2018

Neutronics Studies on the NIST Reactor Using the GA LEU fuel

Kyle A. Britton
Virginia Commonwealth University

Neutronics Studies on the NIST Reactor Using the GA LEU fuel

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

By

Kyle Anthony Britton

Masters of Science at Virginia Commonwealth University 2018

Bachelors of Science at James Madison University 2017

Dr. Zeyun Wu

Assistant Professor Department of Mechanical and Nuclear Engineering

Virginia Commonwealth University

Richmond, Virginia

Fall, 2018

Acknowledgments

The author wishes to thank several people. I would like to thank my parents for their unending love and support. I would like to thank my thesis advisor, Dr. Zeyun Wu, for his constant input, direction, and patience with me on this project. I would like to thank Dr. Robert Williams at NIST for the NBSR input deck. Last but not least I would like to thank Dr. Danyal Turkoglu at NIST as well for his aid in developing the code needed for the equilibrium core.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

NIST: National Institute of Standards and Technology

NBSR: National Bureau of Standards Reactor

HEU: High Enriched Uranium

LEU: Low Enriched Uranium

RERTR: Reduced Enrichment for Research and Test Reactors

GTRI: Global Threat Reduction Initiative

GA: General Atomics

TRIGA: Training Research Isotopes General Atomics

HPRR: High Powered Research Reactor

ATR: Advanced Test Reactor

MITRR: Massachusetts Institute of Technology Research Reactor

MCNP: Monte Carlo N-Particle

SS-304: Stainless Steel Composition 304

SU: Start Up

BOC: Beginning Of Cycle

MOC: Middle of Cycle

EOC: End of Cycle

PPF: Peaking Power Factor

ABSTRACT

Neutronics Studies on the NIST Reactor Using the GA LEU fuel

By Kyle Anthony Britton M.S.

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2018.

Major Director: Dr. Zeyun Wu, Assistant Professor, Department of Mechanical and Nuclear Engineering

The National Bureau of Standards Reactor (NBSR) located on the National Institute of Standards and Technology (NIST) Gaithersburg campus, is currently underway of fuel conversion from high enriched uranium (HEU) fuel to low enriched uranium (LEU) fuel. One particular challenging part of the conversion of the NBSR is the high average flux level (2.5×10^{14} n/cm^2-s) required to maintain experimental testing capabilities of the reactor, without significant changes to the external structures of the reactor. Recently the General Atomics (GA) Training Research Isotopes General Atomics (TRIGA) fuel has shown some promising features as a LEU candidate for the high performance research reactors such as the NBSR. The GA fuel has a long history of success in conversion of research reactors since it was developed in 1980s. The UZrH compound in the GA fuel has seen success in long term TRIGA reactors, and is a proven safe LEU alternative. This study performs a neutronics evaluation of the TRIGA fuel under the schema of the NBSR's heavy conversion requirements in order to examine whether the TRIGA fuel is a viable option for conversion of the NBSR. To determine the most optimal path of conversion, we performed a feasibility study with particular regard to the fuel dimensions, fuel rod configurations, cladding, as well as fuel structure selection. Based on the outcome of the feasibility study, an equilibrium core is then generated following the NBSR's current fuel management schema. Key neutronics performance characteristics including flux distribution, power distribution, control rod (i.e., shim arms) worth, as well as kinetics parameters of the equilibrium core are calculated and evaluated. MCNP6, a Monte Carlo based computational modeling software was intensively used to aid in these calculations. The results of this study will provide important insight on the effectiveness of conversion, as well as determine the viability of the conversion from HEU to LEU using the GA fuel.

CHAPTER 1. INTRODUCTION

The Global Threat Initiative (GTRI) enacted in the late 1950's was enacted by the United States in an effort to reduce the amount of High Enriched Uranium (HEU) around the world, a nonproliferation anti-nuclear terrorism initiative. Under this umbrella initiative, the Reduced Enrichment for Research and Test Reactors (RERTR) was formed in the late 1970's to fulfill this need in the context of research and test reactors. Typically the high enriched fuel used in research and test reactors is near weapons grade (>90% enriched) posing a serious security risk of the fuel being used for weapons of mass destruction. The RERTR has three main focuses, the development of new LEU fuels to be used in reactors(<20% enriched), the safety, design, and conversion analysis for these newly developed fuels, as well as a new means of producing the medical isotope Molybdenum-99 with the LEU fuels(our study focusing on the second of the three goals). Many power and research reactors have already been converted without issue, but some High Powered Research Reactors (HPRR) pose greater challenges. One of these reactors is the National Bureau of Standards Reactor (NBSR) located on the National Institute of Standards (NIST) campus in Gaithersburg, Maryland. The NBSR is a 20MW HPRR moderated by heavy water. The NBSR has not been converted, mainly due to its extremely high average thermal flux (2.5×10^{14} n/cm^2*s) required for the continued experimentation and operation at NIST [1,2]. Currently the NBSR is using material test reactor (MTR) HEU curved plate type fuel that is 93% enriched. The fuel is U_3O_8 with aluminum cladding, and has been shown to be extremely efficient. Other LEU fuels have been tested with some success in the NBSR, in particular the U-10Molybdenum and U-7Molybdenum LEU plate type fuels. The fuels have shown similar results, both being safe, efficient, and effective suffering a minor 10% loss in flux capabilities over the course of its use [3,4]. The downside is that these fuels are still years away from being able to be processed and used on the market. With many fuels failing to meet the high needs of the NBSR, and the viable fuels still years away from being manufactured, a clear solution would be to use a well-tested, well known, and already qualified fuel under the RERTR to perform this conversion. The Training Research Isotope General Atomics (TRIGA) fuel is exactly this. The TRIGA fuel [5] first commissioned in 1956, was developed by Edward Teller to be a fuel that "Could be given to a bunch of high school children to play with without fear they would get hurt". Designed to be inherently safe the UZrH fuel is well known for its prompt negative temperature coefficients, making a meltdown near impossible by design. The TRIGA fuel is also well known for its long core lifetimes, by name it was designed specifically for use in HPPRS. A 250kW TRIGA reactor operating 200 days a year, 8 hours per day the ^{235}U consumption is approximately 20 grams per year[6]. Lastly the TRIGA fuel is operationally flexible and currently available on the market. With minimal effort the fuel can be bought, and fit, into the pre-existing fuel elements in the NBSR, or made to fit as over 7 different types of TRIGA fuel exist around the w**o**rld[7]. TRIGA fuel also comes in a variety of enrichments, ranging from 8.5% wt. total uranium for lower power reactors to 45% wt. uranium for more powerful research reactors. Typically the TRIGA rods, because of their variety, range in radius and length. This characteristic will be utilized later on in determining the optimal fuel by making the radius a key design parameter.

Keeping this in mind, the overarching focus of our study is to answer the question "Can the NBSR's conversion needs be met by the TRIGA fuel, and can it be done without changes to the structure of the reactor?" To answer this question, one would have three requirements to meet the NBSR's specialized needs. The first is the current core fuel holdings must be maintained.

The cost and time for heavy construction on the core would invalidate the TRIGA fuel as a viable option for conversion. In this sense all changes inside the reactor must be restricted to purely the fuel elements in order to maintain the integrity of the NBSR. The second requirement for a successful conversion is to maintain the NBSR's current irradiative testing capabilities. The current flux (2.5×10^{14} n/cm^2*s) must be maintained as it is vital for the continued experimentation of the NIST facilities. Finally the last requirement for conversion is the fuel must meet the appropriate safety requirements for its new core schema. Shut down margins, moderator temperature coefficients, and other safety parameters must be examined to ensure the safety of the NBSR. In this paper, we first perform a feasibility examination of the TRIGA fuel in the NBSR. Varying fuel compositions, claddings, and fuel positions are examined to determine the exact makeup of the fuel element that will fit the NBSR's needs. The Monte Carlo code MCNP6 is used to determine the optimal reactivity given these varying parameters. After determining the optimal fuel element makeup using fresh fuel, the equilibrium core is evaluated to determine the fuels effectiveness over time using the MCNP BURN card along with python 3.6 for the fuel management code. Finally key neutronics parameters are determined by evaluating the equilibrium core for peaking power factors, flux, and other key neutronics parameters. With this we hope to make a constructive argument on whether the TRIGA fuel can meet the heavy demands of operation in the NBSR.

1.1 Overview of the NBSR

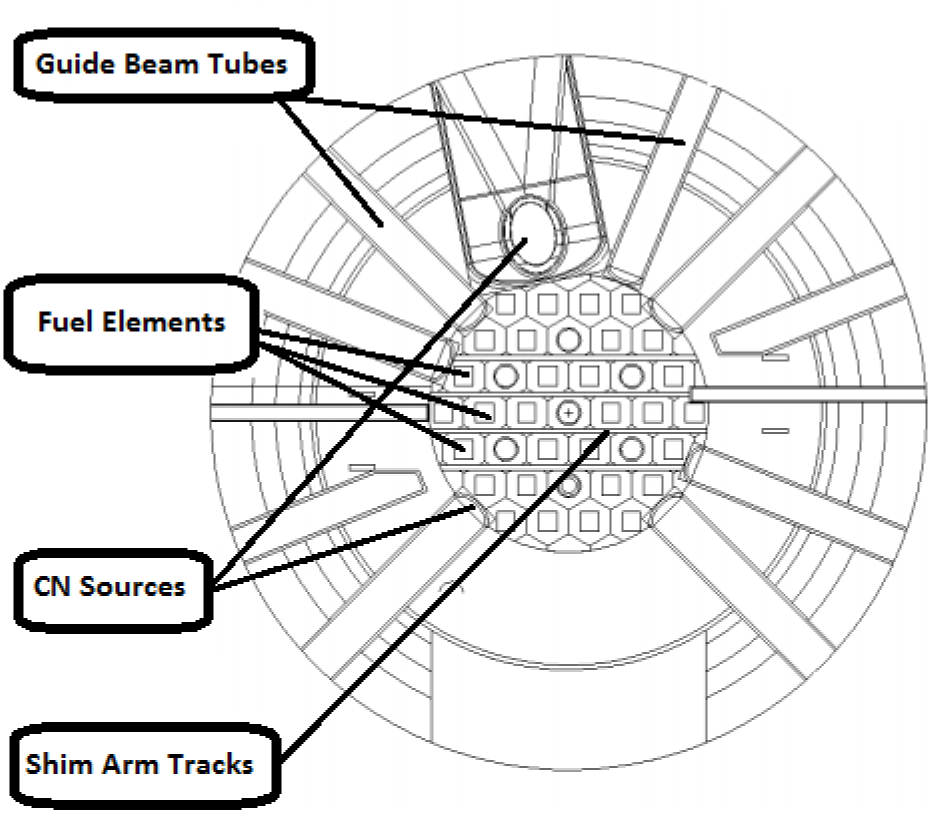


Figure 1. Core of NBSR labeled with shims.

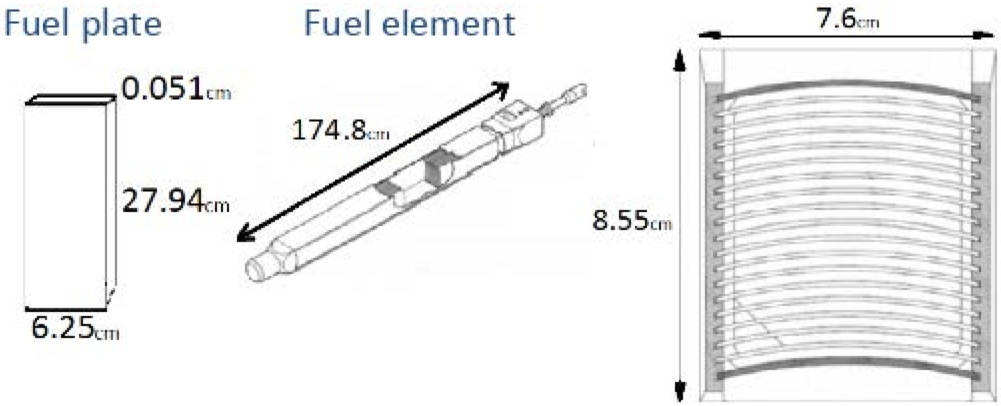


Figure 2. Individual fuel element and plate dimensions.

The NBSR first went critical December 7, 1967, and is a 20 MW thermal research reactor. The main focus of the NBSR is it is a neutron beam research and currently is equipped with 28 neutron research instruments. The NBSR is well known currently hosting over 2000 guest researchers per year. One of the most unique aspects of the NBSR is the liquid hydrogen cold moderator that is used to slow neutrons down to less than 5 meV to be transferred far away and to be utilized by multiple neutron scattering instruments in the experimental hall. This cold neutron source, only installed in the last 10 years, is currently used by 21 of the 28 instruments. Four cadmium shim arms moderate the NBSR's function, they are inserted at an angle through the side of the reactor. The angle is adjusted to control moderation. The NBSR contains 30 fuel elements arranged in 3 concentric rings as shown in the Figure 1. The fuel elements are split into 2 separate sections 15.2 cm apart as shown in Figure 2, allowing for the neutron beam instruments to point directly to the center of the core without direct line of sight of the fuel. Each element is currently fueled using 17 U_3O_8 curved MTR fuel plates with aluminum cladding, height, width, and length of 27.94cm, 0.051cm ,and 6.25cm respectively. This gives the total fuel volume in each element 296cm^3 and at 93%wt enriched, 350 grams of ^{235}U for fission. The entire core is cooled by heavy water, allowing the core to have a "loose" makeup. The heavy water moderation allows the fuel to be farther apart than other HPRR's while still maintains the integrity of its reactivity.

