The Pennsylvania State University

The Graduate School

USE OF THE FISSION MATRIX FOR STEADY STATE AND QUASI-STATIC KINETIC MODELING OF THE TREAT REACTOR WITH TEMPERATURE FEEDBACK

A Thesis in

Nuclear Engineering

by

Alvaro Pizarro-Vallejos

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2019

The thesis of Alvaro Pizarro-Vallejos was reviewed and approved* by the following:

William Walters
Assistant Professor of Nuclear Engineering
Thesis Advisor

Azaree Lintereur
Assistant Professor of Nuclear Engineering

Arthur Motta
Professor of Nuclear Engineering
Head of the Department of Nuclear Engineering

*Signatures are on file in the Graduate School

ABSTRACT

In order to rapidly and accurately investigate the steady-state and kinetic behavior of the Transient Test Reactor (TREAT) at Idaho National Laboratories, we have coupled temperature-dependent fission matrix models to point kinetic calculations. The Monte Carlo reactor physics code Serpent is used to obtain a fission matrix that gives us the number of fission neutrons born in one core cell per fission neutrons in another cell. This fission matrix allows us to calculate two important parameters, the reactivity and neutron source distribution in each cell.

Fission matrices were pre-calculated at different temperature distributions in order to develop a fission matrix database set. Given an arbitrary temperature distribution, fission matrices from this database set can be interpolated in order to estimate the fission matrix for the given distribution. This interpolation method was evaluated on several artificial temperature distributions to test the accuracy in the fission matrix eigenvalue (multiplication factor) and eigenvector (source distribution). For a realistic temperature profile, the fission matrix interpolation method gave results within 50 pcm and 40 pcm of the Serpent reference result.

TREAT is typically used to perform temperature-limited transients, which deposit large amounts of energy into a central fuel testing location. Therefore, the kinetic simulation of TREAT is also important, in order to estimate the power profile and energy deposited in the test cell during transient operation. The kinetic behavior of a temperature-limited transient was modeled using point-kinetics, and by using the fission matrix with a quasi-static kinetic model. The quasi-static method accounts for the re-distribution of the power in the reactor during a transient, as opposed to the static power profile assumption of point-kinetics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my thesis advisor, Dr. William Walters, for providing me with the opportunity to work on this project, as well as his guidance and mentoring. I would also like to thank his research group members for providing me with constant help and support. This project could not have been completed without their contributions.

In addition, I would like to give thanks to Dr. Azaree Lintereur and Dr. Arthur Motta for agreeing to serve as my thesis readers. Furthermore, my special thanks go to the faculty and staff for being supportive during my time as a Master of Science student in Nuclear Engineering at Penn State.

Most importantly, I would like to thank my family and friends for their inspiration and encouragement throughout my graduate studies. Their unfailing love and support have given me the strength to overcome hardships throughout this journey.

Chapter 1

INTRODUCTION

1.1 Research Objectives

Although there have been several Monte Carlo simulations run for the Transient Test Reactor (TREAT) at Idaho National Laboratory and several benchmarks have been set, a fission matrix has not been calculated to explicitly study the effect of temperature distribution nor characterize the TREAT core. The fission matrix method is important because it allows us to obtain a very accurate neutronics solution, including temperature feedback, at a relatively low computational cost.

The defining objective of this research project is to develop a methodology using the fission-matrix to study the steady-state and kinetic behavior of TREAT with temperature feedback mechanisms. In order to conduct simulations, the Monte Carlo code Serpent is used with a benchmark model that was developed during the TREAT restart effort [1]. Knowing the geometry and material composition, Serpent is used to generate a set of fission matrices at different core temperature distributions. From these, we can solve for the reactivity, as well as the spatial fission distribution for an arbitrary temperature distribution by interpolating and combining the pre-calculated fission matrices. The accuracy of this fission matrix interpolation method is tested for several artificial temperature distributions.

In addition, the kinetic behavior of TREAT was analyzed using point kinetics and a quasi-static model using the fission matrix interpolation.

1.2 TREAT Background

TREAT is an air-cooled, thermal, homogeneous test facility designed to test reactor fuels and structural material performance. A central experimental location can be loaded with fuel and exposed to conditions simulating different types of nuclear excursions, such as a Loss of Coolant Accident (LOCA) or Reactivity Initiated Accident (RIA), and transient undercooling situations that could occur in a nuclear commercial power reactor. More specifically, transient tests are performed by exposing the fuel materials to short pulses of high-power radiation [1]. Conditions of TREAT normally require the rapid movement of control rods, easy access to the center of the core, and a temperature-dependent shutdown mechanism. Experimental fuels that are being tested are placed at the center of the core, and the cameras used for visualization are placed in the walls surrounding the reactor. A fast-neutron hodoscope is placed in one wall in order to detect fission neutrons emitted by the experimental fuel and obtain a spatial and time resolution of fuel motion during transients as well as fuel distribution [2]. The structure of the reactor is displayed in Figure 1. Operation of TREAT has a long history beginning in 1959, when it was first constructed. There have not been significant changes made in the structure of the reactor.

