.1. Modeling Approach

The overall computational framework adopted in this study is depicted in Figure 2.1. First, a steady state Serpent 2 model is developed to generate a steady state reference solution to verify the Rattlesnake model and to generate the homogenized multigroup material-based cross sections required by the Rattlesnake simulation. Serpent 2 is a three-dimensional continuous-energy Monte Carlo reactor physics burnup calculation code which can be used to simulate particle transport for multiple applications [7]. Serpent 2 uses cross sections from continuous energy ACE format cross section libraries and additional thermal scattering data libraries for common moderators. After the steady state calculation is completed, the multigroup cross sections were condensed and homogenized for each component of the reactor. Next, the external YakXS utility is used to search the Serpent 2 output and extract the cross sections into an XML library usable by Rattlesnake [8].

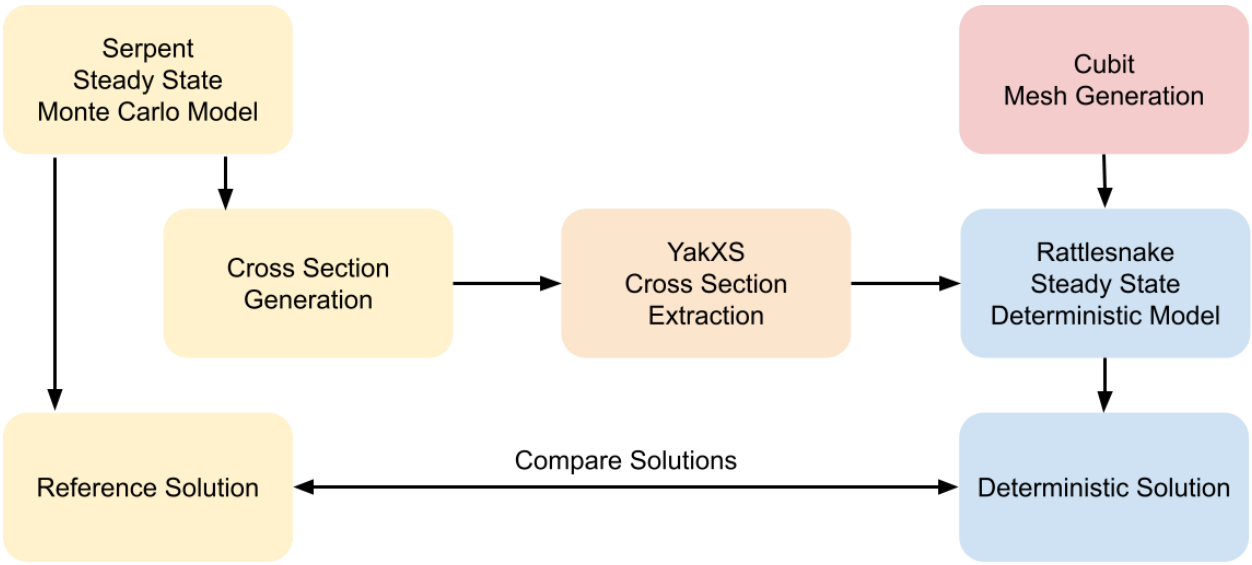

Figure 2.1: Steady state model validation flow diagram

Cubit was employed to generate the meshed geometry for the reactor into an Exodus file usable by Rattlesnake. Cubit is a comprehensive geometry and meshing toolkit owned by Sandia National Laboratory for hex mesh generation [9]. Although Rattlesnake, like other MOOSE-based codes, has built-in mesh generation functions, the complex geometry of the full PPR makes it difficult to properly model using these built-in mesh generation tools. Cubit has geometry and mesh generation and visualization tools that simplify the process of meshing complex geometries for simulations, making it ideal for generating the mesh for Rattlesnake in this study. For this study, the academic version of Cubit—Cubit Learn—was used, which limits exported mesh files to 50,000 elements, which was sufficient for the purposes of this study. Further mesh refinement, however, could improve the fidelity of the results.

Once the cross sections and mesh are generated in formats usable by Rattlesnake, the steady state Rattlesnake input was built and run. Finally, the Serpent 2 and Rattlesnake steady state solutions were compared as part of the verification of the Rattlesnake model.

2.2. Serpent Steady State Model

The Serpent 2 steady state model (see Appendix A for code) was developed to generate a reference solution as part of the verification of the Rattlesnake model and to generate cross sections for use in the Rattlesnake model. The Serpent 2 model used vacuum boundary conditions because in the conceptual design of the PPR reactor, the reactor is exposed to near-vacuum conditions. To simulate the bullet movement and control drum rotation, Serpent 2's transform functions were used to translate the bullet through the barrel and to rotate the control drums. The simulated neutron population for cross section generation included 10×10^6 neutrons per generation with 2,000 active generations and 500 inactive generations.

2.2.1. Cross Section Generation

To determine the appropriate energy bins for cross section generation at a relatively low computational cost as required by the simulation, a sensitivity study was carried out to find a proper energy structure for the neutronics simulation. In the sensitivity study, cross sections were generated using 4, 6, 7, 12, and 13 energy groups. A parametric study shows that 7-group cross sections provided the best coverage of the energy range while limiting the computational resources required to ensure minimal error. Table 2.1 lists the energy bounds of the 7-group structure.

Table 2.1: 7-Group Energy Structure

Group	Upper Energy Bound (eV)
1	1.00×10^7
2	1.36×10^6
3	9.2×10^3
4	55.6
5	4.1
6	0.63
7	0.13

To accurately account for the spatial dependence of the cross sections, it was determined that the barrel section of the reactor should be split into three separate regions radially for cross section generation to account for the high neutron leakage rate into the center bore of the reactor and between the outside of the barrel and the control drums. To determine an appropriate way to partition the barrel for cross section generation, a simplified 2D model of the reactor that only included the barrel was built in Serpent 2. Because the rate of neutron leakage in this reactor is influenced by proximity to the void spaces, the barrel was partitioned into three radial regions: an inner ring, a middle ring, and an outer ring as illustrated by Figure 2.2. Three cases with differing radial region size were considered in this study and are listed in Table 2.2.