1.2 General Atomics LEU Fuel

As mentioned earlier, the TRIGA fuel is an attractive candidate for the conversion because of its proven safety and long core lifetimes made especially for research reactor use. TRIGA is also particularly attractive because of its history, TRIGA has been around for 50+ years giving it credibility and time to be tested in other research reactors. Currently the

Advanced Test Reactor (ATR) is analyzing the use of TRIGA fuel as a possible avenue of conversion with promising results. Lyons found that the TRIGA fuel was able to maintain the 56 day cycle of the ATR, with the minimum fission rate and power density as set forth by the navy. The reactor suffered only slight variations in the lobe power and the fast to thermal neutron flux ratios[8]. The Massachusetts Institute of Technology Research Reactor (MITRR) is also analyzing the use of the TRIGA fuel as a possible route for a quick conversion. Dunn's safety analysis of TRIGA fuel in the MITRR found that the fuel maintained the critical heat flux and reactivity necessary for operation in the beginning of life state, but failed to maintain CHF into the end of cycle[9]. TRIGA is used around the world, the Romanian IAN-R1 reactor has gone through extensive neutronics analysis for its conversion. In Jose's analysis , it was determined that the LEU conversion of the reactor could maintain thermal neutron flux throughout the lifetime of the core, as well acceptable multiplicative factor and cross sections.[10]

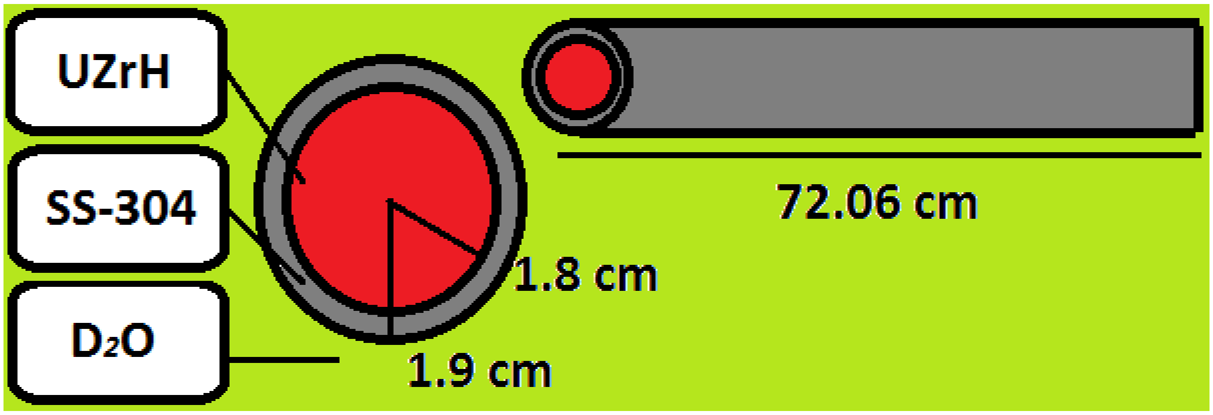


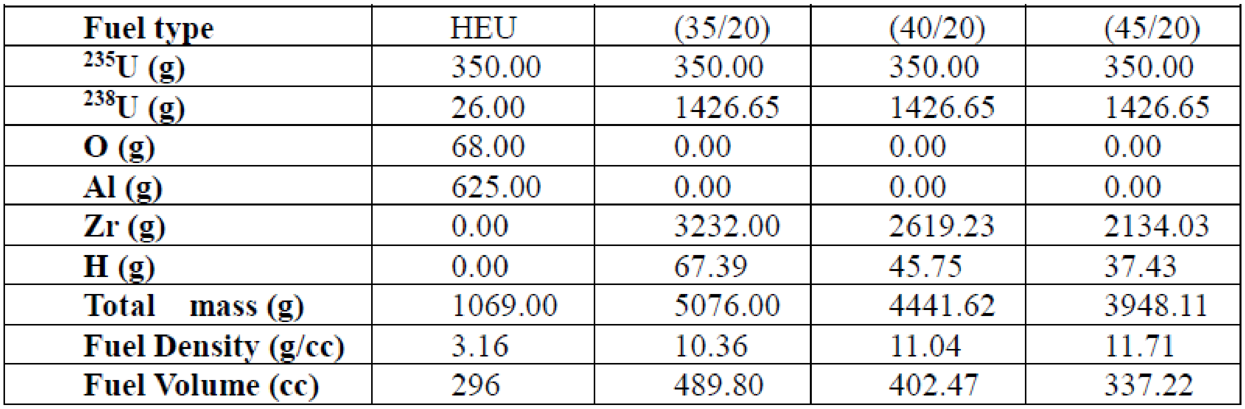
Figure 3. TRIGA Fuel Rod Dimensions and Cross Section.

The NBSR, unlike these previous studies, has unique problems and complications involved with its own conversion. The NBSR is heavy water moderated, as well as no grid flexibility, the geometry of the fuel is different, and the cladding must be more moderating for safety reasons. The lack of grid flexibility will mean the LEU compact core with have difficulty maintaining the flexibility and range of beam science experiments currently being conducted. The fuel currently being used in the NBSR is a curved plate type fuel, as opposed to the rod type TRIGA fuel with no bonding space between the fuel and cladding adding extra complications. The cladding on the HEU is alumina, as opposed to the TRIGA which for safety reasons (more in the procedure) must use stronger more moderating cladding. A more moderating cladding will make it more difficult to maintain the reactivity over the core lifetime.

CHAPTER 2. FEASIBILITY STUDY

For all of our experiments, we used MCNP6[11], a Monte Carlo based neutron tracking code, to model the reactor and determine the neutronics characteristics of the core with varying cladding, fuel composition, and fuel rod configuration. MCNP6 offers an extremely powerful 3-D neutron tracking, flexibility with calculations, and with statistical accuracy higher than its competitors. MCNP only suffers the downside of high computational time typical with Monte Carlo computations. MCNP6 is used often as a “benchmark” in industry to measure the accuracy of other computational methods, lending credibility to the software. By gaining a wide view of the parameter space of the TRIGA fuel, we hope to understand fully the effect of these parameters on the core’s health. With proper understanding of the most optimal fuel element, we can extend the analysis further into a view of the fuels effectiveness over the lifetime of the core.

Table 1. HEU Fuel vs GA TRIGA Fuel Compositions.



Fuel type	HEU	(35/20)	(40/20)	(45/20)
^{235}U (g)	350.00	350.00	350.00	350.00
^{238}U (g)	26.00	1426.65	1426.65	1426.65
O (g)	68.00	0.00	0.00	0.00
Al (g)	625.00	0.00	0.00	0.00
Zr (g)	0.00	3232.00	2619.23	2134.03
H (g)	0.00	67.39	45.75	37.43
Total mass (g)	1069.00	5076.00	4441.62	3948.11
Fuel Density (g/cc)	3.16	10.36	11.04	11.71
Fuel Volume (cc)	296	489.80	402.47	337.22

For each of the fuel element compositions used in the feasibility procedure, the amount of ^{235}U in the LEU compositions matches the amount of ^{235}U in the HEU with a tradeoff consideration of performance and economy. TRIGA fuel comes in three commercially available compositions, 35/20, 40/20, and 45/20. The first number represents the weight percent of uranium in the fuel, while the second number represents ^{235}U enrichment of the fuel (19.7%wt). As shown in Table 1, the fuels individual compositions are shown in grams in comparison to HEU for one fuel element. The $UZrH_{1.6}$ ratio was chosen for this study, as shown in source [12]. The fuel can maintain its phase, and is the most popular ratio used for this type of fuel. Notice the densities for each of the fuels in Table 1, the densities for the LEU compositions were determined using the weight percent's of each of the individual elements. The density of $ZrH_{1.6}$ was determined to be 5.66 g/cc and natural uranium density as 19.1 g/cc [13]**,** and the formula used below.

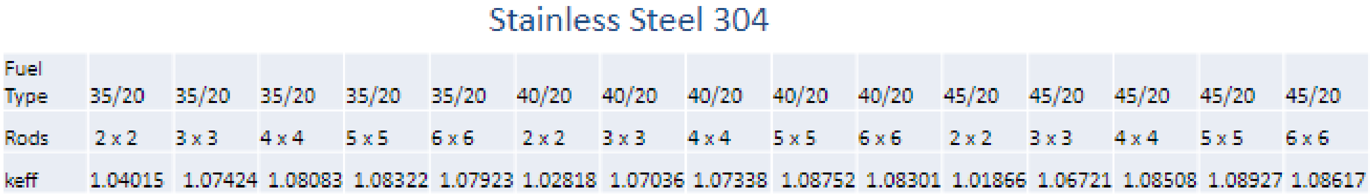
$$\text{Density of fuel} = \text{Weight\% U} \times \text{Density U} + \text{Weight\% ZrH} \times \text{Density ZrH}$$

Stainless Steel-304(SS-304) and Incoloy-800, are the most common cladding materials used for the TRIGA fuel, and they are the two considered in this study. As shown in the figure below, the SS cladding is predominantly iron while the Incoloy is a mixture of predominantly iron and nickel. Aluminum has been used in the past as a cladding, as it absorbs less neutrons and can be useful in preserving flux. In this study it is not considered, due to safety concerns, as the long lifetime requirement of the TRIGA fuels can lead to significant corrosion and blistering[13]. Four different homogenous fuel configurations were examined in this study, as shown in the figure above, the rods were arranged in a 3 × 3, 4 × 4, 5 × 5, and 6 × 6 array each keeping the amount of ^{235}U at 350 grams per element. This was kept constant by limiting the radius of the rods depending on the composition, and the fuel length was set at 33 cm in length to match the length of the current HEU plates. The 15 inch gap distance in the HEU core is also kept constant in the LEU core. Fuel rod configuration is generally considered vital to the neutronics of a heterogeneous core. In thermal reactors the geometry of the fuel can lead to a difference in neutron escape probability during the neutron slowing down process, leading to direct differences to the reactivity of the core. It is likely that the more homogenous fuel cells will have higher reactivity, due to less geometric shielding (higher surface area to volume ratios). The material compositions and densities were determined using the PNNL compendium[14]. For each criticality calculation in MCNP, the KCODE card was used with 10,000 particle histories for a total of 110 cycles with 10 skipped in order to bring the standard error of the reactivity to below .001(.1%).

2.1 Results of the Feasibility Study

The k_{eff} results of the LEU cores are summarized in Table 2 and Table 3.

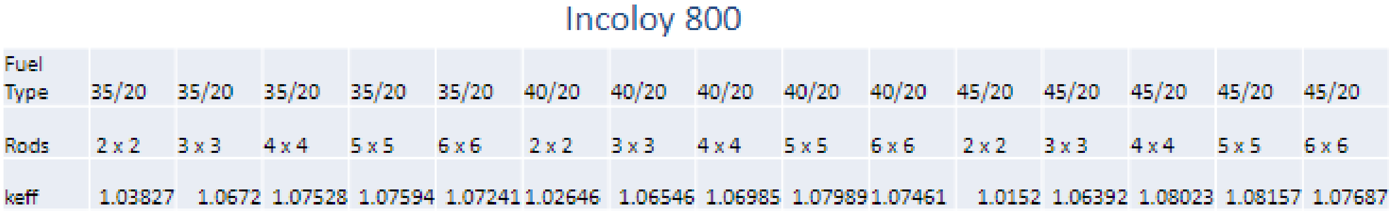
Table 2. Reactivity of SS-304 Clad Rods.



Stainless Steel 304

Fuel Type	35/20	35/20	35/20	35/20	35/20	40/20	40/20	40/20	40/20	40/20	45/20	45/20	45/20	45/20	45/20
Rods	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6
keff	1.04015	1.07424	1.08083	1.08322	1.07923	1.02818	1.07036	1.07338	1.08752	1.08301	1.01866	1.06721	1.08508	1.08927	1.08617

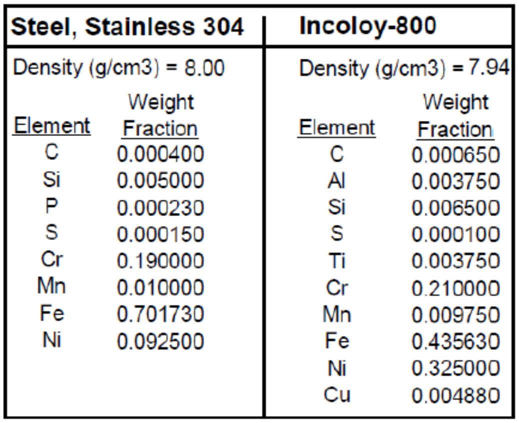
Table 3. Reactivity of SS-304 Clad Rods.



Incoloy 800

Fuel Type	35/20	35/20	35/20	35/20	35/20	40/20	40/20	40/20	40/20	40/20	45/20	45/20	45/20	45/20	45/20
Rods	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6	2 x 2	3 x 3	4 x 4	5 x 5	6 x 6
keff	1.03827	1.0672	1.07528	1.07594	1.07241	1.02646	1.06546	1.06985	1.07989	1.07461	1.0152	1.06392	1.08023	1.08157	1.07687

Table 2 contains the results using the Incoloy-800 cladding while Table 3 is the results using the Stainless steel-304 cladding. Examining the difference in k_{eff} values between the graphs shows a general trend of the SS-304 cladding having a higher reactivity then the Incoloy-800.



Steel, Stainless 304		Incoloy-800	
Density (g/cm3) = 8.00		Density (g/cm3) = 7.94	
Element	Weight Fraction	Element	Weight Fraction
C	0.000400	C	0.000650
Si	0.005000	Al	0.003750
P	0.000230	Si	0.006500
S	0.000150	S	0.000100
Cr	0.190000	Ti	0.003750
Mn	0.010000	Cr	0.210000
Fe	0.701730	Mn	0.009750
Ni	0.092500	Fe	0.435630
		Ni	0.325000
		Cu	0.004880

Figure 4. Cladding Compositions.

In Figure 4, the specific compositions of each of these claddings are defined, and with careful examination, the Incoloy-800 cladding is made up of 32.5% nickel by weight. The higher absorption cross section of the nickel in comparison to iron (the main metal in the SS-304 cladding), would lead to a total higher absorption cross section in the Incoloy-800, properly justifying the k_{eff} difference. Careful comparison of the tables shows the case with the highest reactivity, was the stainless steel 304 cladding with 45/20 fuel arranged in a 5 × 5 configuration(k_{eff} =1.08927). Likewise the lowest reactivity was exhibited by Incoloy-800 clad 45/20 fuel in the 3 × 3 configuration (k_{eff} = 1.06392). The highest reactivity case can be naturally explained due to the higher uranium content of the rod, and the homogeneity of the configuration. More Uranium by weight means less moderating hydride metal and a higher reactivity. As mentioned before as well, the higher density fuel compositions have a significantly smaller volume. This smaller volume and similar surface area leads to the fuel experiencing less geometric self-shielding, and therefore a higher reactivity. The lowest reactivity is more difficult, the high volume of the rods causing self-shielding and incoloy-800 cladding having a higher absorption cross section appear to be the main factors in why it is the lowest. Figure 5 are the results of the stainless steel-304 clad rods plotted on the same graph, the k_{eff}-variation curve displays the linear relationship between the increase and decrease in fuel enrichment. For the 5×5 and 6×6 configurations the reactivity appears to increase linearly, while the 4×4 is in the middle and the 3×3 decreases with enrichment. This is most likely due to the changes of the moderating condition from over moderating (for 5×5 and 6×6 cases) to under moderating (for the 3×3 case) and the 4×4 case is on the barrier between the two.

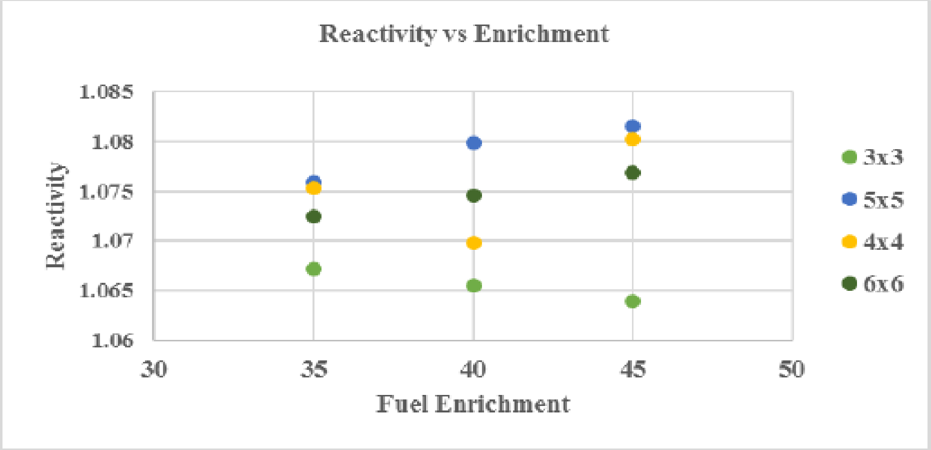


Figure 5. k_{eff} values determined vs fuel compositions, color coated for geometry.

In summary the results presented in tables 2 through 4 show the level of homogeneity in the configuration of the rods can directly affect the reactivity. The higher absorption cross section in the Incoloy-800 cladding had a negative effect on the reactivity of the rods, and the 45/20 fuel contained the highest amount of uranium and displayed the highest reactivity. The results determined that the highest reactivity and therefore most attractive case for determining the equilibrium core was the 5 x 5 configuration using SS-304 cladding with the 45/20 fuel composition.