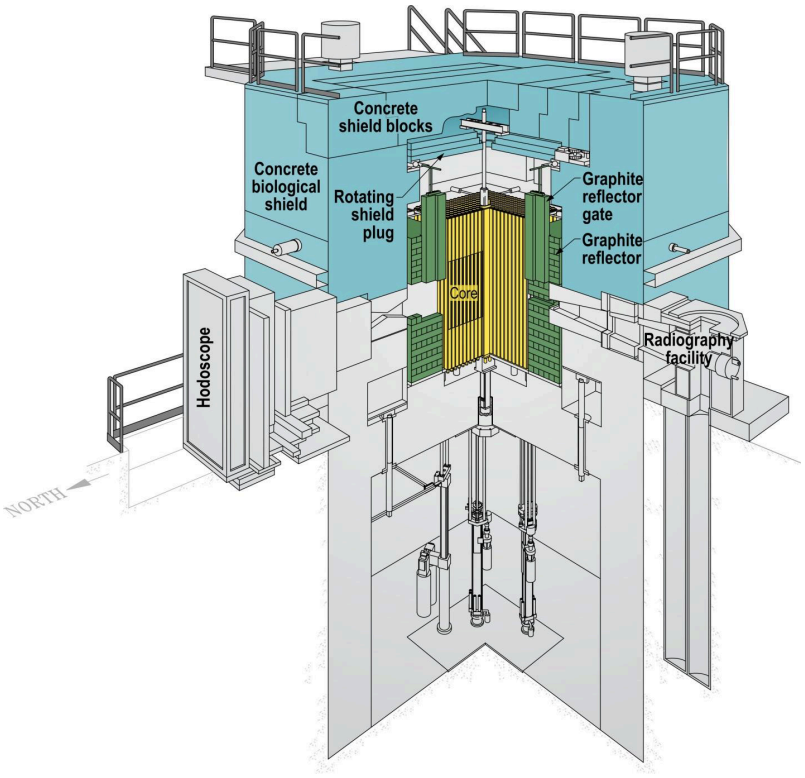

Figure **1-1**: Main Components of the TREAT Facility, including the Hodoscope [1]

TREAT was constructed in 1959 and ran continuously until 1994. During this time, TREAT generated over 720 MWh of energy in 6604 startups and 2884 transient irradiations. It was upgraded in 1988, in which new instrumentation and control systems were installed with refurbishment of rod drive systems [2]. The reactor was restarted in 2018 and the transient tests are scheduled to begin before 2020. For this research, the benchmarks that were used to conduct Monte Carlo simulations are based on the TREAT restart effort [3]. Restarting TREAT is a way to ensure that there is a method for testing any newly developed accident tolerant fuels. Currently, there are various NRC regulations regarding commercial reactor operations and fuel cycle performances. Since the purpose of TREAT is to test commercial reactor fuel, it significantly contributes to the nuclear industry growth. However, we need accurate methods to model these transients in order to facilitate future experiments at the facility.

Chapter 2

Description of Theoretical Concepts Applied

2.1 Fission matrix

2.1.1 Fission matrix Methodology

Monte Carlo simulation is one of the most prominent techniques when it comes to accurately calculating nuclear reactor cores parameters. However, this technique can be computationally expensive, since it requires a sequence of neutronics calculations [4]. This is especially true when Monte Carlo calculations, which rely on stochastic simulations of individual particles, attempt to reach an acceptable level of uncertainty. One of the most common contributions to Monte Carlo computational expenses are the calculations for localized tallies, like power distribution [4]. In order to avoid repeated calculations, a methodology known as the fission matrix is introduced. The fission matrix method has existed for a relatively long time, but over the recent years has gained more widespread interest due to increased computational capabilities. However, interests in using pre-calculated fission matrices has grown recently [5]. These interests involve a variety of combinations and interpolations of fission matrices, as was the case for this project. An accurate calculation of the fission matrix also has its own computational difficulties. Such difficulties arise when there are sharper changes in cell properties, such temperature and materials. Nevertheless, various correction methods have been and are continuing to be developed. Correction and improvement of the fission matrix method has become a vigorous area of research in recent years.

The basic fission matrix formulation is given in Equation 1.

$$F_i = \frac{1}{k}\sum_{j=1}^{N} a_{ij}F_j \quad (1)$$

In equation 1, a_{ij} is the number of neutrons created in cell *i* due to a neutron generated in cell *j*, which is calculated by Serpent. F_j and F_i are the number of fission neutrons in cell *i* and *j*, respectively; *k* is the eigenvalue.

The fundamental eigenvalue of the fission matrix is equivalent to the standard multiplication factor, and the fundamental eigenvector is the neutron source distribution. The fission matrix has been used previously to accelerate source convergence for Monte Carlo calculations [6]. Recently, it has been used as a direct method to perform criticality calculations by combination of fission matrices [7].