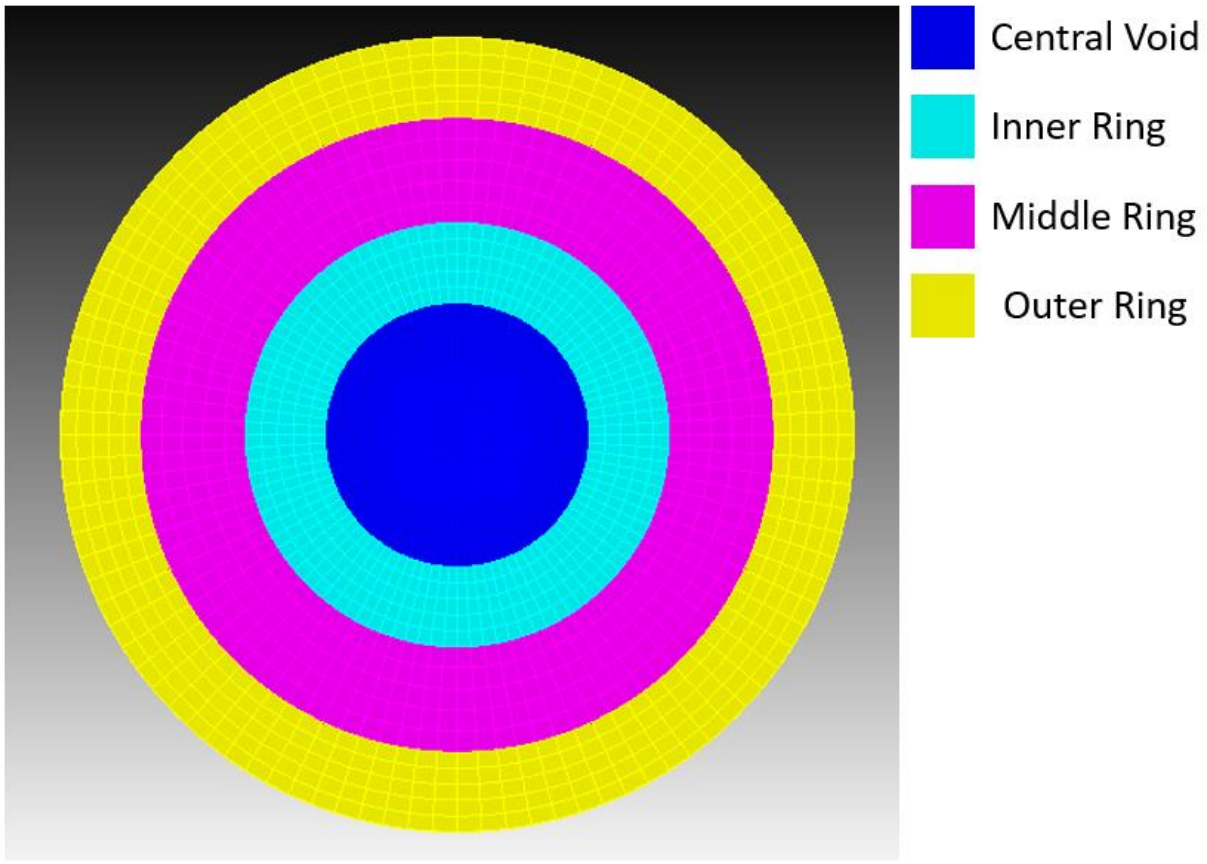

Figure 2.2: Partition of barrel for spatial cross section generation

Table 2.2: Dimensions of Barrel Partitions for Spatial Cross Section Generation

	Outer Radius (cm)			
Case	Central Void	Inner Ring	Middle Ring	Outer Ring
1	5	8	12	15
2	5	7	13	15
3	5	6	14	15

For each barrel partitioning scheme, the multigroup absorption cross sections generated by Serpent 2 were compared. Figure 2.3 compares the multigroup absorption cross sections of the three regions for each case based on their energy. In each case, each ring exhibited different absorption cross sections in the thermal and epithermal range. For cases 1 and 2, the inner and outer rings had smaller absorption cross sections in the thermal range and larger absorption cross sections in the epithermal range than the middle ring. This difference most likely stems from the spectrum shift due to the high rate of neutron leakage from the inner ring into the center void region and from the outer ring into the surrounding void. The middle ring did not experience this same rate of neutron leakage because it was not adjacent to the void regions of the reactor and thus more closely approximated the homogeneous cross sections for the entire barrel.

For case 3, there was an unexpectedly high absorption cross section for the outer ring, which was not expected because the outer ring is directly exposed to vacuum boundary conditions and should instead have a higher neutron leakage as in the first two cases. The most likely reason for this discrepancy is that the radial width of the outer ring is too thin to get enough neutron interactions in the region to generate good statistics for Serpent 2 to accurately calculate and generate cross sections for this radial region.

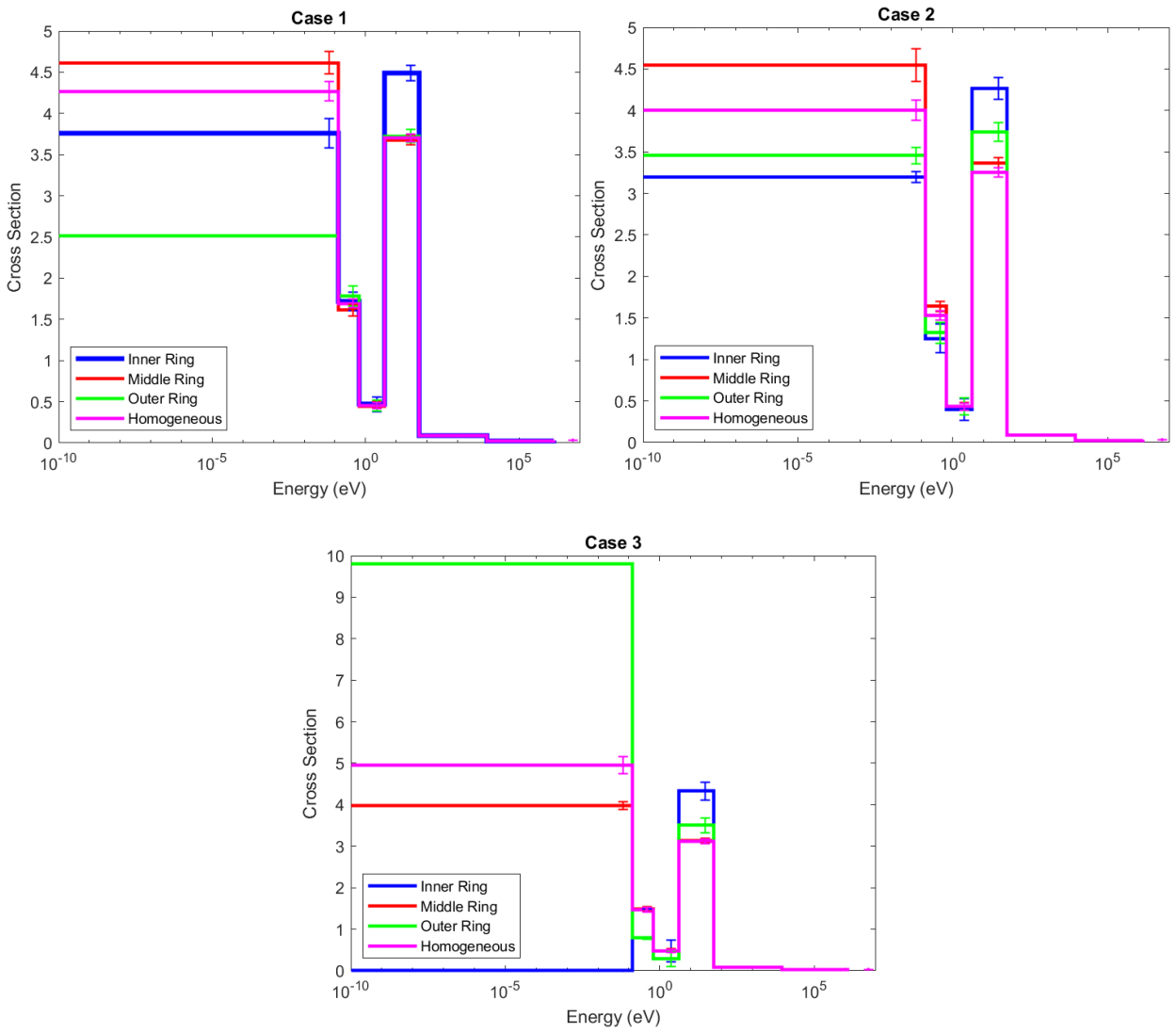

Figure 2.3: Comparison of regional absorption cross sections in the barrel

The cross sections generated by Serpent 2 for these three cases were then simulated with Rattlesnake to examine the impact of the regional cross section generation on the eigenvalue of the reactor. The Rattlesnake model performed best with the first case, which had the closest eigenvalue to the reference solution. Thus, the first case was chosen for further cross section generation. Once the best option for partitioning of the barrel was determined, the full reactor

model including the control drums was run in Serpent 2 to generate the full reactor cross sections.

2.3. Rattlesnake Steady State Model

The Rattlesnake steady state model used the S_N solver SAAF-CFEM-SN with two polar and three azimuthal calculation angles. The 7-group cross sections generated by Serpent 2 as described in section 2.2.1 were extracted into an XML library using the YakXS utility then provided to Rattlesnake as mixed neutronics materials. Vacuum boundaries were chosen for the top, bottom, and radial bounds of the meshed reactor and void because the conceptual design of the reactor exposes the reactor to near-vacuum conditions. The convergence criteria used in the solver is 10^{-6} for the eigenvalue, fission source, and flux.