CHAPTER 3. EQUILIBRIUM CORE GENERATION

One of first tasks of any reactor physics analysis is determining the equilibrium core for a given fuel. The equilibrium core is the fuel's effectiveness over the course of its lifetime, and involves burning and shuffling the fuel to maximize its efficiency. Determining the equilibrium core is absolutely necessary as evaluation of a fuel's effectiveness when its fresh, does not allow an accurate analysis of the fuels performance in its steady state over years. The LEU TRIGA fuel evaluated feasibility involved study of only fresh fuel elements, meaning for a proper neutronics analysis an equilibrium core must be obtained to determine the steady state power and flux it operates at. The TRIGA fuel element configuration determined to be most effective in the previous study[15] will serve as the basis for our equilibrium core search.

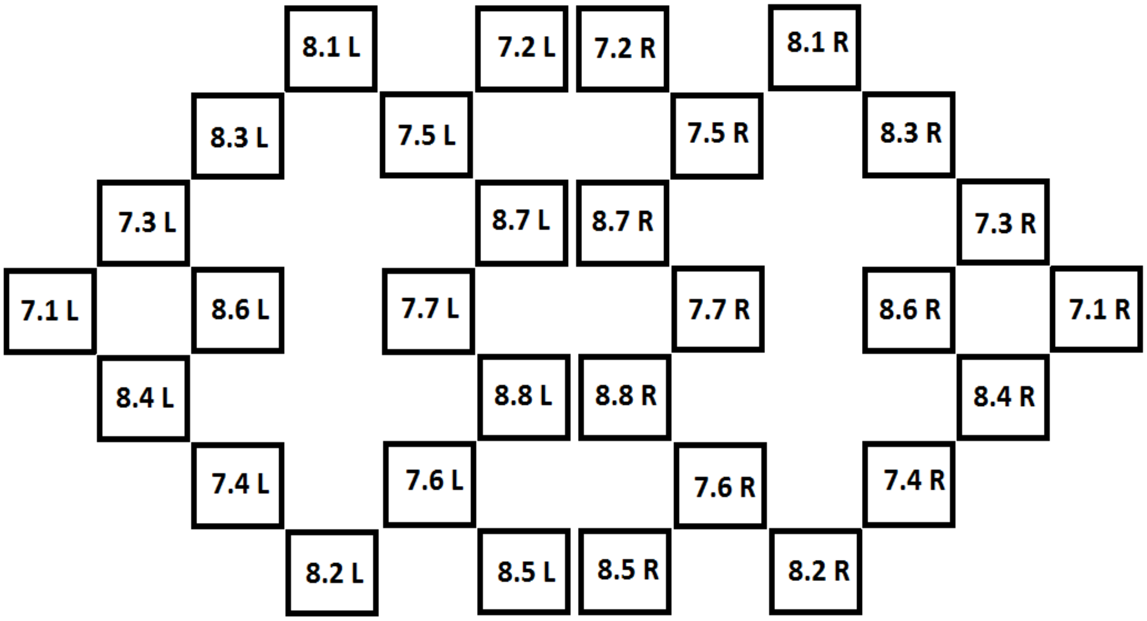


Figure 6. Fuel shuffling scheme in NBSR.

The NBSR contains 30 fuel elements, and under its current shuffling schema, this means 16 elements will be burned for eight cycles while 14 elements are burned for seven cycles. The fuel management scheme for the NBSR is shown in Figure 6, with all thirty elements labeled. Each fuel position is identified with two numbers and a letter, the first number denoting how many cycles the fuel will be in the core, with the second designating the current cycle the fuel cell is in. The letter designates the side of the reactor, as the shuffling is symmetric. This means that at the beginning of the cycle, the 8.1 and 7.1 fuel elements are the fresh newly loaded fuel, and the 8.8 and 7.7 fuel elements have been burned for their full lifetime. At the end of the cycle the 8.8 L and 7.7 L fuel elements will be disposed of, and the 8.7 Land 7.6 L fuel elements will be moved into their place, and up the line the fuel elements are all moved to the new position to be until the 8.1 L and 7.1 L positions are filled with new fuel and the cycle restarts. This means at the beginning of each cycle 4 fresh elements are loaded and 4 elements are removed. Each cycle in the NBSR is split into 4 separate sections, each with an individual shim are position as well as burn time. The first the startup (SU) state is 1.5 days long, and is necessary to allow burnable neutron poisons such as Xe-135 to reach an equilibrium before the beginning of core (BOC) and transitioning to the middle of core (MOC) states. The BOC and MOC are 18.5 days long each, and transition into the end of core (EOC) state, in which the reactor is shut down and allowed to cool for 14 days. The MOC state marks the point at which the shim arms are halfway inserted, and the EOC is when the shims are completely out.

Currently the NBSR operates on a 38.5 day cycle using its HEU fuel. This is absolutely necessary to maintain for continued ease of operation and experimental capabilities of the facilities. Keeping this in mind, our study has prioritized that the conversion between the HEU and LEU fuels are economic and as timely as possible. For this reason, the HEU fuel shuffling scheme is kept and used in for the TRIGA fuel, as it is proven and effective in the NBSR. The cycle length and cycle states are also similar. Table 4 shows a comparison between the cycles for the HEU and LEU case. As shown the only difference in the methodology between the two shuffling schema, are the shim angles at which each cycle state rests as well as the difference in fuel. The fuel rods were split into six axial zones in each of the 30 fuel elements for homogeneity in the burn (three above the 15 inch gap and three below). The fuel rods are clad in 0.04 cm of SS-304, and are arranged in a 5×5 configuration at 0.5 cm in diameter. This is the same as what was determined in the feasibility study above.

Table 4. The Control Shim Positions of HEU and LEU Core at Equilibrium States.

	HEU[a]	LEU
SU	19.7[b]	23.0
BOC	14.6	14.0
MOC	9.20	7.00
EOC	0.00	0.00

[a]The results for HEU were obtained from Ref. [16].

[b]The position is shown in the unit of angle of degrees.

3.1 Equilibrium Core Generation Methodology

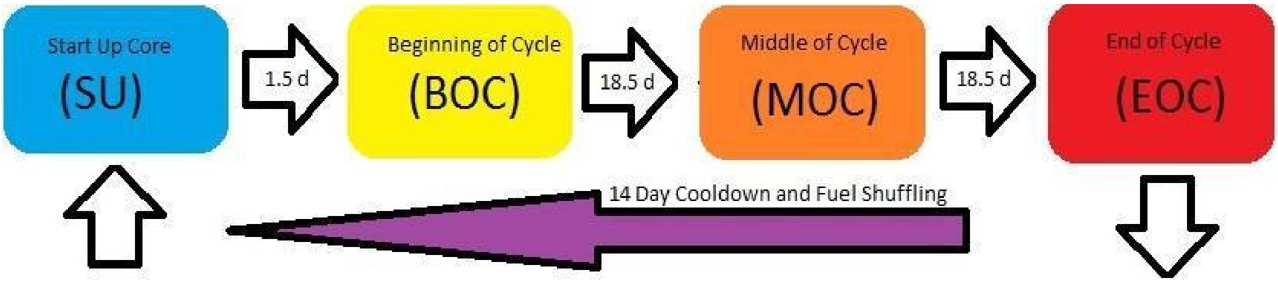


Figure 7. A multi-step diagram to generate the equilibrium core.

The fuel inventories were calculated using the MCNP6 BURN card feature. MCNP6 marked the merge of MCNPX and MCNP5, creating a neutron transport code all in one package. On top of that the MCNP6 is significantly faster, boasting speeds from 30-50% faster with the combination of MPI and threading[17]**.** The MCNP input deck gifted from NIST was modified with the appropriate BURN parameters. A fuel management code was used to automate this process for Python 3.6 on Windows 10. Shown in Figure 7 is a flow diagram to help illustrate the iterative schema developed. The process automates the creation of MCNP input decks, writing a text file for each cycle state depending on the individual shim position, burn time, and runs MCNP remotely. The process also automates the shuffling and refueling of the fuel elements by reading the output file created by MCNP, allowing nearly full automation of the equilibrium core process. MCNP6 suffers from a small bug in its BURN feature. When an isotope is not recognized as a part of the MCNP library, it is ignored resulting in a non-physical loss of mass in the core (<<1%).To compensate this, the fuel management scheme sums the masses at the end of a state, and does a comparison. If there is a difference, it will make up for it using bismuth, as this was found to have little to no effect of the BURN process[16].The MCNP input used 180 materials for the burn(30 fuel elements* 6 axial zones) and operated with 10,000 particle histories per cycle, for 100 cycles and 10 skipped cycles. This was decided so the standard error on the k_{eff} would be less than 0.001. The MCNP command line was also adjusted to run using all 8 cores. During the calculations, an element in the fuel is disregarded if it reaches below 10^{-10} grams, and each of the four states are given 14 days rest to properly adjust the k_{eff}.

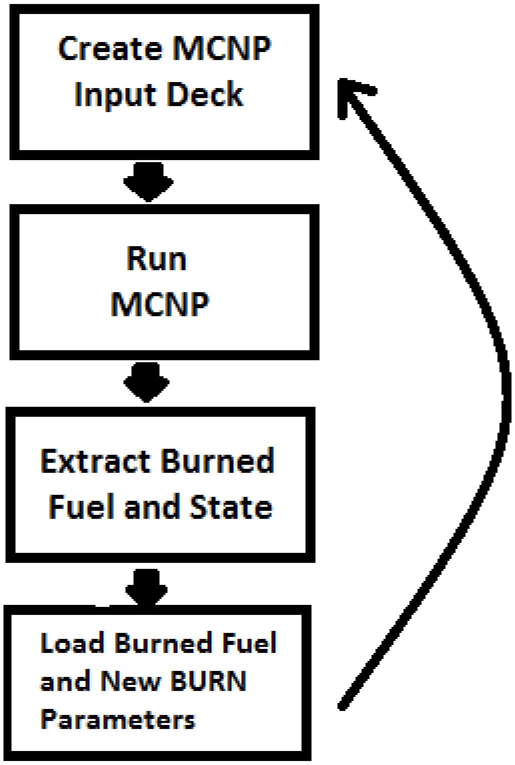

Create MCNP Input Deck
Run MCNP
Extract Burned Fuel and State
Load Burned Fuel and New BURN Parameters

Figure 8. A flowchart to perform fuel management in Python 3.6.

3.2 Results of the Equilibrium Core

The results of the equilibrium core are shown below in Figure 9, the individual reactivities determined for each of the four states are plotted vs number of cycles to illustrate the convergence of the k_{eff} value. In initial testing, the fuel rods were failing to maintain criticality over the full length of the cycle, and consequently to the full time it takes to achieve equilibrium. The cycle length and the shuffling scheme must be maintained as a part of the conversion, so to remedy this, the diameter of the fuel rods were increased from 0.51 cm to 0.70 cm. The new fuel compositions per fuel element are listed below for clarity on this adjustment in table 5. This is very common in many of these LEU fuel conversion studies, maintaining the amount of ^{235}U consistent between the HEU and LEU cores has not seen success yet without trading cycle length or shortening the shuffling scheme[4,16,18].

Table 5. Fuel Compositions in Equilibrium Core per Element

Fuel Type	HEU	Old LEU element	New LEU element
^{235}U(g)	350.00	350.00	483.88
^{238}U(g)	26.00	1426.65	1972.38
O(g)	68.00	0.00	0.00
Al(g)	625.00	0.00	0.00
Zr(g)	0.00	2134.03	2950.35
H(g)	0.00	37.43	51.74
Total mass(g)	1069.00	3948.11	5458.36
Density(g/cc)	3.76	11.71	11.71
Fuel volume(cc)	296.00	337.22	466.52

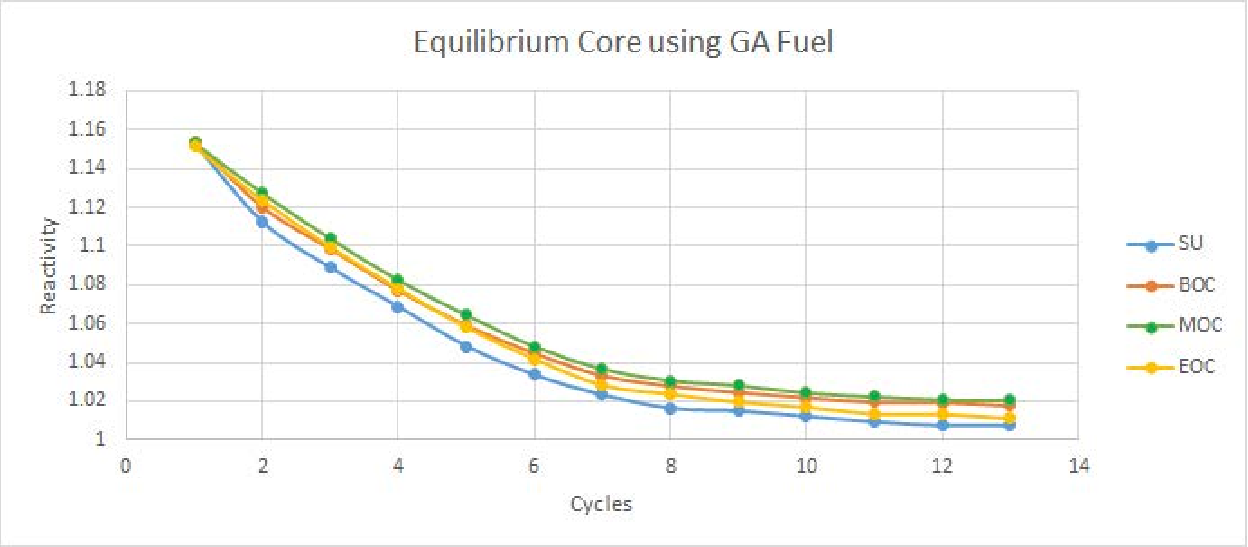


Figure 9. The k_{eff} changes along the iteration cycle number for all states in the equilibrium core search procedure.

As shown in Figure 9, the k_{eff} gradually approaches unity ($k_{\text{eff}} = 1$) for each of the states, bringing the core to a state of equilibrium. The calculation above took approximately 500 hours using 8 cores in Windows. The error bars are on the graph above, but with the error at ± 0.0006 they are currently not visible on the scale of the graph. Note the the SU k_{eff} is lower than the other states in the cycle. This may seem conceptually backwards, but this is due to the position of the control rods. The SU state may be slightly over moderated, explaining it's particularly low equilibrium $k_{\text{eff.}}$The temperatures used in the core were the nominal temperatures provided in the original input deck from NIST.

3.3 Actinide Build Up and Consumption

The burnup and buildup of some key actinides at the end of equilibrium cycle (using results obtained in the Cycle 12) are shown in Table 6.

Table 6. Burnup and production rates for fissionable actinides 8 cycles.

Fuel	^{235}U %	^{235}U (g)	^{238}U (g)	^{239}Pu (g)
TRIGA	59.50	207.10	21.00	8.38
U_3O_8	75.00	N/A	N/A	N/A

A post-processing code was developed in Python to extract the burned fuel inventories. Then using these fuel inventories the code sorts through for the element of interest and calculates the burned fuel. The burnup percent was calculated by taking the amount of uranium in the fresh fuel elements (fuel 8.1 and 7.1 in Fig. 4) and subtracting the amount of uranium in discharged fuel elements (fuel 8.8 and 7.7 in Fig. 4). The difference was then divided by the starting uranium to determine the percent of uranium burned. One interesting point to note from Table 3 is the comparison of the percent burnup between the HEU and TRIGA fuel. The TRIGA fuel nearly performed as well as the HEU in terms of efficiency, exceeding most typical LEU fuels. The closest competitor would be the U-10Mo fuel that achieved a 40% burnup in the new core, nearly 20% less burnup than the TRIGA fuel 16. . The TRIGA fuel produces on average 8.38 grams of 239Pu as shown in Table 3, this was determined by the same method as the 235U. Having a larger 239Pu production and 238U burned is very typical of LEU fuels as seen in earlier studies [15, 17], most likely due to the higher U238 content.