Fission source convergence is important because it determines the quantities that can be calculated in Monte Carlo criticality calculations. One key assumption is that the fission matrix is dependent on the fission source. This is a consequence of space discretization. Another assumption is that in rare cases the number of zones is infinite, which makes the fission matrix becomes dependent of the energy rather than the source [8]. With these assumptions, the fission matrix could be correctly calculated with any fission source and no inactive cycle would be needed. A limited case would be where only one zone is superimposed on the whole system in which the same number of inactive cycles as in standard Monte Carlo calculations would be required. Therefore, the reactor core with a large number of zones makes the fission matrix less sensitive to the errors in the initial fission source, requiring less inactive cycles [8]. However, the TREAT reactor consists of a small number of zones requiring various inactive cycles in order to make the fission matrix less error prone.

As mentioned before, the eigenvector has proven capable of accelerating source convergence. It is convenient to normalize the fission neutron source in order to obtain the fission

neutron weight in each zone, or cell [9]. Knowing the weight is crucial to being able to calculate any power distribution.

A fission matrix that is dependent on time can also be constructed. This is known as the Transient fission matrix. It can be used to obtain certain kinetic values, such as the delayed neutron fraction β_{eff} and neutron generation time Λ [7]. A transient fission matrix involves additional operators. The first and initial matrix takes into account the distinct behavior between the prompt and delayed neutrons [10]. Different matrices are calculated, one that represents the prompt or delayed emission spectrum and another that represents the prompt or delayed production of neutrons. The second kind addresses the kinetic aspect [10]. However, for the transient work presented in this project, we used the β_{eff} and Λ as calculated directly from Serpent and assumed to be constant.

2.1.2 Fission Matrix Combination

In order to obtain a fission matrix with an arbitrary temperature distribution, the fission matrix combination technique is required. The MATLAB code used to implement this technique is shown in appendix C. It is a separate file that conducts fission matrix combination as a function of the temperature distribution. Equation 15 is used to construct a new fission matrix based on the temperature distribution, and the MATLAB function is solely based on that formula. The function was called in the main file to conduct the fission matrix calculation at every time step.

As mentioned before, the fission matrix is needed to calculate the fission distribution and reactivity which sets the foundation for the power and temperature distribution. The fission distribution and reactivity calculations were the values that the function solves for before they are used in the main input file to obtain a new power and temperature distribution shape at every time step. When the process is repeated at the next step, the quasi-static relies on the arbitrary fuel

temperature distribution. The FM point kinetics model, on the other hand, uses a uniform temperature for every fuel present.

This set of operations makes the acceleration of source distribution possible. However, making calculations for a total operation time of 2 seconds implements obtaining 20000 different power and temperature distribution shapes. While the fission matrix allows to obtain an accurate representation of the property distributions, the number of models needed to illustrate spatial dependent transient calculation decelerates the source distribution convergence. This affects one of the main advantages of the fission matrix methodology.

2.1.3 Applications and Past Work

One of the most prominent examples of using the fission matrix for direct criticality calculation is for the Penn State Breazeale Nuclear Reactor (PSBR) at the Pennsylvania State University [11]. The goal was to converge the source distribution of the core. Figure 2-1 displays the core layout.

As indicated in Figure 2-1, the core is divided into cells when obtaining the fission matrix. Each of these cells has a corresponding number of fission neutrons which are generated either within their corresponding cells or from a neutron that comes from a different cell. When the eigenvalue was obtained, it was compared with the effective neutron multiplication factor, K_{eff} calculated by Serpent. The values are shown in Table 2-1. For the analysis of the PSBR, the Serpent calculation results were treated as the true values. It serves as a reference for testing the accuracy of the fission matrix method for determining the reactor criticality.

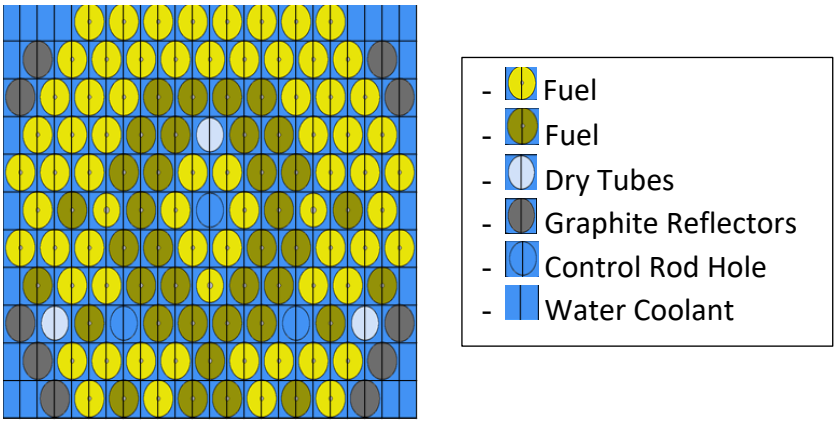

Figure **2-1**: Penn State Breazeale Nuclear Reactor Core Layout

Table **2-1**: PSBR fission matrix Data

Pitch x (cm)	2.1768
Pitch y (cm)	3.7703
Lattices x	24
Lattices y	11
dx (cm)	26.122
dy (cm)	20.737
height (cm)	38.1
Fission Matrix Keff	1.13631
RMS (Difference between Fission Matrix and Serpent Eigenvectors) (%)	1.44
RMS (Serpent relative uncertainty) (%)	1.19
Serpent Keff	1.13584
Keff relative uncertainty (pcm)	61.2
Error Between Fission Matrix and Serpent Keff (pcm)	38.3