The discretization scheme used for the Rattlesnake steady state model was the self-adjoint angular flux (SAAF) continuous finite element method (CFEM) with discrete ordinates (SN). SAAF is a second order formulation which is a transformation of the Boltzmann transport equation [10]. CFEM is a method for discretizing the transport equation in space that yields a continuous flux solution and has a built-in treatment for time and space [10]. SN is a method for treating the angular variable suitable for heterogeneous calculations where higher angular resolution is desired [10]. The solver used seventh-order anisotropic scattering.

2.3.1. Mesh Generation

The multigroup cross sections were assigned to computational meshes generated using Cubit to enable the Rattlesnake calculation. To model the neutrons re-entering the system and prevent the loss of neutrons in the Rattlesnake simulation, the void inside the barrel and the void encompassing the reactor and control drums were also meshed. Including the void encompassing

the control drums increased the complexity of the geometry and mesh generation, requiring manual generation of some parts of the geometry and mesh.

To generate the mesh, first the reactor geometry was built in Cubit. First, the 2D version of the reactor was created including the bullet, barrel, control drums, and the central and outer void regions. The geometry was then extended to 3D by using Cubit's sweep function. A similar approach was taken to apply a mesh to the geometry: the top face of the reactor was meshed, and the mesh was extended through the reactor by using Cubit's sweep function. Once the mesh was defined, the boundary side set was defined as the top, bottom, and side faces of the reactor for application of the boundary conditions in the Rattlesnake input. Finally, block numbers were assigned to each material region such that materials could be assigned to each block in the Rattlesnake input.

Additionally, it was determined that the mesh must be sufficiently refined to reduce the difference in eigenvalue between the Serpent 2 and Rattlesnake solutions. Two levels of refinement were examined: 8,382 elements and 29,348 elements. For the coarse mesh, the bullet, barrel, control drums, and included void were each partitioned into 20 segments azimuthally. Each of the rings comprising the bullet, the mesh of the inner void, and the barrel each had 2 radial intervals. The mesh for the control drums and exterior void were generated with Cubit's paving capability. The finer mesh was generated from the coarse mesh using Cubit's built-in mesh refinement functions.

For the coarser mesh, the Rattlesnake eigenvalues at each simulated bullet position and at the initial control drum position were on average about 865 pcm lower than the reference Serpent 2 values. This discrepancy between the solutions was deemed unacceptable within the scope of this study. The refined mesh, however, resulted in Rattlesnake eigenvalues that were within 255

pcm of the Serpent 2 reference values. This difference between the reference solution and the fine mesh solution was acceptable within the scope of this study.

Moving any part of the reactor geometry required re-meshing and re-building the boundary side sets because of the change in how the components in the reactor align, which can be a time consuming and complex task when simulating several bullet and control drum positions. To simplify simulation of the translation of the bullet through the barrel and the rotation of the control drums in Rattlesnake, it was determined that the best method would be to discretize the bullet path and control drums into equal blocks such that only the material assignments of individual blocks had to be changed for each movement.

Figure 2.4 illustrates the discretization of the bullet path and the discretization of the control drums with alternating colors. The bullet path was partitioned axially into 11 equal length segments of 4 cm each with four contiguous segments assigned to the bullet material and the rest of the segments assigned to void. To simulate movement, block material assignments were simply switched between the bullet material and void. Each control drum was partitioned into six 60^{o} sectors, with three contiguous sectors assigned to HALEU and the remaining three contiguous sectors assigned to beryllium. To simulate rotation, sector material assignments were switched between HALEU and beryllium. Swapping material definitions instead of physically transforming the mesh to simulate movement in the reactor greatly reduced the complexity of simulating movement. The bullet path and control drums discretization could be refined to evaluate the effect of smaller movements on the reactor, if desired.

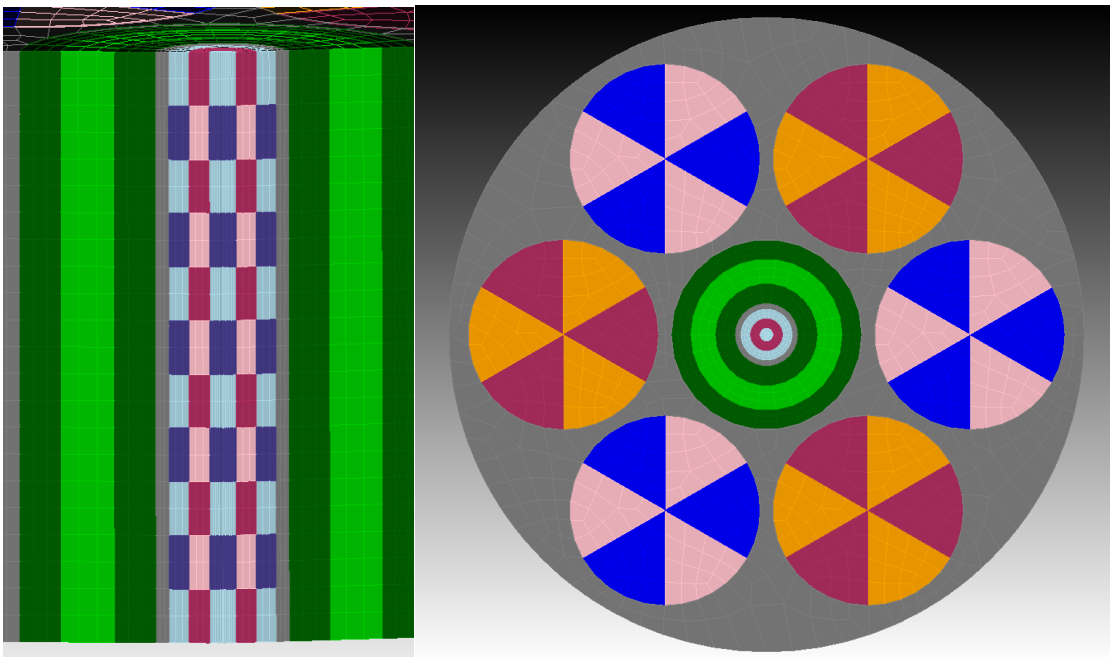

Figure 2.4: Discretization of bullet path and control drums

2.4. Preliminary Rattlesnake Transient Model

For the preliminary Rattlesnake transient model (See Appendix C for example code), a simplified version of the reactor geometry was used. As shown in Figure 2.5, only the bullet and barrel were modeled, and the control drums were ignored to focus on developing the transient of the bullet moving through the barrel. In Figure 2.5, blue corresponds to the barrel, green to the bullet, and black to the empty central bore. The LiH bullet core was also ignored for the purposes of developing the initial transient model. Like the steady state model, the bullet path in the simplified transient model was discretized into 11 segments of 4 cm each. In future work the control drums will be integrated into the transient model.

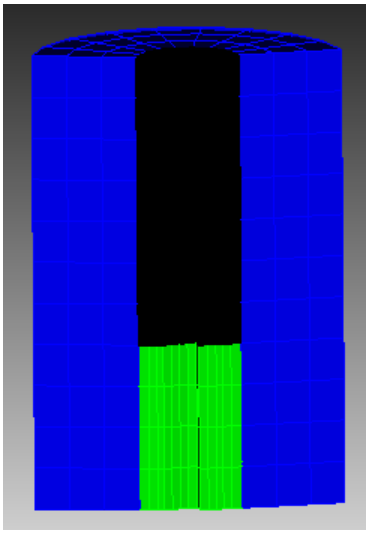

Figure 2.5: Simplified bullet and barrel geometry

The bullet was assumed to move through the barrel at a velocity of 1600 m/s. For 11 steps of 4 cm, this equates to approximately 25 μs for each step. Thus, each step of the transient was modeled for only 25 μs with time steps of 1 μs each. For consistency and because it can be used in transient calculations, the discretization scheme remained SAAF-CFEM-SN. The convergence criteria listed for the Rattlesnake steady state model in Section 2.3 were also maintained in the transient simulation.