MCNP does not normalize the flux it generates, so in order to calculate the physical flux, the generated flux must be normalized. By using the NBSR's power, and assuming 200 MeV per fission with 2.44 neutrons per fission, we have determined a normalization factor of 1.523×10^{18} neutrons/s [7]. By incorporating the material inventories generated from the multicycle equilibrium core, key neutronics characteristics can be calculated using MCNP. Fast and thermal flux, power, neutron lifetimes, and shim worth are all calculated using these inventories following.

$$\begin{aligned}\text{Total source} &= (2.44\,\text{neutrons/fission})(20 \times 10^{6}\,\text{J/s})/ \\ &\quad [(200\,\text{MeV/fission})(1.602189 \\ &\quad \times 10^{-13}\,\text{J/MeV}] \\ &= 1.523 \times 10^{18}\,\text{neutrons/s}\end{aligned}$$

This factor is used here after to normalize all talleys to the reactors power to determine the absolute flux.

CHAPTER 4. NEUTRONICS PERFORMANCE CHARACTERISTICS

4.1 Flux Distributions

As mentioned earlier in the study, the flux of the reactor is absolutely necessary to properly determine the viability of the TRIGA fuel conversion. The flux is the key factor in the experimental studies currently being performed at NIST, and thus a thorough evaluation is needed. In calculating the flux for each of the 4 states, the 13th cycle of the equilibrium core was used. The material composition was extracted, and the FMESH tally was used to determine the flux. FMESH allow the user to imaginably break the 3D space into small tally bins following different geometric setting, and tally the flux for each of those bins. A right cylindrical geometry setting is adopt for the flux calculation as it is most close to a reactor shape. Using FMESH, the inner reactor was split into 50 radial segments, 80 z segments, and 50 theta, for a total mesh of 125,000 bins. The flux was also split into fast and thermal at 20 and 0.625E-6eV respectively. The radius was from 0 to 56cm, the theta from 0 to 2π, and the Z from -40.7 to 40.7, where the center of the reactor is at z=0. The coordinates and flux were then extracted using MATLAB and plotted on a mesh plot to show the reactor from an x-y view as well for each state. 36,000 particle histories and 2000 cycles were used to reduce the error on the flux to on average 1.3%.

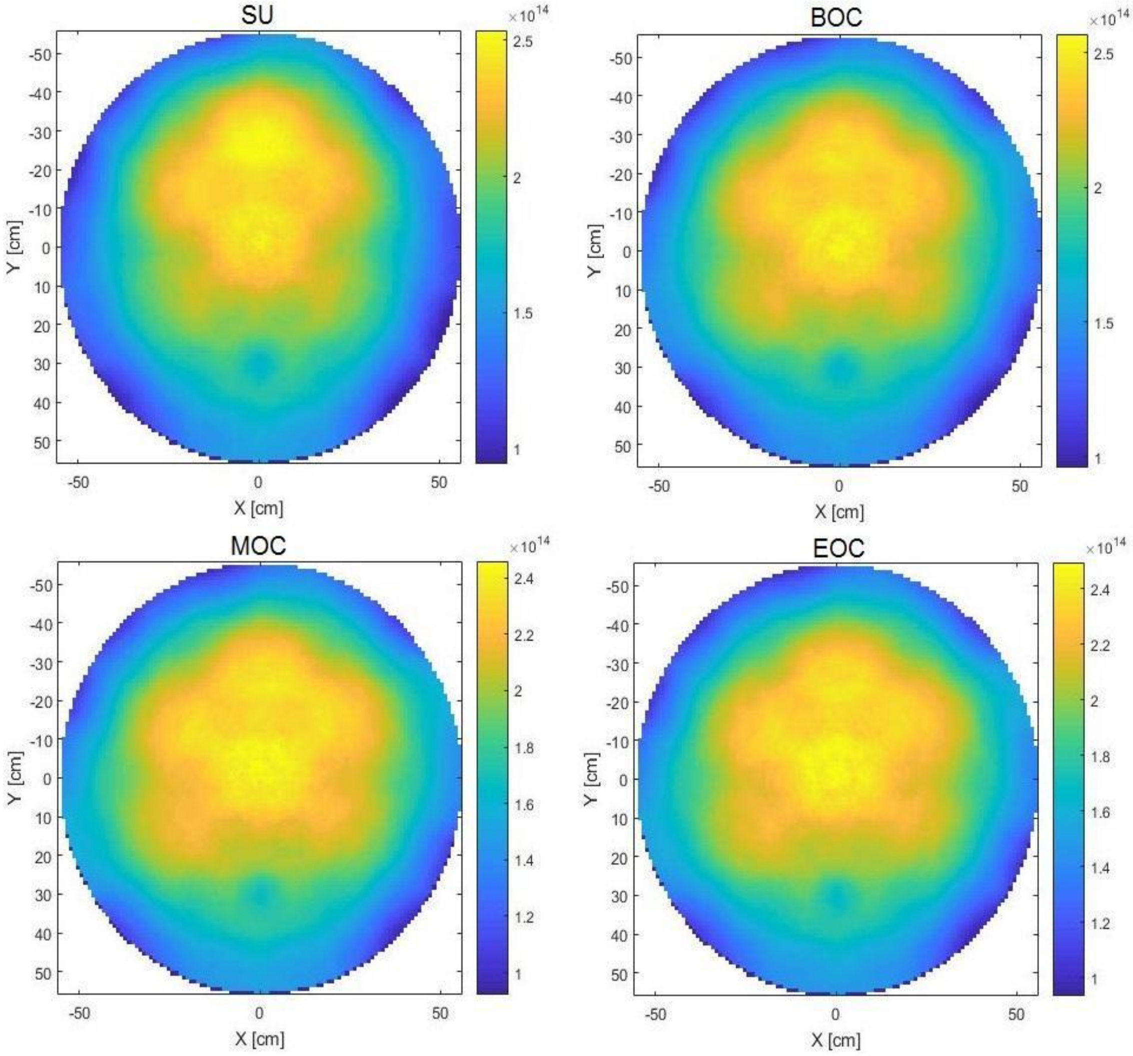


Figure 11. Radial Flux Distribution for core states for Z= 40 cm for thermal flux.

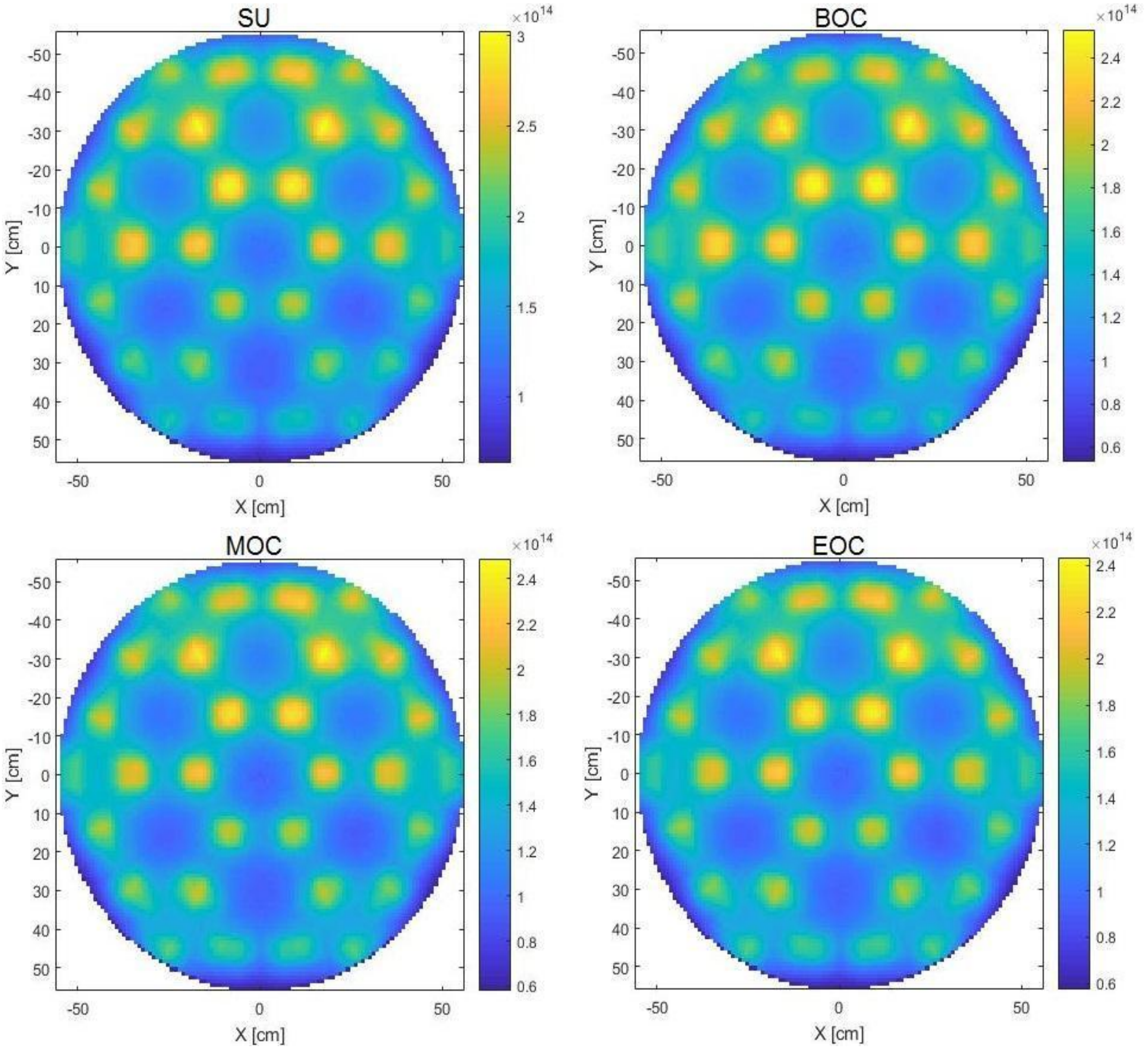


Figure 12. Radial Flux Distribution for core states for Z= 20 cm for fast flux.

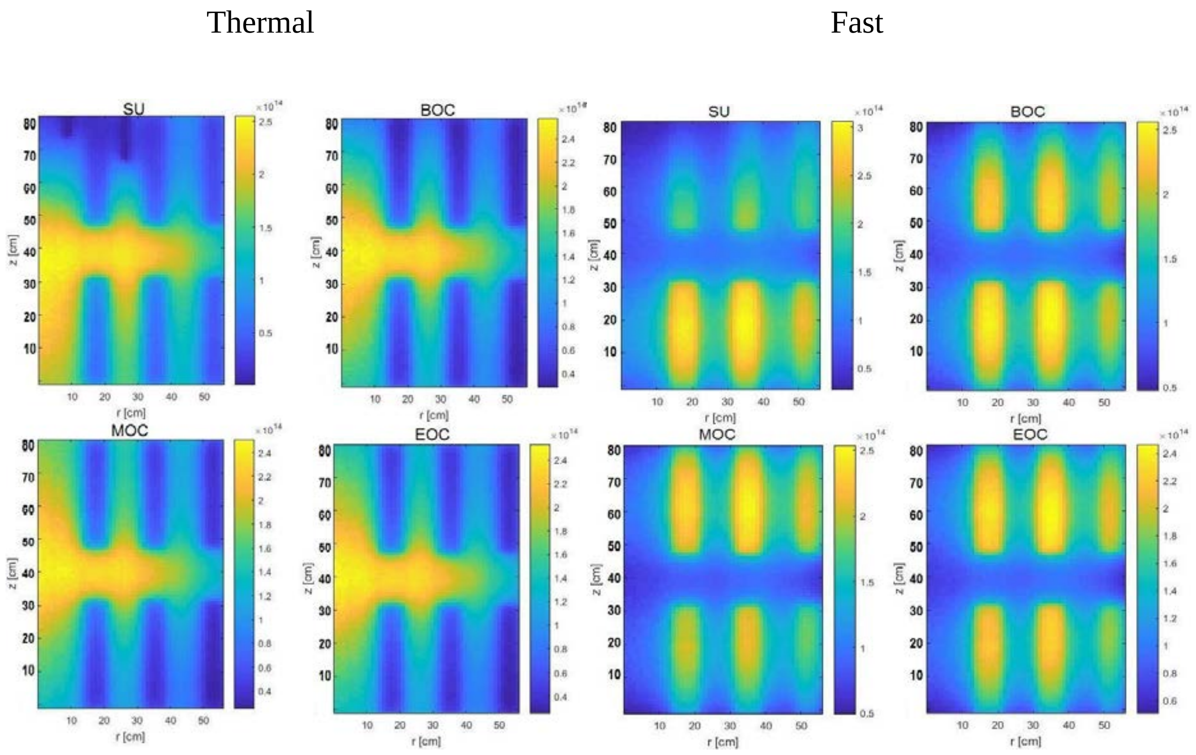


Figure 13. Axial flux distributions for fast and thermal flux at azimuthal $\theta = \pi$.

As the physical plots appear very similar, the flux is much higher in these spots in the beginning stages of the core. The thermal flux has a max at 2.5×10^{14} n/cm^2-s in the startup, trailing off over the course of the cycle to 2.4×10^{14} n/cm^2-s in the EOC state for the places of interest. The thermal flux reaches a max of 2.45×10^{14} n/cm^2-s in BOC and a minimum of 2.40×10^{14} n/cm^2-s in the EOC. This is of particular interest in reference to the current operating capacity (2.5×10^{14} n/cm^2-s) as the average flux must be of merit for its considered use in the NBSR. As a comparison to the current, the thermal flux would suffer a 5% when measured in reference to the center of the core. The areas of high flux activity in the fast flux plots are the fuel elements. Fission primarily being the source of fast flux, the fuel elements are highlighted by the fast flux plots. The thermal flux from slow neutrons primarily highlighting the heavy water in the reactor. Thermal neutron cross sections mean the heavy water will absorb the majority of that flux. As shown above, the flux is noticeably concentrated in the northern sector of the reactor. This is likely due to the location of the NBSR's cold neutron source located in between the middle northernmost elements 7.2L and 7.2R. This was by design, as flux concentrated near the cold neutron source (CNS) can be captured more easily for experimental purposes. Another worthy of merit comparison is how the flux changes with z, the height. In the figure shown above, the fast and thermal flux are plotted for each state to show the flux concentration as a function of height. The thermal flux is noticeably concentrated in the center, likely due to the amount of heavy water and lack of fuel. This is of more interest, as the states progress, it appears the flux changes from being concentrated in the lower half of the core, to the upper half of the core. This may be due to the retraction of the control elements throughout the cycle's timeline. The shims are retracted from the bottom up, creating an area of no moderation below the reactor, and normal moderation above the middle of the reactor. This is likely the cause of the migration of the flux from the bottom to the top of the reactor.