As noticeable in Table 2-1, there is little a relatively low error in the fission matrix eigenvalue calculation. The accuracy of the eigenvector obtained from the fission matrix is noted by the Root Mean Square (RMS). The RMS shown is the difference between the fission matrix eigenvector and the one calculated by Serpent. Since it is a relatively low quantity, we can see the reliability of the eigenvector in obtaining a precise eigenvalue.

One prominent analysis of the PSBR in which the fission matrix was used involved temperature feedback. In particular, the strong negative temperature reactivity coefficient due to thermal upscattering was studied. This provides the foundation for temperature feedback, which effect is larger than that of Doppler broadening [5]. For this project, temperature dependence was studied, requiring coupled neutronic and thermal hydraulic calculations. Fission matrices were obtained using Serpent at several uniform fuel temperatures, which involved taking values from several thermal scattering cross section libraries and Doppler broadening for fuel temperature [5]. Once these Serpent calculations were complete, extraction and interpolations were performed in order to form a new fission matrix with an arbitrary spatial temperature distribution. A fission source from the fission matrix was then calculated and compared with the Serpent result. The results obtained proved the fission matrix methodology in this case to be effective. They also accurately displayed how the effects of negative temperature feedback caused the source for the variable temperature to be lower than that for uniform temperature [5].

In addition to testing temperature feedback, fission matrix correction methods have been applied to PSBR core analysis. These corrections were implemented to address the errors that fission matrix combination can introduce, which, for this project, were also studied for TREAT. These errors affect the fission source distribution, which influences the convergence efficiency. Among the errors analyzed include those caused by changes in material properties in a Serpent simulation [12]. In order to address this problem, methods to correct these errors at reflector and fuel type boundaries were developed. Development was made without adding significant computational cost, which lead to major improvement in fission source estimation [13]. The correction developed was used to correct the “end method”, which consists of combining fission matrices calculated on models with uniform temperature and material composition in order to apply them to models with non-uniform properties. It is this method, however, that introduces the errors in the fission source distribution. To address this error, the “ratio-correction” method was

developed, which required an additional Monte Carlo fission source estimate in a non-uniform model [14]. The ratio of this fission source is taken with the "end method" estimate of the fission source. As opposed to iterating to find the fission source, a uniform source distribution was assumed, which led to the conclusion that the correction factors were linearly proportional to fuel enrichment [14]. A similar process was conducted for TREAT but instead the relations between errors and kinetics were analyzed for this research project.

Besides temperature feedback, the fission matrix method has also been applied while taking into account control rod movement and its effect on reactivity and power. The PSBR contains four control rods. Three of the control rods have additional fuel below the absorber, known as fuel followers, while the fourth one lies in a hollow aluminum tube and is used to actuate the rod in square-wave and pulse operation modes [4]. The same fission matrix combination and interpolation protocol was followed [14]. Only this time, in order to account for the control rod insertions, additional database fission matrices were calculated at different control rod positions [4]. A bilinear interpolation was necessary in order to take into account both fuel temperature and control rod position. Just like previous applications, these interpolations help to obtain fission matrices for any temperature distribution and control rod position, which can then be used to solve for fission distribution (eigenvector) and Keff (eigenvalue) [4]. The results, when compared with serpent calculations, went on to show that the fission matrix is capable of simulating temperature feedback of the reactor core, taking into account the control rod positions, with minimum error.

Another major reactor in which the fission matrix methodology was applied to validate this method for computed power distributions is the advanced test reactor (ATR) at Idaho National Laboratory. For this project, the fission matrix was used to calculate the spatial distribution, similar to the process used for TREAT. For the ATR, the analysis of the computed and measured power distribution for code validation purposes was accomplished by an adaptation

of standard least-squares adjustment techniques [15]. To adjust any vector of computed quantities against a vector of corresponding measured data points that can be related to the quantities of interests through a matrix transformation, the least-square methodology is commonly used. This helps produce the best estimate of the quantities of interest and their uncertainties [15]. Quantities of interest, as a result of this process, can be used to estimate any bias as well as the uncertainty of any computational model. Thus, it is helpful for computational model improvement. A similar computational model test was conducted for TREAT, and its effectiveness was analyzed for this research project.