For the first step of the transient simulation, the initial state of the reactor was set such that the bottom of the bullet was flush with the bottom of the reactor and the top of the bullet was 16 cm above the bottom of the reactor. The transient was the bullet moving an additional 4 cm into the barrel, which was simulated by switching the bottom block material—in this case uranium—with the material of the block directly above the bullet—in this case, void.

3. RESULTS AND ANALYSIS

3.1. Steady State Model

3.1.1. Verification of Rattlesnake Steady State Model

As part of the verification of the Rattlesnake steady state model, the Rattlesnake and Serpent 2 steady state solutions were compared for the case in which the control drums have the fuel halves turned completely toward the reactor for different bullet positions. The eigenvalues computed in the Rattlesnake model were consistently up to 255 pcm higher than the Serpent 2 calculated reference eigenvalues, as shown in Figure 3.1. This difference was deemed to be acceptable for the scoping of this study of the behavior of the PPR reactor.

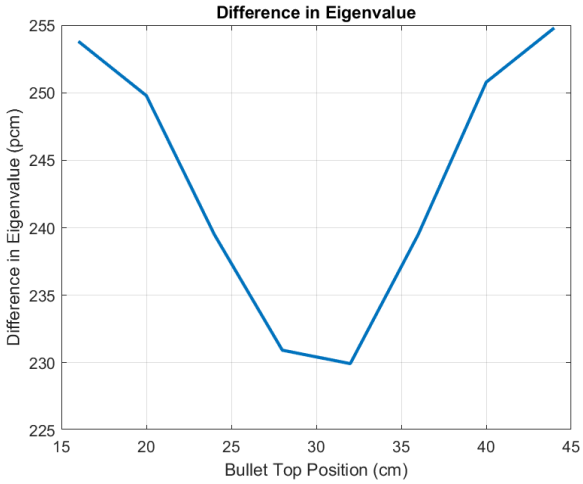

Figure 3.1: Difference in eigenvalue between the Serpent 2 and Rattlesnake non-rotated steady state solutions

Discrepancies between the Rattlesnake and Serpent 2 steady state solutions most likely stem from the region-wise cross section generation and the unoptimized computational meshes.

3.1.2. Impact of Bullet Movement on Eigenvalue

Figure 3.2 compares the Rattlesnake and Serpent 2 eigenvalues at each simulated bullet position with the control drums turned such that the fuel halves are completely facing the barrel. The maximum eigenvalues occurred when the bullet was slightly below and slightly below the center for both the Rattlesnake and Serpent 2 models with a slight dip in eigenvalue when the bullet was at the center of the reactor. The Rattlesnake maxima occurred when the top of the bullet was 20 cm and 40 cm above the bottom of the barrel. The Serpent 2 maxima occurred when the top of the bullet was at 24 cm and 36 cm from the bottom of the barrel. The eigenvalues at 20 cm and 24 cm and 36 cm and 40 cm are close for both the Rattlesnake and Serpent 2 models, suggesting that the true maxima occur between 20 and 24 cm and 36 and 40 cm. Further discretization of the bullet path could refine these results and enable finding more precise maxima of the bullet criticality. The absolute error for the Serpent 2 MC model calculation of the eigenvalues was 4.7×10^{-6} for all bullet positions.

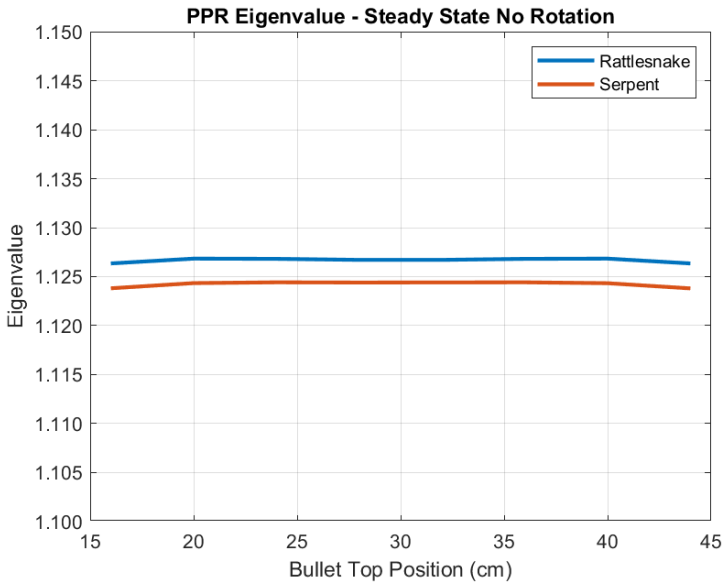

Figure 3.2: Reactor eigenvalue comparison between Rattlesnake and Serpent 2 steady state models

The reactor experienced a decrease in eigenvalue when the bullet was at the center of the reactor. Upon investigation of this phenomenon, it was determined that the confluence of neutron scattering, the fast fission factor, and fast non-leakage probability were the likely causes of the decrease in system criticality when the bullet was at the center.

Figure 3.3 displays the first two orders of the two-group neutron scattering cross sections for the reactor based on bullet position. When the bullet is at the center of the barrel, the neutron scattering cross section was at its maximum regardless of control drum rotation, energy group, and scattering order.

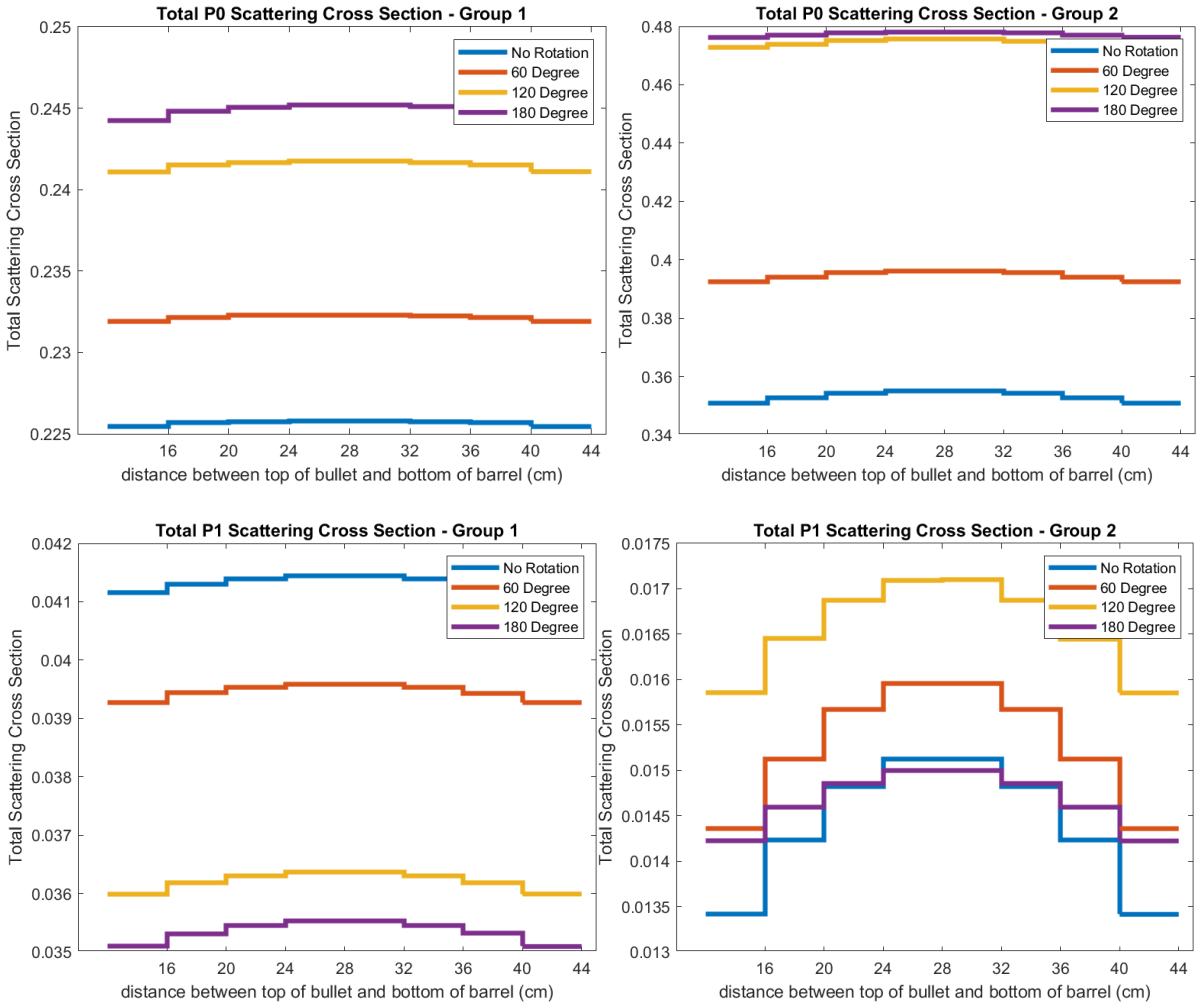

Figure 3.3: Two-group scattering cross sections by bullet position and control drum rotation