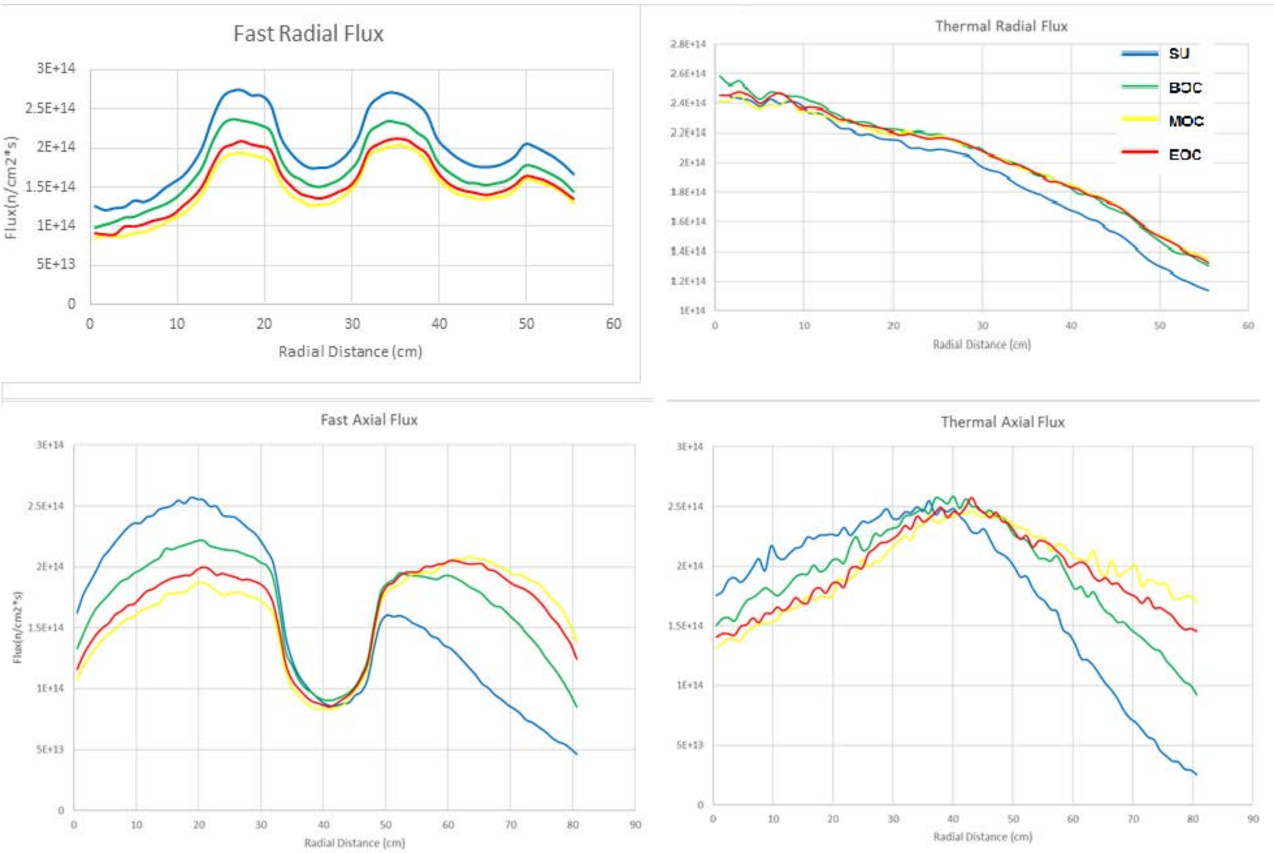


Figure 14. One-dimension flux distribution for all four states. Fast Z=20cm Thermal Z=40cm.

Above the radial flux for each of the 4 states are plotted linearly to better visualize the relationship with the radial distance. Note the fast flux peaking at 17.5, 35.0 cm, and 50 cm, this radius is the location of fuel elements in the core. The thermal flux peaks at the center of the core, with a slow drop off in all four states, likely because that again is the largest volume of heavy water in the core. The fast flux is clearly larger in the earlier stages because of the fresh fuel, but sinks to the thermal flux magnitude by the end of cycle. The larger amount of fissile material meaning more fissions and more fast flux. Another explanation for the larger fast flux in the early cycle, would be solely from the moderation. The cadmium shim arms (known for absorbing slow neutrons), absorb enough of the thermal flux that the fast flux needs to compensate. Over the course of the cycle there is less moderation and therefore less fast flux. Above the thermal and fast flux are plotted linearly in respect to height to better illustrate the relationship in all four states. The fast flux peaks at z = 20cm and z = 60 cm for all four states, most likely because this is the location of the center of the fuel rods. The fast flux has a minimum, and the thermal flux has a maximum at Z = 40cm for all four states. This is likely because of the gap between the fuel rods that is filled with heavy water. Graphed in color on the contour plot, as well as shown here the thermal flux will peak in places of high moderation. The thermal flux concentration at the center of the core is not ideal, but very typical of LEU fuels. Flux concentrated farther out radially is easier to capture and use by the instruments. In previous studies, U-10Mo and U-7Mo/Al performed similarly with flux peaking near the center, making it more difficult to harvest the neutrons. The flux peaking in the center axially is ideal, the neutron tubes are centered meaning it is easier axially for them to be harvested.

4.2 Power Distributions

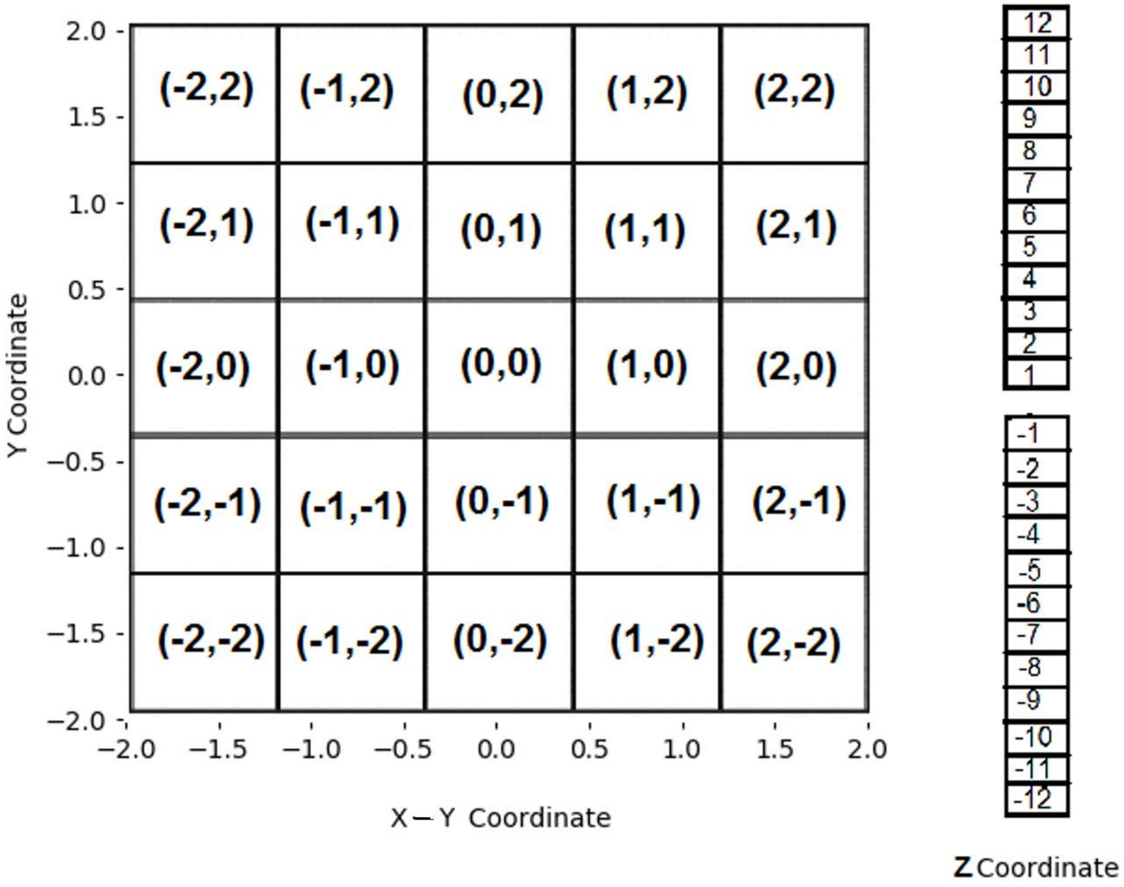
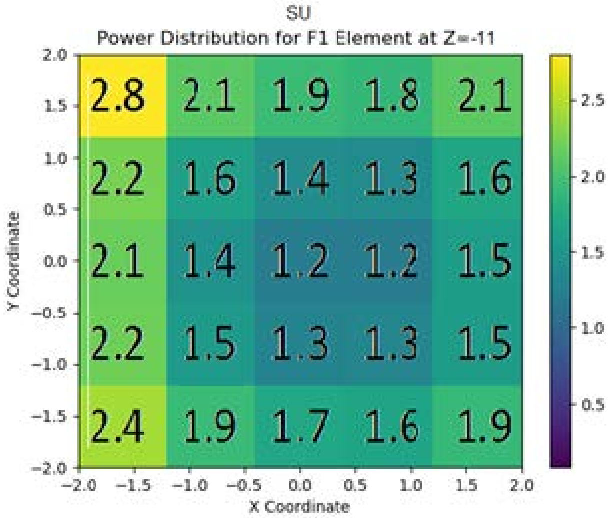
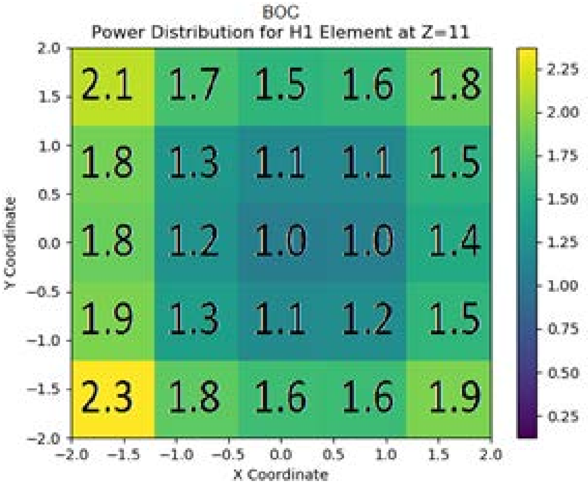


Figure 15: Labeling system for a fuel element for power calculations.

Calculating the power in the core is important to the safety to the reactor, hot spots need to be found and determined to properly evaluate the behavior of the fuel. The rods were each split into 12 axial sections, and with each rod evaluated separately throughout the reactor. This means the mesh for the power has a volume of 1.154cm^3 with a total of 18,000 meshes. The power distribution was calculated using MCNP6, with the table 128 method[18]**.** Table 128 is a summary of the neutrons that enter, collide, and fission in every cell. The fission number in a cell is directly related to the power, and can be determined by normalizing the number of fissions on average per cell. The 13th cycle of the equilibrium core was used to determine the power, the material compositions at each of the four states were extracted and the table 128 was produced with 36,000 particle histories along with 2,000 cycles unlike previous studies, but still too keep error below 100 pcm. Python 3.6 was used to extract the contents of the results, and to normalize and calculate the power. Python was also used to sort and graph the power distribution at the peaking factors shown below. The fission numbers were normalized by the average per cell (4,000 particles per bin at 18,000 bins).





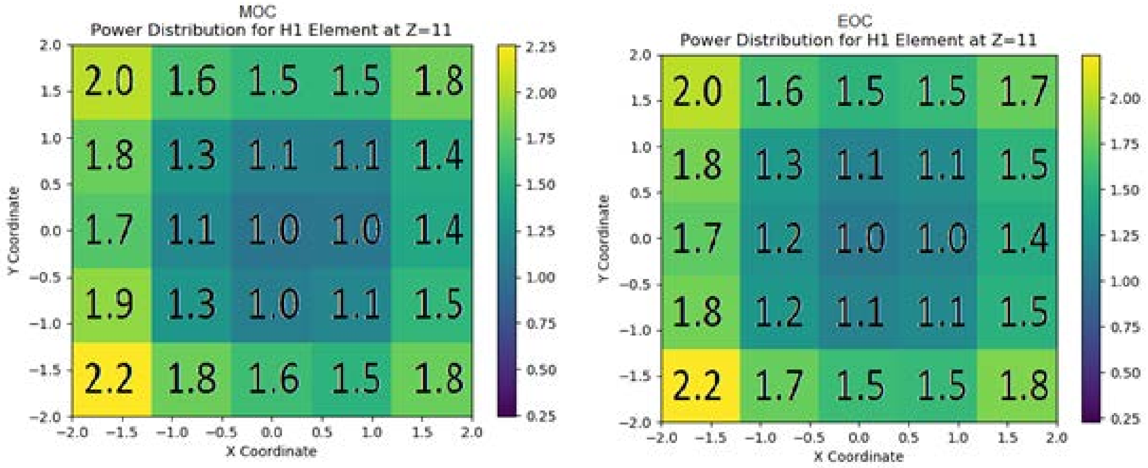


Figure 16. Power factors at the hottest element for each burnup state.