As mentioned before, a fission matrix can also be used to conduct transient studies. These are known as the transient fission matrix, which has been utilized in the past to analyze a Molten Salt Fast Reactor (MSFR). A transient fission matrix is key to performing spatial kinetic calculations with reduced computational cost through a pre-calculation of the Monte Carlo spatial and temporal response of the system, as was the case for the MSFR. Among the transient studies include reactivity insertion and load following capacities. This implements using an interpolation model to take into account the thermal hydraulics feedback on the neutronics, which include reactivity and neutron flux variation [10]. For the MSFR, the fission matrix evolves due to the reactor fluctuating during transient calculations. The neutron propagation can be impacted by the temperature. Therefore, transient fission matrix makes it unnecessary to conduct new Monte Carlo simulations [10]. For this project however, a transient fission matrix was never obtained. Rather, a quasi-static approach was used to combine the steady-state fission matrix with point kinetics.

2.2 Reactor Physics

2.2.1 Statics and Kinetics

In addition to the fission matrix, reactor statics and kinetics also played a crucial role in studying temperature feedback. Statics studies reactor operations at steady state, with dependence on energy and space. Steady state condition was used to conduct an initial test of the fission matrix applied to TREAT as it helped us obtain the spatial fission distribution and examine energy-dependent particle transport. Kinetics looks into how a reactor behaves during a transient operation. A point kinetics model is used to observe how the power output responds to a reactivity change over time. Unlike reactor statics, point kinetics has no dependence in space nor energy. The exact point kinetics equation was crucial to obtaining the reactivity feedback model during time of operation. The point kinetics equation is derived from the general time-dependent multigroup diffusion equation. Equation 2 displays time-dependent neutron diffusion equation.

$$\left[-M + F_p\right]\Phi(r,E,t') + S_d[\Phi(r,E,t')] = \frac{1}{v}\frac{\partial}{\partial t}\Phi(r,E,t) \qquad (2)$$

In equation 2, $\left[-M + F_p\right]$ is the diffusion operator, with M as removal and scattering and F_p as the prompt fission operator. These two variables are applied to the total flux Φ at position r, energy E, v is the velocity, and time t. S_d is the conventional delayed neutron source which has the form of a convolution integral over the flux history [16]. One way we can treat the central problem of the spatial dynamics is through factorizing the total flux $\Phi(r,E,t)$ into an amplitude $\Phi(t)$, and shape function $\psi(r,E,t)$. Equation 3 shows this factorization.

$$\Phi(r,E,t) = \phi(t)\psi(r,E,t) \qquad \phi(0) = 1 \qquad (3)$$

This factorization requires a separation condition for t>0. The splitting would occur in a way that $\phi(t)$ contains the main time-dependence of the total flux so that the time-dependence of $\psi(r,E,t)$ only has to account for the space-energy variations. Many conditions can be used, but

they all require that $\psi(r,E,t)$ remain positive and bounded at all points in space (r,E) for all time. Equation 4, which gives us a constant, would help meet those requirements [16].

$$\iint \frac{\psi^*(r,E,0)\psi(r,E,t)}{v} drdE = constant \quad (4)$$

By inserting equation 3 into equation 2, we can obtain the formula for the change in flux distribution shape. Dividing by $\phi(t)$ gives us equation 5.

$$\left[-M + F_p\right]\psi(r,E,t) + \frac{S_d[\phi(t')*\psi(r,E,t')]}{\phi(t)} = \frac{1}{v}\left[\frac{d\phi}{dt} * \frac{\psi(r,E,t)}{\phi(t)} + \frac{\delta}{\delta t}\psi(r,E,t)\right] \quad (5)$$

In addition to meeting the constraint requirements, equation 4 facilitates the transition to the point kinetics formulation. By splitting the fluxes $\phi(t)$ and $\psi(r,E,t)$ into two equations, we are able to couple them in one direction so that the necessary iterations between solutions can converge rapidly. Using equation 4 to reduce the $\phi(t)$ equation into the form of the point kinetics equation. Equation 6 displays the point kinetics formulation [16].

$$\frac{d\phi(t)}{dt} = \frac{[\rho(t)-\beta(t)]}{\Lambda(t)}\phi(t) + \sum_k \lambda_k \zeta_k(t) \quad (6)$$

Where, λ_k is the delayed neutron group k decay constant, $\zeta_k(t)$ is the power contribution from k , ρ is the reactivity, β is the delayed neutron fraction, Λ is the neutron generation time, and ϕ is the neutron flux. The flux $\phi(t)$ is proportional to the power, so equation 6 serves as the foundation for using point kinetics to make a transient calculation. Equation 6 takes into account the k groups of delayed neutrons, but for this project we only looked at a 1-group model. The exact point kinetics model, based on the proportionality between the flux in equation 5 and the power, is obtained using equations 7 and 8 where $\dot{p(t)}$ is the rate of change in power over time, and $\dot{\zeta}_k(t)$ is the rate of power contribution from the delayed neutron groups over time.