Figures 3.4 and 3.5 show the fast fission factor and fast non-leakage probability, respectively, for the PPR for each bullet position and each control drum rotation. The fast fission factor and fast non-leakage probability were significantly higher for the case in which the control drums had the fuel facing completely inwards—or the non-rotated case—than in the rotated case. The significant difference in these values between the non-rotate case and the rotated cases probably contributed to the slight dip in eigenvalue at the center of the non-rotated case whereas the rotated cases did not experience this slight dip in eigenvalue.

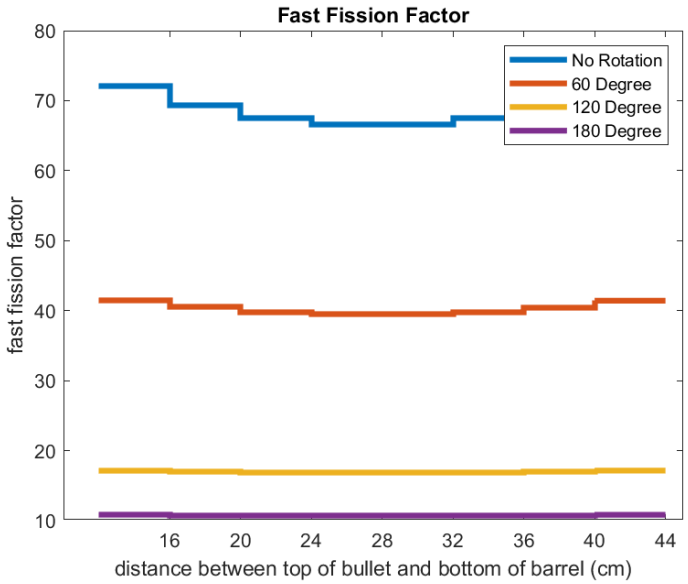

Figure 3.4: Fast fission factor by bullet position and control drum rotation

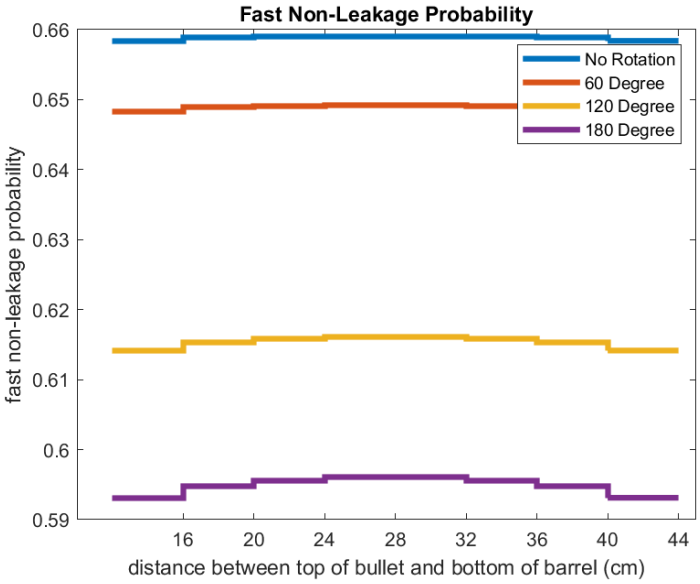

Figure 3.5: Fast non-leakage probability by bullet position and control drum rotation

3.1.3. Impact of Control Drum Rotation on Eigenvalue

After determining that the Rattlesnake steady state model for the PPR reactor is valid, the control drum rotation was simulated to determine the effect of control drum position on the reactor eigenvalue at each bullet position during steady state. In addition to the non-rotated case discussed in section 3.1.2, three different control drum positions rotating clockwise from an initial position with the HALEU half of the control drums facing completely inwards. The overall eigenvalues of the reactor decreased as the fuel half of the control drums rotated away from the center of the reactor. Rotating the fuel away from the center of the reactor decreased the exposure of fuel to the neutrons and increased the exposure of the beryllium absorber, resulting in a decreased likelihood that neutrons produced in the barrel and bullet would interact with fuel before being absorbed by the beryllium or leaking out of the reactor.

Figure 3.5 displays the steady state reactor eigenvalues for each bullet position at the three additional control drum rotational positions modeled. Unlike the non-rotated system, the

eigenvalue did not decrease when the bullet was at the center of the barrel for any of the rotated cases in both the Rattlesnake and Serpent 2 steady state models. Instead, the eigenvalues peaked when the bullet was at the center of the barrel for the rotated control drum cases.