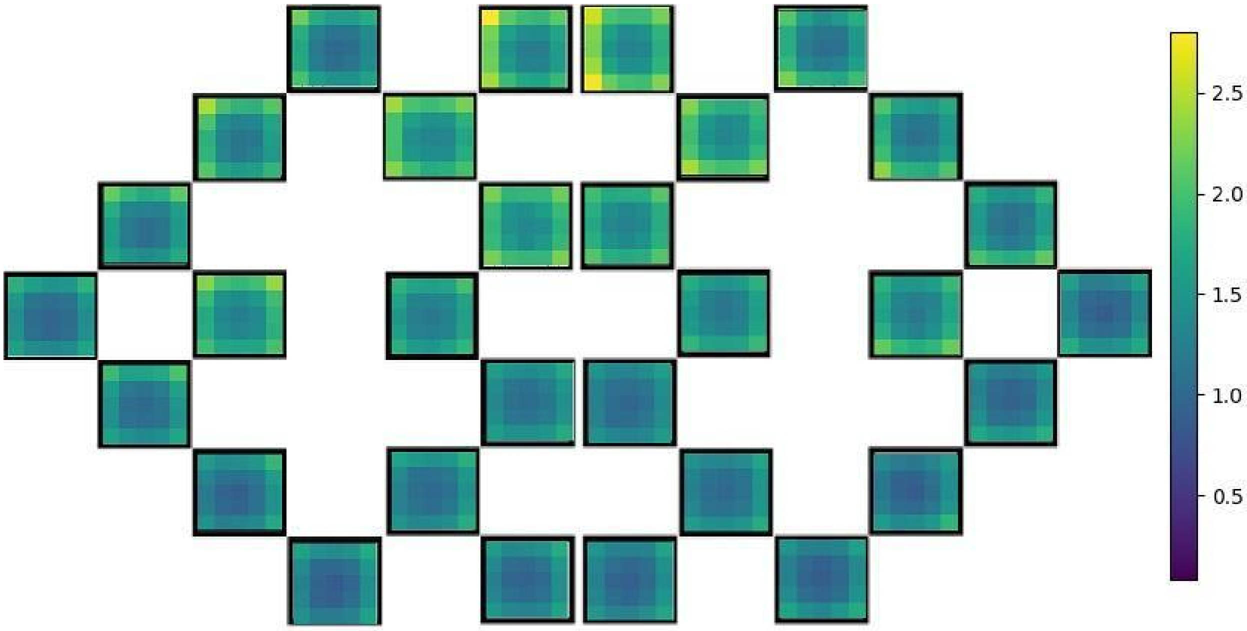
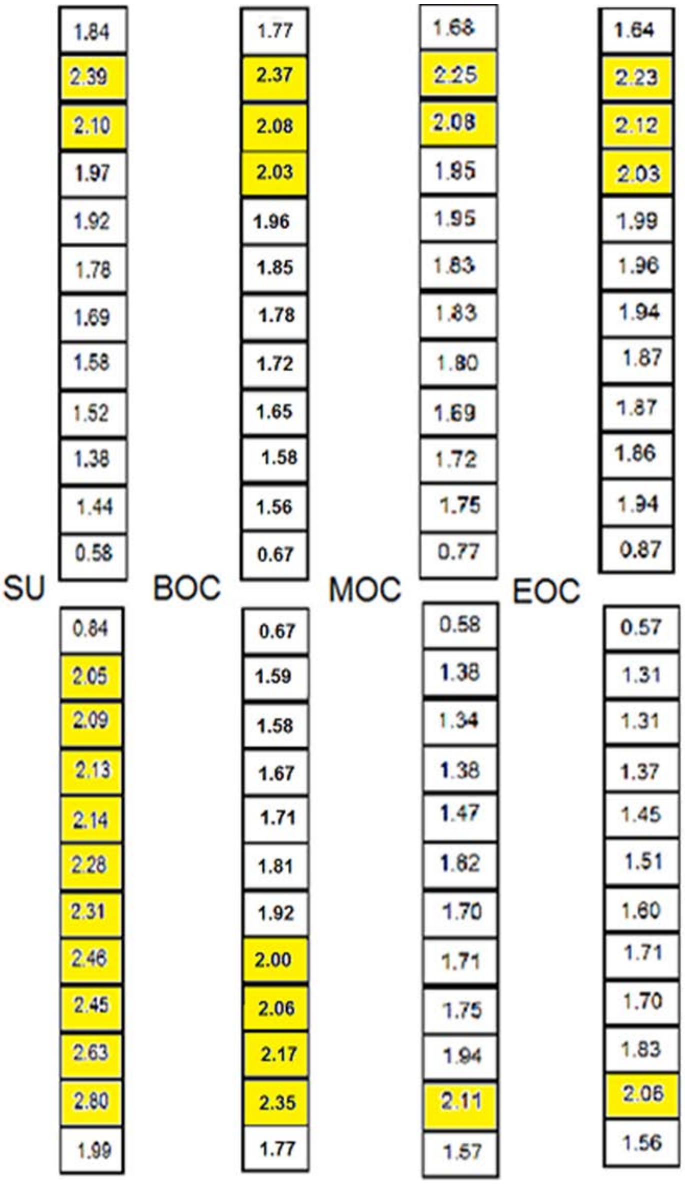


Figure 17. Full Core Power Distribution in SU state at Z=-11.

Table 7: Peaking power factors for different core states

Core State	SU	BOC	MOC	EOC
PPF	2.80	2.37	2.26	2.24
Z-position	-11	11	11	11
Fuel Rod (x,y)	(-2, 2)	(-2,-2)	(-2,-2)	(-2,-2)
Fuel Element	7.2L	7.2R	7.2R	7.2R

From Table 7, the first figure of merit are the peaking power factors for the differing states. When determining the safety of a fuel, a PPF of 2.5 or below for a state is considered an acceptable standard. The SU state is the only state that does not meet this standard, diverging by 10.7%. This is as expected, the SU state has the freshest fuel of the four states, and therefore should have the highest peaking power factor. Generally larger PPF are allowed at 0% power because the core is not producing heat. This would mean that even though the SU is hot, there is no threat to the cladding integrity.The Z position of the PPF is in reference to the center of the core. In the SU state, the power peaks in the lowest part of the reactor, while in the other three states, the power peaks in the highest section of the reactor. This is likely due to the control shims being drawn out over the course of the cycle. As the shims exit the lower core, the power shifts from the lower to upper core. An interesting note for the peaking power, after the SU state, the power peaks in the same fuel element, and the same rod, for the remainder of the cycle. The coordinates for the rod position are designate above in Figure 15, with the (0, 0) coordinate representing the middle rod in a fuel element. This can be seen more readily in Figure 17, as the power is clearly concentrated in the northern core. This is likely due to the beam tubes being located in the north left side. Unlike any other LEU fuel, the TRIGA fuel power appears to be concentrated to the top edges of the reactor. Generally in LEU fuels, such as U10Moly and U7Moly, the power tends to be concentrated towards the center of the reactor. This is a problem because the central based power decreases the amount of neutrons that enter the beam tubes[19]. Having the power concentrated at the edges of the reactor creates less of a hotspot and a safer reactor, as well as allowing the maximum amount of neutrons to the beam tubes.



SU	BOC	MOC	EOC
1.84	1.77	1.68	1.64
2.39	2.37	2.25	2.23
2.10	2.08	2.08	2.12
1.97	2.03	1.85	2.03
1.92	1.96	1.95	1.99
1.78	1.85	1.83	1.96
1.69	1.78	1.83	1.94
1.58	1.72	1.80	1.87
1.52	1.65	1.69	1.87
1.38	1.58	1.72	1.86
1.44	1.56	1.75	1.94
0.58	0.67	0.77	0.87
0.84	0.67	0.58	0.57
2.05	1.59	1.38	1.31
2.09	1.58	1.34	1.31
2.13	1.67	1.38	1.37
2.14	1.71	1.47	1.45
2.28	1.81	1.82	1.51
2.31	1.92	1.70	1.60
2.46	2.00	1.71	1.71
2.45	2.06	1.75	1.70
2.63	2.17	1.94	1.83
2.80	2.35	2.11	2.06
1.99	1.77	1.57	1.56

Figure 18: PPF for fuel rods at global max fuel each state. Note PPF above 2.0 are highlighted

Above in Figure 18, the hottest fuel rods for each cycles are shown with their peaking power factors at each Z. Note there are 12 sections per rod, with a space in the middle just as there is a space in the NBSR. The power appears to be centered in the bottom half of the reactor for the initial stages. SU has the hottest rod below, BOC is clearly comparable between the top and bottom. This evolves over time to the power moving upward for the MOC and EOC cycles. This is most likely due to the position of the shim arms in the reactor.

4.3 Neutron Lifetimes

The prompt neutron lifetime is defined as the average time between the generation of fission neutrons and when they are absorbed [4]. The neutron lifetime is particularly important when examining a fuel because it gives insight into the multiplicative process. A higher neutron lifetime means there is more time between emission and absorption giving the reactor operator more control over the reactor. The contrast of this the lower the lifetime the less control over the reactor. The MCNP KOPTS card was used to determine the lifetimes using the equilibrium material makeup for all four states, and the results are summarized in Table 8.

Table 8. Neutron lifetime (microseconds) at each state between HEU and LEU[4,20]

Fuel type	SU	BOC	MOC	EOC
TRIGA	519 ± 15	476 ± 11	494 ± 12	538 ± 13
U_3O_8	698	NA	NA	731

As shown in the table, the neutron lifetime for the TRIGA LEU fuel ranges from 550 to 650 microseconds, while a typical value of HEU for the heavy water reactor is ~700 microseconds.

4.4 Shim Arm Worth

Control shim worth allows an evaluation of the shims ability for reactor operation and safety management. The shim worth at different burnup state indicates what percent of the shims were used over the course of the cycle. The shim worth results for the TRIGA fuel is summarized in Table 9. The shim arm worth is calculated for all four cycles. The worth is calculated by determining the k_{eff} with all the shim arms in, as well as with the shim arms fully withdrawn. The shim worth is important as it allows insight into how much control the operator has over the core using the shims. Higher percent shim worth indicates more control and the reverse for less percent.

Table 9: Total Shim Arm Worth (Δk/k) Between LEU and HEU[4]

Fuel	SU	BOC	MOC	EOC
TRIGA	15.9	16.5	16.8	17.1
U_3O_8	24.9	NA	NA	27.2

As shown in the table, the control worth for the TRIGA fuel is lower than the HEU. This is as expected, as LEU fuel historically has shown to have lower shim worth [4]. The shims have lower worth, but this is still within the acceptable margin of use in the NBSR

CHAPTER 5. SUMMARY

In conclusion the initial feasibility study performed determined that the TRIGA fuel can be used in the NBSR fuel elements. It also determined that the optimal conditions for operation using the TRIGA fuel are using a 5x5 configuration, stainless steel 304 cladding, and the 45/20 fuel composition. The 5x5 configuration was likely due to the reduced self-shielding of the fuel from the smaller radius and high surface area that came with the more homogeneous positioning. The stainless steel 304 cladding was chosen because of its material makeup, less weight percent of Ni allowed the cladding to absorb less neutrons while still acting as a sturdy cladding. The fuel composition directly related to the amount of ^{235}U present, leading to the highest weight percent, the 45/20 fuel, to be the optimized choice. The results of this study were absolutely necessary for the determination of the equilibrium core and behavior of the fuel over time. Using the optimized fuel element, it was found that the TRIGA fuel can maintain reactivity, and achieve equilibrium by the 9th cycle, with a slight increase in ^{235}U for each element. The fuel was determined to have a burnup of nearly 60% in the 12 cycle, over 20% more burnup then its leading LEU competitors. This fuel make-up determined from the equilibrium core was necessary for the determination of related neutronics parameters. The peaking power factors for the core were found to be well below 2.5, well within the acceptable safety margin for the cores activity. The thermal flux was found to have a minimum of $2.45*10^{14}$ n/cm^2*s in the center of the core, a mere 5% loss in integrity from the NBSRs current operating capacity. The neutron lifetimes and shim worths were found to be comparable to other LEU fuels on the market. Given the results, the TRIGA fuel can be considered a very viable option, in comparison to its competitors. In the schema of LEU fuel, the TRIGA fuel has outperformed all others in the

NBSR. Keeping this in mind, there was still a loss of flux integrity for the NBSR. With this reduced flux, NIST will lose some of its experimental capabilities, an unacceptable result of the conversion. For future work, we may plan out the process of the conversion as in source [21] as well as [22]. The conversion would require a steady transition into the new fuel with a mixed core state, this would require extensive neutronics evaluation. Our group may also explore the effectiveness of the CNS when using the TRIGA fuel. CNS performance would give a clearer picture as to the fluxes behavior in the core. Another possibility for a future project would me an analysis of the fuels performance in the newly proposed core at NIST. The core was redesigned to be able to use LEU fuels more effectively, and using the TRIGA fuel in the new core may open up some interesting results. More safety analysis on temperature and power coefficients needs to be performed. The negative temperature coefficients means these would likely breed promising results but none the less they need to be calculated to be thorough.

BIBLIOGRAPHY

[1] Stahl D.,”Fuels for research and test reactors, status review”,July 1982 (ANL--83-5). United States.

[2] Stevens J., 2010, Technical challenges for conversion of US High-Performance Research Reactors (USHPRR), Argonne National Laboratory.

[3] Hanson A., and Diamond D., 2011, “Calculation of Design Parameters for an equilibrium LEU core in the NBSR,” Brookhaven National Laboratory ,September 30, 2011.

[4] Hanson A., and Diamond D., 2014, “Calculation of Design Parameters for an Equilibrium LEU core in the NBSR using a U7Mo Dispersion Fuel,” Brookhaven National Laboratory.

[5] Negut G., Mladin M., Prisecaru I., Danila N.,2006,”Fuel Behavior comparison for a research reactor”, Journal of Nuclear Materials 352(1-3):157-164,June 2006.

[6] Brown N.,Hanson A., and Diamond D, 2013”Local Burn-Up Effects in the NBSR Fuel Element”,Brookhaven National Laboratory.

[7] Simnad M.T.,1981,”The U-ZRHx alloy: its properties and use in TRIGA fuel”,Nuclear Engineering and Design, April 1981, Pages 403-422

[8] Lyons J., et al., 2014, “Reactor physics scoping and characterization study on implementation of TRIGA fuel in the Advanced Test Reactor”, Nuclear Technology, Vol. 189,202 – 217

[9] Dunn F., Wilson E., Feldman E., Sun K., Wang C., Hu L.,2017, “Evaluation of a uranium zirconium hydride fuel rod option for conversion of the MIT Research Reactor(MITR) from highly enriched uranium to low enriched uranium”, Argonne National Laboratory

[10] Jose A.,Sarta F.,Castiblanco L.,2005,”Neutronic calculations in core conversion of the IAN-R1 Research reactor from MTR HEU to TRIGA LEU fuel”,2005.

[11] “MCNP- A General Monte Carlo N-Particle Transport Code, Version 6.1.1,” LA-UR-03-1987, Los Alamos National Laboratory, August 1, 2017.

[12] Olander D., Greenspan E., Hans D.,Bojan P.,2009,”Uranium-zirconium hydride fuel properties”, Nuclear Engineering and Design 239(8):1406-1424 , August 2009.

[13] Bowman R., Craft B., Cantrell J., Venturini, E., 1985, “Effects of thermal treatments on the lattice properties and electronic structure of ZrHx,” Physical Review B. 31 (9): 5604–5615.

[14] McConn R. J. Jr. et al., 2011 “Compendium of Material Compositions Data for Radiation Transport Modeling,” PNNL-15870 Rev. 1, Pacific Northwest National Laboratory.

[15] Britton K., Wu Z.,”A neutronics feasibility study of the TRIGA LEU fuel in the 20mwt nist research reactor”,2018 26th International Conference on Nuclear Engineering,July 22–26, 2018.

[16] Turkoglu D., Wu Z., Williams R., Newton T.,”Comparison of neutronics performance characteristics of the proposed Nist reactor with different LEU fuels”

[17] Fensin M., et al.,”The New MCNP6 Depletion Capability”,LA-UR-12-22314 (2012).

[18] Wu Z., Williams R., Rowe J., Newton T., and O'Kelly S., 2017. “Neutronics and Safety Studies on a Research Reactor Concept for an Advanced Neutron Source,” Nuclear Technology, 199(1), 67-82.

[19] Hanson A., and Diamond D.,2011,”A neutronics methodology for the NIST research reactor based on mcnpx”,International Conference on Nuclear Engineering Chiba, Japan May 16-19, 2011.

[20] Hanson A., Ludewig H.,Diamond D.,2005,”Calculation of the prompt neutron lifetime in the NBSR.”,Brookhaven National Laboratory, September 2005.

[21] Hanson A., Diamond D., 2011, “Planning the HEU to LEU transition for the NBSR”, Brookhaven National Laboratory.

[22] Bretscher M., Snelgrove J.,1991,”Transition from HEU to LEU fuel in Romania's 14-MW TRIGA Reactor”, XIV International Meeting on Reduced Enrichment For Research and Test Reactors, November 4-7, 1991.

[23] Hanson A., Diamond D.,2010,”Calculation of inventories, power distributions and neutronic parameters for the NBSR using MCNPX.”, TRTR/IGORR Joint meeting, Knoxville, Tennessee,Sept 19-23.

[24] Wu Z., Williams R.,2015,”Reactor Power Distribution Calculation in Research Reactors Using MCNP”,Transactions of the American Nuclear Society, Vol. 113, Washington, D.C., November 8–12.

[25] Snoj L., Matjaz R.,2006,”Calculation of power density with mcnp in TRIGA reactor”, International Conference Nuclear Energy for New Europe 2006 Portorož, Slovenia, September 18-21, 2006.

[26] Wu Z., and Williams R. E., 2015 “A Fast and Self Consistent Approach for Multi-cycle Equilibrium Core Studies Using Monte Carlo Models,” the Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method, Nashville, TN, April 19-23, on CD ROM.