$$\dot{p(t)} = \frac{\rho-\beta}{\Lambda} * p(t) + \frac{1}{\Lambda} * \sum_k \lambda_k * \zeta_k(t) \quad (7)$$

$$\dot{\zeta}_k(t) = -\lambda_k * \zeta_k(t) + \beta * p(t) \quad (8)$$

In order to take the change in the spatial flux and power distribution shape into account, the Quasi-static approximation is used. For this approximation, the time variation of the shape function is of lesser importance in the range of interest, which allows us to neglect the time derivative of $\psi(r, E, t)$. This makes the entire right side of equation 5 equal to zero. The delayed neutron source and $(\frac{d\phi(t)}{dt})\frac{1}{\phi}$ is the same as in equation 6 [16]. The Quasi-static approximation is then calculated using equation 10.

$$\left[-M + F_p - \frac{1}{v} * \frac{1}{\phi} * \frac{d\phi}{dt}\right]\psi(r, E, t) = \frac{-S_d[\phi(t')\psi(r,E,t')]}{\phi(t)} \qquad (10)$$

The fission matrix was used for equation 10 by calculating the eigenvector which gives us the spatial flux distribution. This flux distribution, as well as the reactivity calculated from the eigenvalue, was used to obtain power distribution shape. Since equation 10 minimizes the importance of the time variation of the shape function, it does not fully solve equation 5. To fully solve equation 5, we must replace the time derivative of the flux shape with a backwards-difference approximation [17]:

$$\frac{\delta}{\delta t}\psi(r, E, t) = \frac{[\psi(r,E,t) - \psi(r,E,t-\Delta t)]}{\Delta t} \qquad (11)$$

According to equation 11, $t - \Delta t$ is the time of the last shape calculation. For this project the fission matrix was used to calculate the flux and power distribution at every timestep Δt. This shape function is slowly varying, allowing the first-order difference form to be applied over a much larger time step Δt than would be possible for $\Phi(r, E, t)$. This form can be easily applied to a computer code, and has great stability with large time step lengths, which results in a linear relationship over Δt in equation 4 [16]. For this equation, $S_d[\phi(t')\psi(r, E, t')]$ is calculated from the flux history just like the quasi-static approach and it is treated like an inhomogeneous source in equation 5. Thus, ψ is computed in equation 12 as:

$$\left[-M + F_p - \frac{1}{v}\left(\frac{1}{\phi}\frac{d\phi}{dt} + \frac{1}{\Delta t}\right)\right]\psi(r, E, t) = -\left\{\frac{S_d[\phi(t')\psi(r,E,t')]}{\phi(t)} + \frac{\psi(r,E,t-\Delta t)}{\Delta t}\right\} \qquad (12)$$

Equation 12 is also known as the Improved Quasi-Static (IQS) approach. It can be noticed that the space and energy shape of ψ relatively insensitive to errors in ϕ and its derivative. Computing $\psi(r, E, t)$ allows us to correct S_d to prevent error accumulation [16].

The Improved Quasi-Static Method can be used in a Transient-Reactor eXperiment Simulator (T-Rex) code. This code is capable of solving time-dependent transport equations with an accurate representation of delayed neutrons with minimal approximations using the IQS method and using a Monte Carlo simulation to represent the geometry [17]. One assumption made for the IQS is that the angular flux made be factored into the product. The flux shape is obtained from the solution to a modified fixed source neutron transport equation. An important reason why the IQS helps to compare the T-rex code with the fission matrix is because it represents the time-derivative of the flux shape, which was our major interest for this project [17]. T-Rex uses modified in-house versions of KENO, the code used for the SCALE software, to compute shape functions stochastically while computing the amplitude function deterministically by solving both a set of amplitude functions and precursor concentration equations. Simulations of TREAT using T-Rex have been conducted through KENO Va. and KENO-VI models [18]. KENO Va. code is able to analyze models constructed with combinations of basic shapes such as spheres, cylinders, and cuboids. KENO VI., meanwhile, supports a wide array of geometries including planes, dodecahedrons, wedges, and parallelepiped [17]. This was done for the M8 calibration series [17].

2.3 Reactor Thermal Properties

2.3.1 Heat Transfer

Since the one of the objectives of this research is to use the fission matrix to study temperature feedback mechanism as a result of the statics and kinetics, it is important to incorporate thermal and heat transfer properties of the materials. For the TREAT reactor, graphite thermal properties were used to make temperature change calculations based on the power output obtained from the point kinetics model. Equations 13 and 14 solve for the temperature change as a function of power $p(t)$, heat capacity C_p, density ρ, fuel volume V_f, and timestep dt.

$$T_i = T_{i-1} + \Delta T \tag{13}$$

$$\Delta T = \frac{p(t)*dt}{\rho*V_f*C_p} \tag{14}$$

The power $p(t)$ was obtained directly from the point kinetics model [19]. . Fuel thermal density is assumed to be constant during the entire operation.