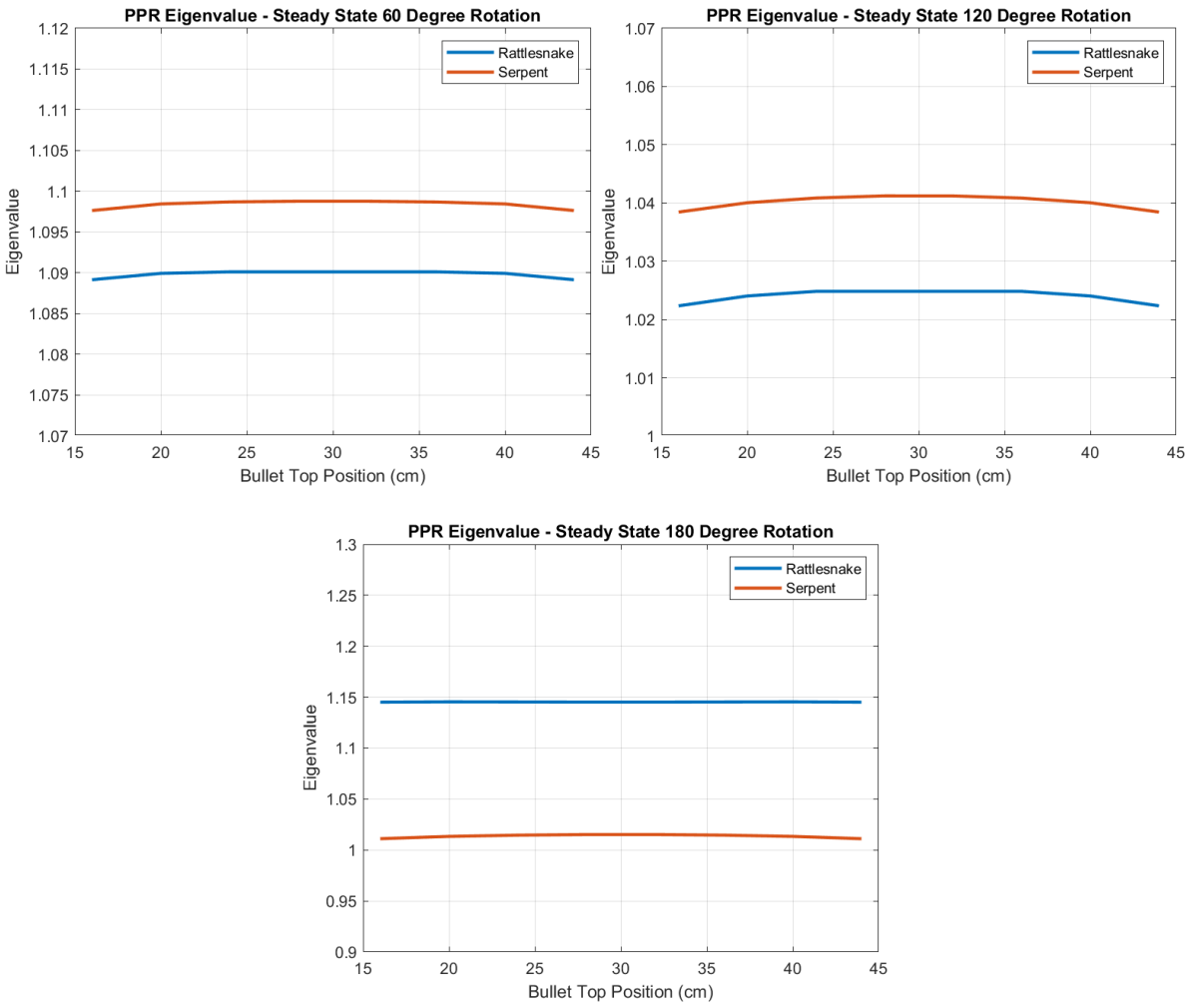

Figure 3.6: Eigenvalues at different bullet positions for different control drum positions

Table 3.1 displays the average difference between the Serpent 2 and Rattlesnake eigenvalues for each control drum position. Although the eigenvalue profiles were consistent, the larger the rotation from the initial position, the larger the discrepancy between the Rattlesnake

steady state solution and the reference solution. These discrepancies are most likely due to insufficiently refined cross-sections and unoptimized meshes. The absolute error in eigenvalues calculated by the Serpent 2 MC model ranged from 4.7×10^{-6} to 5.9×10^{-6} for all bullet and control drum positions, with the error increasing with increasing control drum rotation.

Table 3.1: Difference Between Reference and Rattlesnake Eigenvalues

Control Drum Position	Average Discrepancy (pcm)
0^o	+244
60^o	-856
120^o	-1,614
180^o	+1,897

3.2. Preliminary Rattlesnake Transient Model

For the first step of the transient model using a simplified bullet and barrel mesh, the bullet was instantaneously moved from the bottom of the barrel 4 cm further into the barrel and simulated over a time period of 25 μs with time steps of 1 μs each to approximate the bullet moving at a rate of 1600 m/s. The flux was calculated at the end of each time step and rose about 1.3% within the first 10 μs of the simulation.

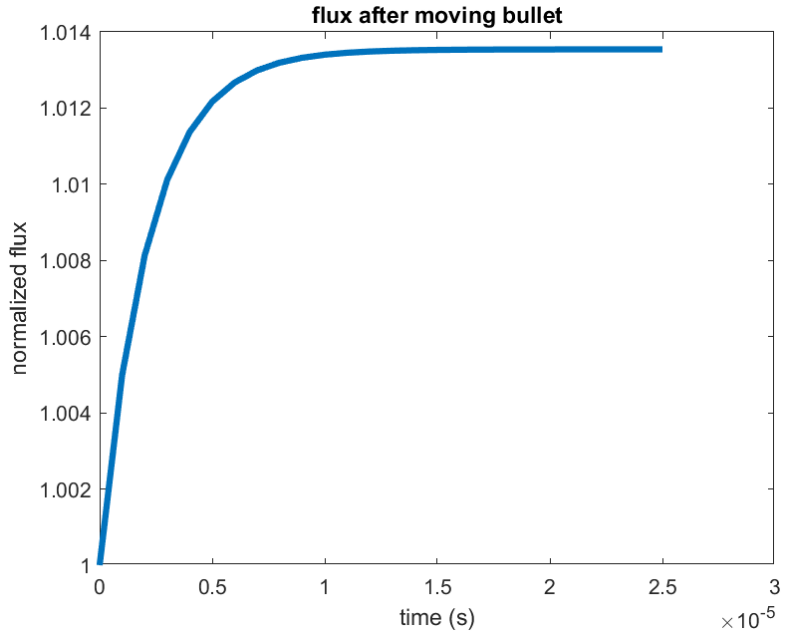

Figure 3.7: Transient flux when moving the bullet from the bottom of the barrel an additional 4 cm into the barrel

4. CONCLUSIONS AND FUTURE WORK

This study aims to evaluate the steady state and transient neutronics performance of the PPR spacecraft propulsion reactor during normal operations using the high-fidelity MOOSE-based reactor physics code, Rattlesnake. It was shown that given sufficiently refined cross sections generated using the Serpent 2 Monte Carlo code and a sufficiently refined mesh generated by Cubit, the Rattlesnake steady state model solution can produce sufficiently accurate results when compared to the reference Serpent 2 Monte Carlo steady state model. This study also demonstrated that control drum rotation can be used successfully to control the criticality of the PPR reactor, confirming that the control drums can be used to effectively manage the neutron population and assist the phase transition of the projectile from a solid to a plasma to produce thrust.

Preliminary transient analysis has shown that introducing the bullet to the reactor system can rapidly increase the flux in the reactor. Further efforts on developing a transient model will be dedicated to extending the transient to model the entire travel of the bullet through the reactor. Once the bullet transient model is complete, the transient model will be expanded to include the full PPR reactor and incorporate control drum rotation. This transient model will help to support the exploratory analysis of the reactor design and formulate a basis for future optimization of the PPR reactor design. Afterwards, a multi-physics model will then be implemented to take into account the thermal-hydraulic and thermos-mechanical feedback.

REFERENCES

[1] T.Howe, F. Bennett, N. Blaylock, “Pulsed Plasma Rocket (PPR) Phase I Final Report,” (2021).

[2] G.R. Schmidt, J.A. Bonometti, P.J. Morton, “Nuclear Pulse Propulsion - Orion and Beyond,” in 36th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, Huntsville, Alabama, July 2000.

[3] R. Adams, et al., “Developing the Pulsed Fission-Fusion (PuFF) Engine,” in 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, Ohio, July 2014.

[4] T. Howe, “Pulsed Plasma Rocket: Shielded, Fast Transits for Humans to Mars,” (2020).

[5] W. Quadros, K. Shimada, S. Owen, “Skeleton-Based Computational Method for Generation of 3D Finite Element Mesh Sizing Function,” *Engineering with Computers,* **Vol. 20**, No. 3, pp. 249-264 (2004).

[6] D. Gaston, et al., “MOOSE: A parallel computational framework for coupled systems of nonlinear equations,” Nuclear Engineering and Design, 239, pp. 1768-1778 (2009).

[7] J. Leppännen, et al., “The Serpent Monte Carlo Code: Status, Development and Applications in 2013,” *Ann. Nucl. Energy,* **82**, pp. 142-150 (2015).

[8] Y. Wang, J. Ortensi, “YAKXS – The XML Multigroup Cross Section Library,” Idaho National Laboratory (2018).

[9] W. Quadros, “CUBIT – Sandia’s Geometry & Meshing Toolkit,” in 28th International Meshing Roundtable, Buffalo, New York, (2019).

[10] Y. Wang, S. Schunert, B. Baker, V. Labouré, “Rattlesnake User Manual,” Idaho National Laboratory (2019).