[27] Wu Z. et al., 2017, “Neutronics and Safety Studies on a Research Reactor Concept for an Advanced Neutron Source,” Nuclear Technology, 199(1), 67-82

[28] KOPETKA P., WILLIAMS R. E., and ROWE J. M., 2008, “NIST Liquid Hydrogen Cold Source,” NIST Internal Report, NISTIR-7352

[29] “Generic Procedures for Response to a Nuclear or Radiological emergency at Research Reactors-Training Material,” International Atomic Agency (2011).

[30] Brewer R., 2009, “Criticality Calculations with MCNP5: a Primer,” Los Alamos National Laboratory.

Appendices

UZRH rod input deck for MCNP

```
c    U=70 IS THE D-7 ELEMENT
2200   0 -940 939 -942 941 -922 920 imp:n,p=1 fill=200 u=70 $ Top Fuel Section
2230   3  -1.0977 -934 933 -946 945 -924 925 imp:n,p=1 u=200 lat=1 $ lat cell 3x3 and 5x5
c 2230   3  -1.0977 934 -933 946 -945 -924 925 imp:n,p=1 u=200 lat=1 $ lat cell 4x4 and 6x6
c                    fill= 240
        fill=-2:2 -2:2 0:0  240 24r
c        *******  TOP FUEL RODS   ******
c                    Fuel types
c Fuel Split into 12 sections for power
10000  1162  -11.708  -4799 -4999 5001 imp:n,p=1 vol=0.562  u=240
10001  1162  -11.708  -4799 -5001 5002 imp:n,p=1 vol=0.562  u=240
10002  1162  -11.708  -4799 -5002 5003 imp:n,p=1 vol=0.562  u=240
10003  1162  -11.708  -4799 -5003 5004 imp:n,p=1 vol=0.562  u=240
10004  2162  -11.708  -4799 -5004 5005 imp:n,p=1 vol=0.562  u=240
10005  2162  -11.708  -4799 -5005 5006 imp:n,p=1 vol=0.562  u=240
10006  2162  -11.708  -4799 -5006 5007 imp:n,p=1 vol=0.562  u=240
10007  2162  -11.708  -4799 -5007 5008 imp:n,p=1 vol=0.562  u=240
10008  3162  -11.708  -4799 -5008 5009 imp:n,p=1 vol=0.562  u=240
10009  3162  -11.708  -4799 -5009 5010 imp:n,p=1 vol=0.562  u=240
10010  3162  -11.708  -4799 -5010 5011 imp:n,p=1 vol=0.562  u=240
10011  3162  -11.708  -4799 -5011 5012 imp:n,p=1 vol=0.562  u=240
c
c                cladding
2300   290  -8.0 -4899 4799 -4999 5012 imp:n,p=1 u=240      $ cladding ss-304
2330   3  -1.0977 4899    imp:n,p=1  u=240  vol=55.98958 $ d2o vol=344.407
c
c
1200   0 -940 939 -942 941 -921 923 imp:n,p=1 fill=100 u=70 $ Bottom Fuel Section
1230   3  -1.0977 -934 933 -946 945 -929 930 imp:n,p=1 u=100 lat=1 $ lat cell 3x3 and 5x5
c 1230   3  -1.0977 934 -933 946 -945 -929 930 imp:n,p=1 u=100 lat=1 $ lat cell 4x4 and 6x6
c                     fill= 140
        fill=-2:2 -2:2 0:0  140 24r
c        *******  BOTTOM FUEL RODS   ******
c                   Fuel Types
c Fuel Split into 12 sections for power
20000  4162  -11.708  -4799 5999 -6001 imp:n,p=1 vol=0.562 u=140
20001  4162  -11.708  -4799 6001 -6002 imp:n,p=1 vol=0.562 u=140
20002  4162  -11.708  -4799 6002 -6003 imp:n,p=1 vol=0.562 u=140
20003  4162  -11.708  -4799 6003 -6004 imp:n,p=1 vol=0.562 u=140
20004  5162  -11.708  -4799 6004 -6005 imp:n,p=1 vol=0.562 u=140
20005  5162  -11.708  -4799 6005 -6006 imp:n,p=1 vol=0.562 u=140
20006  5162  -11.708  -4799 6006 -6007 imp:n,p=1 vol=0.562 u=140
20007  5162  -11.708  -4799 6007 -6008 imp:n,p=1 vol=0.562 u=140
```

```
20008  6162  -11.708  -4799 6008 -6009 imp:n,p=1 vol=0.562 u=140
20009  6162  -11.708  -4799 6009 -6010 imp:n,p=1 vol=0.562 u=140
20010  6162  -11.708  -4799 6010 -6011 imp:n,p=1 vol=0.562 u=140
20011  6162  -11.708  -4799 6011 -6012 imp:n,p=1 vol=0.562 u=140
c                    Cladding and D20
1300   290  -8.0 -4899 4799 -6012 5999 imp:n,p=1 u=140      $ cladding ss-304
1330   3  -1.0977 4899     imp:n,p=1 vol=55.98958 u=140                $ d2o
c
c

c surfaces
921  pz  -7.62   $window for bottom rods
923  pz -40.62      $window for bottom rods
927  px  -3.8964
928  px   3.8964
929  pz  -7.50      $lattice for bottom rods
930  pz -40.70     $lattice for bottom rods
c
c
920   pz   7.62    $window surface top of rods
922   pz  40.62    $window surface top of rods
939   py  -3.3325  $window y
940   py   3.3325     $window y
941   px  -3.7313    $window x
942   px   3.7313       $window x
945   py   -0.6665   $lattice in y 5x5
946   py    0.6665   $lattice in y 5x5
924   pz    40.70     $lattice
925   pz    7.50     $lattice
c ROD DIMENSIONS
c
4799   cz 0.35 $ Fuel meat radius 3x3 5x5
4899   cz 0.39 $ Radius with cladding 3x3 5x5
c
c
4999  pz 42.5  $ Top of rod
5001  pz 39.5
5002  pz 36.5
5003  pz 33.5
5004  pz 30.5
5005  pz 27.5
5006  pz 24.5
5007  pz 21.5
5008  pz 18.5
5009  pz 15.5
5010  pz 12.5
```

```
5011  pz  9.5
5012  pz  6.5    $ bottom of the rod
c              _
c BOTTOM RODS
c
5999  pz -42.5  $ bottom of bottom
6001  pz -39.5
6002  pz -36.5
6003  pz -33.5
6004  pz -30.5
6005  pz -27.5
6006  pz -24.5
6007  pz -21.5
6008  pz -18.5
6009  pz -15.5
6010  pz -12.5
6011  pz  -9.5
6012  pz  -6.5    $ Top of bottom

c Data
kcode  36000 1 10 2000
m1152                         $
     92235 -0.08865        $ U-235
     92238 -0.36135        $ U-238
     1001  -0.00948        $ H
     40091 -0.54052        $ Zrc
m2152                         $
     92235 -0.08865        $ U-235
     92238 -0.36135        $ U-238
     1001  -0.00948        $ H
     40091 -0.54052        $ Zrc
m3152                         $
     92235 -0.08865        $ U-235
     92238 -0.36135        $ U-238
     1001  -0.00948        $ H
     40091 -0.54052        $ Zrc
m4152                         $
     92235 -0.08865        $ U-235
     92238 -0.36135        $ U-238
     1001  -0.00948        $ H
     40091 -0.54052        $ Zrc
m5152                         $
     92235 -0.08865        $ U-235
     92238 -0.36135        $ U-238
     1001  -0.00948        $ H
     40091 -0.54052        $ Zrc
```

```
m6152                    $
      92235 -0.08865         $ U-235
      92238 -0.36135         $ U-238
      1001  -0.00948         $ H
      40091 -0.54052         $ Zrc
```