2.3.2 Temperature Interpolation and fission matrix Combination

Fission matrix combination is conducted as a function of temperature using linear interpolation. It consists of calculating fission matrices for uniform temperatures models, which can then be applied to arbitrary temperature distributions. These fission matrices make up the fission matrix database. In order to obtain a fission matrix with an arbitrary temperature distribution, linear interpolation was conducted between database fission matrices with the nearest temperature. The linear interpolation method for combining fission matrices is illustrated by equation 15 [20].

$$FM_R = FM_0 + (T_R - T_0) * \frac{FM_1 - FM_0}{T_1 - T_0} \tag{15}$$

In equation 15, FM_R is the reactor fission matrix, T_R is the reactor temperature, T_0 and T_1 are the lower and higher database temperatures, respectively, FM_0 and FM_1 are the database fission matrices calculated at T_0 and T_1. It is important to notice that FM_R, FM_0, and FM_1 are single rows of a fission matrix with its own temperature. In addition, T_R is a single value that corresponds to the temperature of a certain region, or cell, of the core. Meanwhile, the values for T_0 and T_1 used in equation 15 to obtain FM_R are selected according to the data temperatures that T_R lies between. This interpolation is then applied to every cell in the core, depending on the local temperature of each cell. Fission matrix combination through linear interpolation of the rows is also known as the 'end method' in the sense that it calculates fission matrices based on the value at the 'end' of the neutron path [14]. The end method can also be applied to models with non-uniform properties without conducting additional Monte Carlo simulations, as was the case for this project.

Chapter 3

TREAT Components

3.1 Core Components and Geometry

3.1.1 Fuel and Control Rod Assembly

The minimum critical core (MCC) core for TREAT is composed of 133 fuel assemblies, 8 fuel assemblies with control rod, and 16 zirconium assemblies. Therefore, in the fission matrix, there are a total of 141 assemblies where fission occurs. The geometric layout of the MCC core is displayed in Figure 3-1 [2].

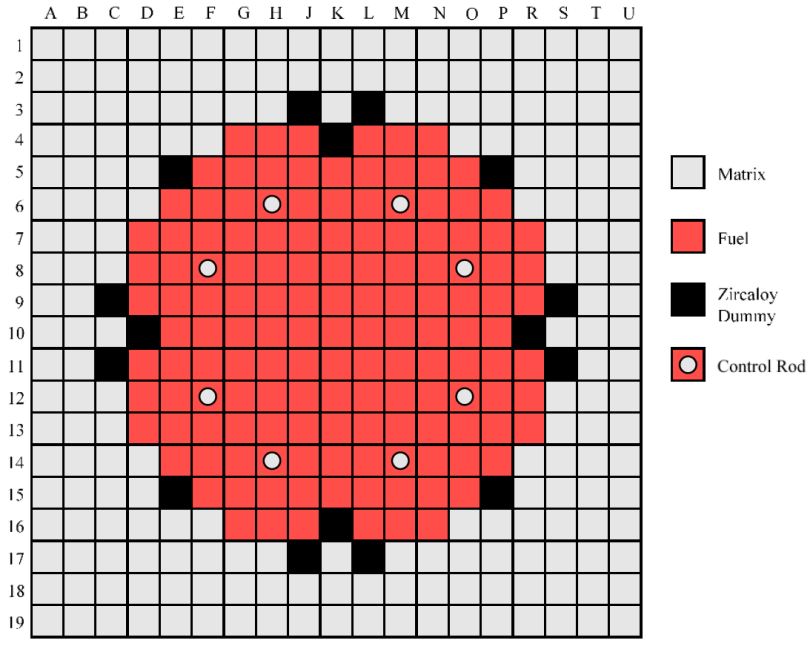

Figure **3-1**: TREAT Minimum Critical Core Configuration [2]

As noticed in Figure 3-1, the MCC configuration is divided into cells, each cell contains a certain material. The cells colored in gray contain no fuel. It can be observed how the control rod cells, which are colored red with gray circle in the middle, are composed of a certain amount of fuel. The fuel assemblies are colored in red, while the zircaloy are colored in black.

A standard fuel assembly in the MCC consists of mostly graphite with a small amount of highly enriched uranium (HEU) fuel (93 wt% U-235) mixed in. The HEU is surrounded by a zircaloy can. Two graphite sections are placed above and below the fuel, which contains outgas tube that collects fission products [2]. This is important for flux control. As suggested by Figure 3-1, the fuel assembly is squared shaped and it is surrounded by chamfered edges that produce small channels for coolant to flow through. Figure 3-2 displays the fuel assembly.