APPENDICES

Appendix A: Full Core Serpent 2 Model for Steady State Simulations

```
% Serpent 2 Steady State Full PPR
% This example is the non-rotated case with the top of the
% bullet 16 cm above the bottom of the barrel. The bullet is
% moved using Serpent 2's translation function and the control
% drums are rotated using Serpent 2's rotation function.

% Fuel
mat fuel -19.3 tmp 1250 rgb 50 50 255
92235.83c 0.2
92238.83c 0.8

% Beryllium
mat ber -1.85 tmp 1250 moder bemet 4009 rgb 255 192 203
4009.83c 1.0

% LiH
mat lih -0.82 tmp 1250 rgb 100 255 100
3007.83c 0.5
1001.83c 0.5

set acelib
"/cm/shared/apps/ncsu/SERPENT/xsdata/endfb71/sss_endfb71u.xsdata
"
"/cm/shared/apps/ncsu/SERPENT/xsdata/endfb7/sss_endfb7u.xsdata"
therm bemet be.04t

% Universe Definition

surf s100 inf

%Barrel Universes
cell c101 1001 fuel -s100
cell c102 1002 fuel -s100
cell c103 1003 fuel -s100

%Drum Universes
cell c111 1011 fuel -s100
cell c112 1012 fuel -s100
cell c113 1013 fuel -s100
cell c114 1014 fuel -s100
cell c115 1015 fuel -s100
cell c116 1016 fuel -s100

cell c121 1021 ber -s100
cell c122 1022 ber -s100
```

```
cell c123 1023 ber -s100
cell c124 1024 ber -s100
cell c125 1025 ber -s100
cell c126 1026 ber -s100

%Bullet Universes
cell c141 1041 lih -s100
cell c142 1042 lih -s100
cell c143 1043 fuel -s100

% Geometry Definition

surf s000 cylz 0 0 100 -100 100

%Barrel
surf s001 cylz 0 0 5  %Bore
surf s002 cylz 0 0 8 -22 22 %Barrel inner zone (3 cm)
surf s003 cylz 0 0 12 -22 22 %Barrel middle zone (4 cm)
surf s004 cylz 0 0 15 -22 22 %Barrel outer zone (3 cm)

%Drums
surf s010 hexxprism 0 0 32 -22 22 %hex guide
surf s011 cylz 32 0 15 -22 22
surf s012 cylz 16 27.7128 15 -22 22
surf s013 cylz -16 27.7128 15 -22 22
surf s014 cylz -32 0 15 -22 22
surf s015 cylz -16 -27.7128 15 -22 22
surf s016 cylz 16 -27.7128 15 -22 22

%Bullet
surf s041 cylz 0 0 1 -38 -22
surf s042 cylz 0 0 2.5 -38 -22
surf s043 cylz 0 0 4 -38 -22

cell 99 0 outside s000 %Outer boundary

cell c001 0 fill 1001 s001 -s002
cell c002 0 fill 1002 s002 -s003
cell c003 0 fill 1003 s003 -s004


%External Void
cell c991 0 void s004 s011 s012 s013 s014 s015 s016 -s010 -s000
s001
cell c992 0 void s004 s011 s012 s013 s014 s015 s016 s010 -s000
s001
```

```
%Control Drums
cell c011 0 fill 1011 -s011 -s010 s004
cell c021 0 fill 1021 -s011 s010

cell c012 0 fill 1012 -s012 -s010 s004
cell c022 0 fill 1022 -s012 s010

cell c013 0 fill 1013 -s013 -s010 s004
cell c023 0 fill 1023 -s013 s010

cell c014 0 fill 1014 -s014 -s010 s004
cell c024 0 fill 1024 -s014 s010

cell c015 0 fill 1015 -s015 -s010 s004
cell c025 0 fill 1025 -s015 s010

cell c016 0 fill 1016 -s016 -s010 s004
cell c026 0 fill 1026 -s016 s010

%Bullet and Bore
cell c041 1 fill 1041 -s041
cell c042 1 fill 1042 s041 -s042
cell c043 1 fill 1043 s042 -s043
cell c000 1 void s043 -s001

cell c999 0 fill 1 -s001

% Boundary conditions
set bc 1

%Neutron population and cycles
set pop 10000000 2000 500

%Plots

plot 13 1000 1000
plot 23 1000 1000
plot 33 1000 1000

plot 1 1000 1000
plot 2 1000 1000
plot 3 1000 1000

/******************
* Group Constants *
******************/
```

```
set gcu 1001 1002 1003 1011 1012 1013 1014 1015 1016 1021 1022
1023 1024 1025 1026 1041 1042 1043
ene g 1 0 0.13E-6 0.63E-6 4.1E-6 55.6E-6 9.2E-3 1.36 10
set nfg g
set micro g

/******************
*    Transform    *
******************/

%The below code is used to translate the bullet through the
%barrel. The reference position has the top of the bullet flush
%with the bottom of the barrel.

trans U 1 0 0 16

%The below code is used to rotate the control drums clockwise.
%This example is non-rotated, so it has been commented out.

%trans U 11 rot 32 0 0 0 0 1 60
%trans U 12 rot 16 27.7128 0 0 0 1 60
%trans U 13 rot -16 27.7128 0 0 0 1 60
%trans U 14 rot -32 0 0 0 0 1 60
%trans U 15 rot -16 -27.7128 0 0 0 1 60
%trans U 16 rot 16 -27.7128 0 0 0 1 60
```

Appendix B: Full Core Rattlesnake Model for Steady State Simulations

```
# Rattlesnake Steady State Full PPR
# This example is the non-rotated case with the top of the
bullet 16 cm above the bottom of the
# barrel. Material block IDs are swapped to simulate rotation
and translation.

[Mesh]
        [fuelpin]
                type = FileMeshGenerator
                file = ppr.e
        []
[]

[TransportSystems]
        particle = neutron
        equation_type = eigenvalue
        G = 7
        VacuumBoundary = '1'
        [SN]
                scheme = SAAF-CFEM-SN
                family = LAGRANGE
                AQtype = Gauss-Chebyshev
                NPolar = 2
                NAzmthl = 3
                NA = 7
                tau = 0.5
                n_delay_groups = 6
                fission_source_as_material = true
        []
[]

[Functions]
        [TotalFlx_function]
                type = ParsedFunction
                value = 'F1 + F2'
                vars = 'F1 F2'
                vals = 'Flux1 Flux2'
        []
[]

[Materials]
        [nothing]
                type = VoidNeutronicsMaterial
                block = '91 74'
        []
```

```
        [bar_out]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1003'
                block = '71'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        []

        [bar_mid]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1002'
                block = '72'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        []

        [bar_in]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1001'
                block = '73'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        []

        # PPR Control Drums No Rotation
        # Fuel and beryllium material blocks are swapped to
simulate control drum rotation.

        [cd_fuel_1]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1011'
```

```
        block = '11 12 13'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_fuel_2]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '21 22 23'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_fuel_3]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '31 32 33'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_fuel_4]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '41 42 43'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_fuel_5]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
```

```
        material_id = '1011'
        block = '51 52 53'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_fuel_6]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '61 62 63'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_1]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '14 15 16'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_2]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '24 25 26'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_3]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
```

```
        library_name = 'fuelpin'
        material_id = '1011'
        block = '34 35 36'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_4]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '44 45 46'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_5]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '54 55 56'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

[cd_ber_6]
        type = MixedNeutronicsMaterial
        library_file = 'fuelpin.xml'
        library_name = 'fuelpin'
        material_id = '1011'
        block = '64 65 66'
        grid_names = 'Tbullet'
        grid = '1'
        isotopes = 'pseudo'
        densities = '1.0'
[]

# PPR Bullets Position 16
```

```
        # Material blocks are swapped between void (the “none”
blocks) and bullet material
        # to simulate translation.

        [bul_out]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1043'
                block = '801 802 803 804'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        []

        [bul_out_none]
                type = VoidNeutronicsMaterial
                block = '805 806 807 808 809 810 811'
        []

        [bul_mid]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1042'
                block = '901 902 903 904'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        []

        [bul_mid_none]
                type = VoidNeutronicsMaterial
                block = '905 906 907 908 909 910 911'
        []

        [bul_in]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1041'
                block = '1001 1002 1003 1004'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
```

```
                densities = '1.0'
        []

        [bul_in_none]
                type = VoidNeutronicsMaterial
                block = '1005 1006 1007 1008 1009 1010 1011'
        []


[]


[Executioner]
        type = NonlinearEigen

        l_tol = 1.0e-6
        l_max_its = 1000
        free_power_iterations = 1
        nl_rel_tol = 1e-6
        nl_abs_tol = 1e-6

        petsc_options_iname = '-pc_type -pc_hypre_type -
ksp_gmres_restart'
        petsc_options_value = 'hypre boomeramg 100'
[]

[Postprocessors]
        [Flux1]
                type = ElementIntegralVariablePostprocessor
                variable = flux_moment_g0_L0_M0
                execute_on = 'linear nonlinear timestep_end'
        []
        [Flux2]
                type = ElementIntegralVariablePostprocessor
                variable = flux_moment_g1_L0_M0
                execute_on = 'linear nonlinear timestep_end'
        []
        [TotalFlx_Fun_PP]
                type = FunctionValuePostprocessor
                function = TotalFlx_function
                execute_on = 'linear nonlinear timestep_end'
        []
[]

[AuxVariables]
        [TotalFlux]
                family = SCALAR
```

```
                order = FIRST
        []
[]

[AuxScalarKernels]
        [TotalScalarFlux_kern]
                type = FunctionScalarAux
                execute_on = timestep_end
                function = TotalFlx_function
                variable = TotalFlux
        []
[]

[Outputs]
        exodus = false
        csv = true
[]
```