Python code for equilibrium core

```
import os
import sys
import datetime as dt
import fnmatch
import traceback, sys
import fileinput
import numpy as np
import subprocess
from os import remove
from shutil import move


class EquilCore():


    def __init__(self):

        self.user_specified_parameters()

        self.setup()

    def user_specified_parameters(self):
        self.fuel_type = ['UZrH'][0]
        #self.fuel_type = ['NBSR'][0]

        self.model_name = 'UZrH_EQ'
        #self.model_name = 'HEU_EQ'

        self.number_of_cycles = 14

        self.mat_str = '*'*4

        self.transverse_zones = False

        self.new_fuel = 9999
        self.fuel_MATs=[1151,1152,1153,1154,1155,1156,1157,1161,1162,1163,1164,1165
                       ,1166,1167,1168,1171,1172,1173,1174,1175,1176,1177,1181,1182,1183
                       ,1184,1185,1186,1187,1188,2151,2152,2153,2154,2155,2156,2157,2161
                       ,2162,2163,2164,2165,2166,2167,2168,2171,2172,2173,2174,2175,2176
                       ,2177,2181,2182,2183,2184,2185,2186,2187,2188,3151,3152,3153,3154
                       ,3155,3156,3157,3161,3162,3163,3164,3165,3166,3167,3168,3171,3172
                       ,3173,3174,3175,3176,3177,3181,3182,3183,3184,3185,3186,3187,3188
                       ,4151,4152,4153,4154,4155,4156,4157,4161,4162,4163,4164,4165,4166
                       ,4167,4168,4171,4172,4173,4174,4175,4176,4177,4181,4182,4183,4184
                       ,4185,4186,4187,4188,5151,5152,5153,5154,5155,5156,5157,5161,5162
                       ,5163,5164,5165,5166,5167,5168,5171,5172,5173,5174,5175,5176,5177
                       ,5181,5182,5183,5184,5185,5186,5187,5188,6151,6152,6153,6154,6155
                       ,6156,6157,6161,6162,6163,6164,6165,6166,6167,6168,6171,6172,6173
                       ,6174,6175,6176,6177,6181,6182,6183,6184,6185,6186,6187,6188 ]



        # fuel management scheme
        self.fuel_MATs_shuffle = {}
        for fuel_MAT in self.fuel_MATs:
            cycle_number = str(fuel_MAT)[-1]
            if cycle_number == '1':
                self.fuel_MATs_shuffle[fuel_MAT] = self.new_fuel
            else:
                self.fuel_MATs_shuffle[fuel_MAT] = fuel_MAT-1



    def setup(self):

        self.delete_extra_mcnp_files()

        self.total_mass = {'UZrH':658.0176992,'NBSR':178.171291}[self.fuel_type] # g, fuel
mass/axial zone- found in mcnp calculation


        if not self.restart:
            self.start_cycle_number = 1
            self.today = dt.datetime.today()
            self.run = '%d-%02d-%02d_%s_Run__1' % (self.today.year, self.today.month,
self.today.day, self.fuel_type)
            if 'runs' not in os.listdir('.'):
```

```
        os.mkdir('runs')
    else:
        while self.run in os.listdir('./runs/'):
            self.run = self.run.split('__')[0] + '__' + str(int(self.run.split('__')[1]) + 1)
    self.runPath = './runs/' + self.run + '/'
    os.mkdir(self.runPath)
    self.logfile = self.runPath + 'logfile.txt'
    os.system('touch "%s"' % self.logfile)
    os.mkdir(self.runPath+'/cycles')
    os.mkdir(self.runPath+'/iterations')


    os.system('cp "%s" "%s"' % ( os.path.basename(__file__), self.runPath ))
    os.system('cp "%s" "%s"' % ( self.model_name, self.runPath )) #
else:
    self.logfile = self.runPath + 'logfile.txt'


self.delete_extra_mcnp_files()
self.refuel = False
self.current_state = 'SU'
self.model_names = {'SU':self.model_name+ '_SU' +"_cycle",
                    'BOC':self.model_name+'_BOC'+"_cycle",
                    'MOC':self.model_name+'_MOC'+"_cycle",
                    'EOC':self.model_name+'_EOC'+"_cycle",}

self.connect_cycles = {'BOC':'SU', 'MOC':'BOC', 'EOC':'MOC'}


self.major_isotopes = [['92235','U-235' ],
                 ['92238','U-238' ],
                 ['94239','Pu-239'],
                 ['94241','Pu-241'],
                 ['93239','Np-239'],
                 ['53135','I-135' ],
                 ['54135','Xe-135'],
                 ['61149','Pm-149'],
                 ['62149','Sm-149']]
self.major_isotopes += {'UZrH':[['1001', 'Hydrogen'],
                ['40091', 'Ziconium']],
                'U7Mo13Al':[['13027','Al-027'],
                 ['42092', 'Mo-92'],
                 ['42094', 'Mo-94'],
                         ['42095', 'Mo-95'],
                         ['42096', 'Mo-96'],
                         ['42097', 'Mo-97'],
                         ['42098', 'Mo-98'],
                         ['42100', 'Mo-100']],
                        'NBSR':[['8016','O'],
                         ['13027','Al-027']]}[self.fuel_type]


        if not self.restart:
            self.write_to_logfile( "Constants and parameters:\n" )

        self.write_to_logfile( "    Fuel (Plates/FE): %s " % (self.fuel_type) )
        self.write_to_logfile( "    Model name : %s "  % (self.model_name ))





    def write_to_logfile(self, string_to_add):
        subprocess.check_output('echo "%s" >> "%s"' % (string_to_add,
self.logfile),shell=True).decode()


    def cat_to_inputfile(self, file1, file2):
        subprocess.check_output( 'type "%s" >> "%s" ' % (file1, file2),shell=True).decode()

    def append_to_inputfile(self, string_to_add, filename):
        subprocess.check_output('echo "%s" >> "%s"' % (string_to_add, filename)
,shell=True).decode()

    def move_files(self, files, path):
        for file in files:
            subprocess.check_output('mv "%s" "%s"' % (file, path),shell=True).decode()

    def create_file(self, filename):
        subprocess.check_output('type NUL > ' + filename,shell=True).decode()

    def delete_extra_mcnp_files(self):
        files = os.listdir('.')
        for file in files:
            if (len(file) == 4) and (file[:3] == 'out'):
                os.remove(file)
            elif (len(file) == 5) and (file[:3] == 'src'):
                os.remove(file)
            elif (len(file) == 6) and (file[:3] == 'run'):
                os.remove(file)
            elif (len(file) == 6) and (file[:3] == 'com'):
                os.remove(file)


    def run_mcnp(self, input_name):
        self.write_to_logfile("    Running MCNP6 with input: %s" % input_name)
        subprocess.check_output('mcnp6 i="%s" o="%s.o" tasks 8 ' % (input_name,
input_name.split('.')[0]), shell=True).decode()


    def get_keff_from_mcnp_output(self, output_file):
        get_keff = False
        found_keff = False
        with open(output_file) as f:
            for line in f:
                if found_keff:
                    continue
                else:
                    if line.startswith(" the estimated average keffs"):
                        get_keff = True
                    elif get_keff and line.startswith("       col/abs/trk len"):
                        keff, keff_unc = float(line.split()[2]), float(line.split()[3])
                        found_keff = True
        return keff, keff_unc




    def substitute_parameters(self, input_model, parameters,run_type='burn'):
        f = fileinput.FileInput(input_model, inplace=True)

        for line in f:
            if run_type == 'kcode' and '$x1' in line:
                continue
            elif run_type == 'burn' and '$x2' in line:
                continue
            line_printed = False
            for key, value in parameters.items():
                if key in line:
                    if type(value) is not type({}):
                        print(line.replace(key, str(value))),
                    else:
                        print(line.replace(key, str(value[self.current_state]))),
                    line_printed = True
            if not line_printed:
                print(line),
        f.close()



    def read_mat_composition_from_tally(self, filepath):

        self.new_compositions = {}

        begin_reading_mat = False
        if self.current_state == 'EOC':
            step = 2
        else:
            step = 1
        self.write_to_logfile( "    Processing output file: %s" % filepath )
        skip_lines = 1000
        get_keff = False
        with open(filepath) as f:
            line_number = 0
            for line in f:
                line_number += 1
                if skip_lines > 0:
                    skip_lines -= 1
                else:
                    if fnmatch.fnmatch(line[:54], ' actinide inventory for material'+self.mat_str+' at end
of step  %d' % step):
                        mat = int(line[32:36].strip())
                        self.new_compositions[mat] = {'totals': 0, 'zaids' : {}}
                        begin_reading_mat = True
                        skip_lines = 3

                    elif fnmatch.fnmatch(line[:57], ' nonactinide inventory for material'+self.mat_str+' at
end of step  %d' % step):
                        mat = int(line[35:39].strip())
                        begin_reading_mat = True
                        skip_lines = 3
                    elif begin_reading_mat:
                        line_split = line.split()
                        num = line_split[0]
                        zaid = line_split[1]
                        mass = float(line_split[2])
                        if num == 'totals':
                            begin_reading_mat = False
                            self.new_compositions[mat]['totals'] += float(line_split[1])

                        elif mass > 0.0: # mass is greater than 0
                            self.new_compositions[mat]['zaids'][zaid] = mass# add zaid and mass to
dictionary for mat
                    elif line.startswith(" the estimated average keffs"):
                        get_keff = True
                    elif get_keff:
                        if line.startswith("       col/abs/trk len"):
                            keff, keff_unc = float(line.split()[2]), float(line.split()[3])
                            subprocess.check_output('echo "    keff:  %.6f +/- %.6f \n" >> "%s"' % ( keff,
keff_unc, self.logfile ), shell=True).decode()
                            get_keff = False
        total_mass_in_core = 0.
        for mat, zaids in self.new_compositions.items():
            total_mass_in_core += zaids['totals']
        self.write_to_logfile("        Total mass in core at end of step: %f g" % total_mass_in_core )
        print("Successfully read tally data!")

    def add_fresh_fuel(self, MAT):

        self.fresh_fuel = "c\
                     \nm%d                  $ \
                     \n      92235 -0.08865       $ U-235\
                     \n      92238 -0.36135       $ U-238\
                     \n      1001  -0.00948       $ H\
                     \n      40091 -0.54052       $ Zr" % MAT
        #os.system('echo "%s" >> "%s"' % (self.fresh_fuel, self.inputfile_name) )
        with open(self.inputfile_name, 'a') as the_file:
            the_file.write(self.fresh_fuel)

    def New_fresh_fuel(self, MAT):

        self.fresh_fuel = "c\
                     \nm%d                  $ \
                     \n      92235 -0.08865       $ U-235\
                        \n      92238 -0.36135          $ U-238\
                        \n      1001  -0.00948          $ H\
                        \n      40091 -0.54052          $ Zr" % MAT
        #os.system('echo "%s" >> "%s"' % (self.fresh_fuel, self.inputfile_name) )
        with open(self.inputfile_name, 'a') as the_file:
            the_file.write(self.fresh_fuel)

    def add_used_fuel(self, MAT, new_MAT):
        total_mass = self.new_compositions[new_MAT]['totals']
        nominal_total_mass = self.total_mass

        self.updated_mat = "c\nm%d" % MAT
        total_mass_fraction = 0.0
        for isotope in self.major_isotopes:
            try:
                mass_fraction =
self.new_compositions[new_MAT]['zaids'].pop(isotope[0])/nominal_total_mass
                mass_fraction=float(format(mass_fraction,"2.2E"))
                total_mass_fraction += mass_fraction
                if mass_fraction > 0.99e-10:
                    self.updated_mat += '\n       %s  -%.10f      $ %s' % (isotope[0], mass_fraction,
isotope[1])

            except Exception as e:
                print(isotope[1] + " not in MAT %d" % new_MAT, isotope[0] not in
self.new_compositions[new_MAT]['zaids'])

                continue
        for zaid, mass in self.new_compositions[new_MAT]['zaids'].items():
            mass_fraction = mass/nominal_total_mass
            mass_fraction=float(format(mass_fraction,"2.2E"))
            if zaid == '83209':
                continue
            elif mass_fraction > 0.999e-10:
                self.updated_mat += "\n       %s  -%.10f      $" % (zaid, mass_fraction)
            total_mass_fraction += mass_fraction
        #print("Mass fraction for MAT %s is %.10f") % (new_MAT, total_mass_fraction)
        if total_mass_fraction < 1.0:
            #print("Mass fraction of Bi-209 added as filler: %.10f" % (1-total_mass_fraction))
            self.updated_mat += "\n       83209  -%.10f      $ Bi-209 filler" % (1.0-
total_mass_fraction)
```

```
        with open(self.inputfile_name, 'a') as the_file:
            the_file.write(self.updated_mat)



    def update_parameters(self, shim_position):

        self.parameters_to_substitute_in_input_file = {'{shim_angle1}':  shim_position,
                                                  '{shim_angle2}':  90+shim_position,
                                                  '{shim_angle3}':  90-shim_position,
                                                  '{burn_time}':{'SU':'1.5, 7.','BOC':'18.5, 7.','MOC':'18.5,
7.','EOC':'0.01, 14.'}}


    def adjust_shim_positions(self, model_name):
        if self.current_state == 'SU':
            shim_position=23.0
        elif self.current_state == 'BOC':
            shim_position=14.0
        elif self.current_state == 'MOC':
            shim_position=7.0
        elif self.current_state == 'EOC':
            shim_position=0

        self.update_parameters(shim_position)


    def write_next_input_file(self, model_name, inputfile_name):

        print("Writing input file name: %s" % inputfile_name)
        self.create_file(inputfile_name)
        self.cat_to_inputfile(model_name, inputfile_name)



        if self.current_state == 'SU' and self.current_cycle==1:
            print('Loading all new fuel')
            for MAT in self.fuel_MATs:
                self.New_fresh_fuel(MAT)
        if self.current_state == 'SU' and self.current_cycle==1:
            return

        if self.current_state == 'SU' and self.current_cycle>1:
           self.refuel=True
        if self.refuel:
            print("Refueling (add fresh fuel and shuffle)")
            for MAT in self.fuel_MATs:
                new_MAT = self.fuel_MATs_shuffle[MAT]
                if new_MAT == 9999:
                    self.add_fresh_fuel(MAT)
                else:
                    self.add_used_fuel(MAT, new_MAT)
            self.refuel = False

        elif not self.refuel:
            print("Using existing fuel.")
            for MAT in self.fuel_MATs:
                self.add_used_fuel(MAT, MAT)
        print("Successfully wrote input file " + inputfile_name)

    def go(self):

        for cycle in range(self.start_cycle_number, self.number_of_cycles+1):
            self.cycle_path = self.runPath+'cycles/cycle%d/' % cycle
            if not os.path.isdir(self.cycle_path):
                os.mkdir(self.cycle_path)

            self.cycle = cycle
            self.current_cycle = cycle


            for state in ['SU','BOC','MOC','EOC']:

                if self.restart:
                    if state != {'SU':'BOC', 'BOC':'MOC', 'MOC':'EOC', 'EOC':'SU'}[self.current_state]:
                        continue
                    else:
                        if state != 'SU':
                            self.refuel = True
                        self.restart = False

                self.current_state = state

                self.write_to_logfile( "Cycle %d - %s" % (self.cycle, self.current_state) )

                self.inputfile_name  = self.model_names[self.current_state] + str(self.current_cycle) +
'.inp'       # designates new input file name
```

```
                self.write_next_input_file( self.model_name, self.inputfile_name  )
# creates new input file and loads fuel

                self.adjust_shim_positions(self.inputfile_name)                           #
determines shim angle based on state

                self.substitute_parameters(self.inputfile_name,
self.parameters_to_substitute_in_input_file)      # substitutes shim angle and burn time into
input

                self.delete_extra_white_rows(self.inputfile_name)                         #
reformats input file to be accepted into mcnp

                self.run_mcnp(self.inputfile_name)                                    # runs
mcnp

                self.read_mat_composition_from_tally(self.model_names[state]+str(cycle)+'.o')
# reads output file and captures burned fuel in temporary dictionary

                self.move_files([self.inputfile_name.split('.')[0] + ext for ext in ['.inp','.o']],
self.cycle_path) # moves the input and output to there special designated folders

                self.delete_extra_mcnp_files()                                    # deletes the
excess mcnp files in the working directory
```

```
    def delete_extra_white_rows(self,inputfile_name):
        self.tem_inputfile_name = self.inputfile_name
        with open(self.tem_inputfile_name,'r') as infile, open(self.tem_inputfile_name+'.o','w') as
outfile:
            for line in infile:
                if not line.strip(): continue  # skip the empty line
                outfile.write(line)  # non-empty line. Write it to output
        remove(inputfile_name)     # removes old file
        move(self.tem_inputfile_name+'.o', inputfile_name)# renames new file as old file
        # adding spaces where you need them
        with open(self.tem_inputfile_name,'r') as infile, open(self.tem_inputfile_name+'.o','w') as
outfile:
            for line in infile:
                if line.startswith('c SURFACES Block:') or line.startswith('c DATA Block:'): #checks
to se if the line starts with a Block point
                    outfile.write('\n') # if so it ads a space
                outfile.write(line)               #if not continues
        remove(inputfile_name)
        move(self.tem_inputfile_name+'.o', inputfile_name)

    def delete_extra_mcnp_files(self):

        files = os.listdir('.')
        for file in files:
            if (len(file) == 4) and (file[:3] == 'out'):
                os.remove(file)
            elif (len(file) == 5) and (file[:3] == 'src'):
                os.remove(file)
            elif (len(file) == 6) and (file[:3] == 'run'):
                os.remove(file)
            elif (len(file) == 6) and (file[:3] == 'com'):
                os.remove(file)



if __name__ == '__main__':
    print("Searching for equilibrium core with MCNP6")
    equil_core = EquilCore(debug=False)

    equil_core.go()
```

Python code for Power distribution

```
import os
import sys
import datetime as dt
import fnmatch
import traceback, sys
import fileinput
import numpy as np
import subprocess
from os import remove
from shutil import move
from operator import itemgetter
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.pyplot
#Kyle Britton

with open('SU_pow') as f:
    line_number = 0
    power = []
    cell =[]
    lat1 =[]
    lat2 =[]
    lat3 = []
    norm_pow=[]
    fe=[]
    for line in f:
                line_split = line.split()
                power.append(float(line_split[0]))
                cell.append(float(line_split[3]))
                lat1.append(float(line_split[7]))
                lat2.append(float(line_split[8]))
   # Normalize power
    norm_array=np.full(len(power),3980)
    norm_pow= np.divide(power, norm_array)
#assign cordinates to each power depending on arrays
    num_bins=18000
    z=[]
    m=0
    r=m
    n=m
    t=m
    e=m
    #creates z cordinates list
    while (n < num_bins):
        if cell[n]<20000:
            z.append(int((abs(cell[n]) % 100))+1)
            n=n+1
        elif cell[n]>=20000:
            z.append(-int((abs(cell[n]) % 100))-1)
            n=n+1

    #creates a fe list
    while (t < num_bins):
        num =str(cell[t])
        fe.append(int(num[1]+num[2]))
        t=t+1

    #places the created lists into one large one
    tot_power=[]
    power=np.zeros(len(norm_pow))
    while (e<num_bins):
        power=[norm_pow[e],lat2[e],lat1[e],z[e],fe[e]]
        tot_power.append(power)
        e=e+1
   #sorts the list and prints the maximum power with cordinates
    tot_power.sort(key=itemgetter(0), reverse=True)
    global_max=tot_power[0]
    global_min=tot_power[num_bins-1]
    print("The Global Max Power is %s located in x=%s y=%s z=%s Fuel element %s"
%(global_max[0],global_max[1],global_max[2],global_max[3],global_max[4]))
    print("The Global Min Power is %s located in x=%s y=%s z=%s Fuel element %s"
%(global_min[0],global_min[1],global_min[2],global_min[3],global_min[4]))
#sort and place into individual FE lists. list name corresponds to element
    i=0
    tot_power.sort(key=itemgetter(4), reverse=True)


    j1=tot_power[0:600]
    h1=tot_power[600:1200]
    f1=tot_power[1200:1800]
    d1=tot_power[1800:2400]
    k2=tot_power[2400:3000]
    i2=tot_power[3000:3600]
    e2=tot_power[3600:4200]
    c2=tot_power[4200:4800]
    l3=tot_power[4800:5400]
    h3=tot_power[5400:6000]
    f3=tot_power[6000:6600]
    b3=tot_power[6600:7200]
    m4=tot_power[7200:7800]
    k4=tot_power[7800:8400]
    i4=tot_power[8400:9000]
    e4=tot_power[9000:9600]
    c4=tot_power[9600:10200]
    a4=tot_power[10200:10800]
    l5=tot_power[10800:11400]
    h5=tot_power[11400:12000]
    f5=tot_power[12000:12600]
    b5=tot_power[12600:13200]
    k6=tot_power[13200:13800]
    i6=tot_power[13800:14400]
    e6=tot_power[14400:15000]
    c6=tot_power[15000:15600]
    j7=tot_power[15600:16200]
    h7=tot_power[16200:16800]
    f7=tot_power[16800:17400]
    d7=tot_power[17400:18000]

    xval=-2    #input here
    yval=2    #input here
    zval=-11    #input here
    FE=f1      #input here
    fe_graph='f1'  #input here

# Sort by Z value
    FE.sort(key=itemgetter(3), reverse=True)
    j1plt=[]
    tot_powerj1=[]
    while (i<len(FE)):
        j1plt= FE[i]
        if j1plt[3]==zval:
            j1plt2=j1plt[:4]
            tot_powerj1.append(j1plt2)
        i=i+1
    tot_powerj1.sort(key=itemgetter(2,1), reverse=True)
    tot_powerj1= np.array(tot_powerj1)
# Reformat data for heat map of element
    i=4
    j=0
    k=0
    j1p1 = np.zeros((5,5))
    while (j<5):
        while (i>-1):
            j1plt= tot_powerj1[k]
            j1plt2=j1plt[:1]
            j1p1[(j,i)]=j1plt2
            i=i-1
            k=k+1
        i=4
        j=j+1

    u = np.linspace(-2, 2, 6)
    v = np.linspace(-2, 2, 6)

    j1p1=j1p1[[4,3,2,1,0],:]
    plt.pcolor(u,v,j1p1,vmin=global_min[0], vmax=global_max[0])
    plt.colorbar()
    plt.title('Power Distribution for %s Element at Z=%s' %(fe_graph,zval))
    plt.ylabel('Y Coordinate')
    plt.xlabel('X Coordinate')
    plt.show()

# Determining the power in the Z
    Zpow=[]
    i=0
    while (i<len(FE)):
        j1plt= FE[i]
        if j1plt[1]==xval and j1plt[2]==yval:
            j1plt2=(j1plt[0],j1plt[3])
            Zpow.append(j1plt2)
        i=i+1
# plot in z
    print(Zpow)
```

Vita

Kyle Anthony Britton was born December 6, 1994, in Baltimore, Maryland, and is an American citizen. He graduated from Glen Allen High school, Glen Allen, Virginia in 2013. He received his Bachelor of Science in Physics from James Madison University, Harrisonburg, Virginia in 2017 before traveling the world. He received Masters of Science in Mechanical and Nuclear Engineering in 2018.