Appendix C: Partial Core Rattlesnake Model for Transient Simulations

```
# PPR Rattlesnake Transient Model - Bullet and barrel only
# This is the steady state file from which the transient file
# draws from. The initial bullet position is with the top at 16
# cm from the bottom of the barrel.

[Mesh]
        [fuelpin]
                type = FileMeshGenerator
                file = solid.e
        []
[]

[TransportSystems]
        particle = neutron
        equation_type = eigenvalue
        G = 7
        VacuumBoundary ='1'
        [./SN]
                scheme = SAAF-CFEM-SN
                family = LAGRANGE
                order = FIRST
                AQtype = Gauss-Chebyshev
                NPolar = 8
                n_delay_groups = 6
                fission_source_as_material = true
        [../]
[]

[Functions]
        [./TotalFlx_function]
                type = ParsedFunction
                value = 'F1 + F2'
                vars = 'F1 F2'
                vals = 'Flux1 Flux2'
        [../]
[]

[Materials]
        [./bullet]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1043'
                block = '101 102 103 104'
                grid_names = 'Tbullet'
```

```
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        [../]
        [./barrel]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1002'
                block = '10'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        [../]
        [./path]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1002'
                block = '105 106 107 108 109 110 111'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        [../]
[]

[Executioner]
        type = NonlinearEigen
        solve_type = 'PJFNK'
        petsc_options_iname = '-pc_type -pc_hypre_type -
ksp_gmres_restart '
        petsc_options_value = 'hypre boomeramg 100'
[]

[Postprocessors]
        [./Flux1]
                type = ElementIntegralVariablePostprocessor
                variable = flux_moment_g0_L0_M0
                execute_on = 'linear nonlinear timestep_end'
        [../]
        [./Flux2]
                type = ElementIntegralVariablePostprocessor

                variable = flux_moment_g1_L0_M0
                execute_on = 'linear nonlinear timestep_end'
```

```
        [../]
        [./TotalFlx_Fun_PP]
                type = FunctionValuePostprocessor
                function = TotalFlx_function
                execute_on = 'linear nonlinear timestep_end'
        [../]
        [./avg_power]
                type = ElementAverageValue
                execute_on = 'linear initial timestep_end'
                variable = power
        [../]
[]

[Outputs]
        execute_on = 'timestep_end'
        file_base = SS_out
        exodus = false
        csv = true
[]
```

```
# PPR Rattlesnake Transient Model - Bullet and barrel only
# This is the transient file. The transient used in this file is
# moving the bullet from the initial position 4 cm further into
# the barrel.

[Mesh]
        [fuelpin]
                type = FileMeshGenerator
                file = solid.e
        []
[]

[TransportSystems]
        particle = neutron
        equation_type = transient
        G = 7
        VacuumBoundary ='1'
        [./SN]
                scheme = SAAF-CFEM-SN
                family = LAGRANGE
                order = FIRST
                AQtype = Gauss-Chebyshev
                NPolar = 8
                n_delay_groups = 6
                fission_source_as_material = true
        [../]
[]

[Functions]
        [./TotalFlx_function]
                type = ParsedFunction
                value = '(F1 + F2)/Norm'
                vars = 'F1 F2 Norm'
                vals = 'Flux1 Flux2 NormalizationFlux'
        [../]
[]

[Materials]
        [./bullet]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1043'
                block = '102 103 104 105'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
```

```
                densities = '1.0'
        [../]
        [./barrel]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1002'
                block = '10'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        [../]
        [./path]
                type = MixedNeutronicsMaterial
                library_file = 'fuelpin.xml'
                library_name = 'fuelpin'
                material_id = '1002'
                block = '106 107 108 109 110 111 101'
                grid_names = 'Tbullet'
                grid = '1'
                isotopes = 'pseudo'
                densities = '1.0'
        [../]
[]

[Postprocessors]
        [./Flux1]
                type = ElementIntegralVariablePostprocessor

                variable = flux_moment_g0_L0_M0
                execute_on = 'initial linear nonlinear
timestep_end'
        [../]
        [./Flux2]
                type = ElementIntegralVariablePostprocessor

                variable = flux_moment_g1_L0_M0
                execute_on = 'initial linear nonlinear
timestep_end'
        [../]
        [./TotalFlx_Fun_PP]
                type = FunctionValuePostprocessor
                function = TotalFlx_function
                execute_on = 'initial timestep_end'
        [../]
        [./NormalizationFlux]
```

```
                type = Receiver
                #outputs = none
                execute_on = 'initial'
        [../]
        [./avg_power]
                type = ElementAverageValue
                execute_on = 'initial timestep_end'
                variable = power
        [../]
[]

[MultiApps]
        [./initial_solve]
                type = FullSolveMultiApp
                app_type = RattlesnakeApp
                execute_on = initial
                positions = '0 0 0'
                input_files = move_0.i
        [../]
[]

[Transfers]
        [./copy_solution]
                type = MultiAppSystemCopyTransfer
                direction = from_multiapp
                multi_app = initial_solve
                execute_on = initial
                scale_with_keff = false
        [../]
        [./Copy_pp_Flux0]
                type = MultiAppPostprocessorTransfer
                direction = from_multiapp
                reduction_type = minimum
                from_postprocessor = Flux1
                to_postprocessor = Flux1
                multi_app = initial_solve
                execute_on = initial
        [../]
        [./Copy_pp_Flux1]
                type = MultiAppPostprocessorTransfer
                direction = from_multiapp
                reduction_type = minimum
                from_postprocessor = Flux2
                to_postprocessor = Flux2
                multi_app = initial_solve
                execute_on = initial
        [../]
```

```
        [./Copy_NormFlux_PP]
                type = MultiAppPostprocessorTransfer
                direction = from_multiapp
                reduction_type = maximum
                from_postprocessor = TotalFlx_Fun_PP
                to_postprocessor = NormalizationFlux
                multi_app = initial_solve
                execute_on = initial
        [../]
        [./Copy_avg_power]
                type = MultiAppPostprocessorTransfer
                direction = from_multiapp
                reduction_type = maximum
                from_postprocessor = avg_power
                to_postprocessor = avg_power
                multi_app = initial_solve
                execute_on = initial
        [../]
[]

[Executioner]
        type = Transient
        solve_type = 'PJFNK'
        petsc_options_iname = '-pc_type -pc_hypre_type -
ksp_gmres_restart '
        petsc_options_value = 'hypre boomeramg 100'
        start_time = 0.0
        end_time = 2.5e-5
        dt = 1.0e-6
        l_tol = 1e-8
        nl_rel_tol = 1e-6
[]

[Outputs]
        execute_on = 'timestep_end'
        file_base = Tr_out
        exodus = false
        csv = true
[]
```