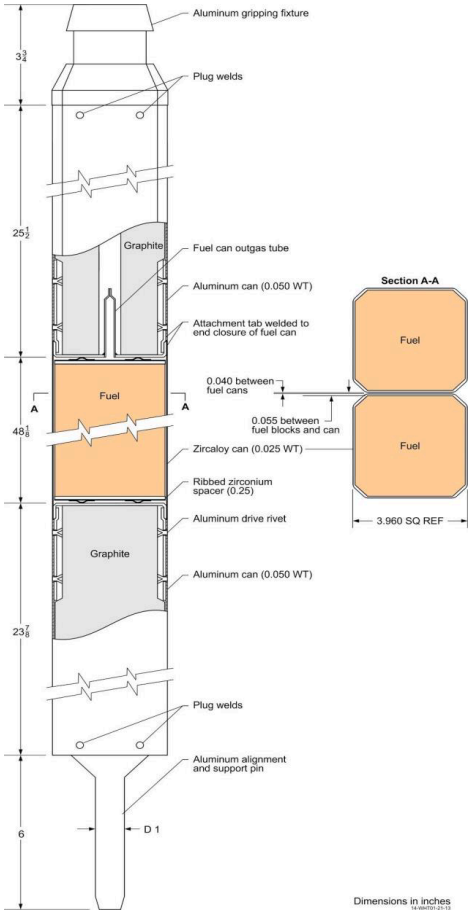

Figure **3-2**: Standard Fuel Assembly [21]

The standard control rod assembly is a fuel assembly with a hole in the center. This hole allows for vertical control rod movement. For the MCC core, however, the hole in the assembly was left empty. It was done since the objective of configuring an MCC, after TREAT was restarted, was to measure the temperature coefficient, as part of this benchmark. For other configurations, a control rod would be inserted into the holes of any assemblies in order to conduct other TREAT measurements.

Other assemblies that are part of the MCC configuration are composed of aluminum-clad and zircaloy-clad dummy fuel assemblies [1]. The purpose of the aluminum-clad dummy is to provide the MCC with additional reflection when fewer assemblies composed of fuel are present. As for the zircaloy-clad dummy, it serves to provide a thermal buffer [2]. These assemblies don't add neutrons to the fission matrix since, in the Serpent code run to test the benchmarks, they were treated as not containing fuel or any other fissile material.

3.2 Material Composition and Temperature

3.2.1 Fuel and Overall Assembly

The fuel assembly is mainly composed of graphite. Carbon is present in the assembly both as an element and graphite. The carbon that was not in the form of graphite is important to take into account for neutronic analysis. Therefore, when conducting neutronic analysis, the scattering behavior of carbon is treated differently whether it is graphite or not. For the graphite fabrication process, the formulation used included 58 wt% graphite flour, 19.3 wt% thermax, and 22.7 wt% koppers coal tar [22]. In the coal tar, the weight fraction for carbon content is reported to be 0.913. A baking process called graphitization was performed during fabrication, in which a

temperature of 950 °C was maintained [23]. This allowed for 59% graphitization. Table 3-1 shows the spectrochemical analysis of the core graphite. The values presented in this Table are the impurities of graphite. Table 3-2, meanwhile, shows the composition of uranium in the core.

Table **3-1**: Core Graphite Impurities (wppm) [2]

Sample No.	B	Fe	V	Gd	Eu	Sm	Cd
1	13	300	30	<0.2	<0.2	<1	N/A
2	4	400	30	<0.2	<0.2	<1	N/A
3	10	400	30	<0.2	<0.2	<1	N/A
4	5	500	30	<0.2	<0.2	<1	N/A
5	7	1000	N/A	N/A	N/A	N/A	N/A
6	4	1000	N/A	N/A	N/A	N/A	15

Table **3-2**: Core Uranium Composition [2]

Isotope	Content (wt.%)
234U	0.91
235U	93.239
236U	0.428
238U	5.413

The assemblies are also composed of zirconium, aluminum, steel, as well as other metals. However, their composition did not affect the thermal and heat transfer properties of the fuel. Due to the composition of these metals shown in the Serpent input material cards, we ignored the metals listed above, and used graphite properties in our heat transfer calculations.

3.2.2 Control Rod

Control rods were not inserted into the TREAT MCC when it was restarted. Nevertheless, it is still important to take them into account since they contribute in particular to any reactivity insertion. Reactivity insertion is fundamentally crucial in a point kinetics model.

The control rod has a poison section, B_4C, in powder form. Its density has changed twice since 1960 mainly due to densification and voiding issues. Impurities for B_4C powder are negligible given its density of 1.8 g/cm^3 [24]. Incoloy alloy is used to provide the cladding with strength at high temperatures and corrosion resistance [1]. The standard composition of incoloy alloy is listed in Table 3-3.

Table **3-3**: Chemical Compositions (wt%) of Incoloy Alloy [25]

UNS Designation	N08800	N08810	N08811
Incoloy Alloys	800	800H	800HT
Ni	30-35	30-35	30-35
Cr	19-23	19-23	19-23
Fe	39.5 min.	39.5 min.	39.5 min.
Cr	0.10 max.	0.05-0.10	0.06-0.10
Al	0.15-0.60	0.15-0.60	0.25-0.60
Ti	0.15-0.60	0.15-0.60	0.25-0.60
Al+TI	0.30-1.20	0.30-1.20	0.85-1.20
ASTM grain size	N/A	5 or coarser	5 or coarser

The density of the incoloy alloy is approximately 7.94 g/cm^3 [25]. This control rod serves to control the neutron flux. This resistance to high amounts of heat as well as corrosion tolerance makes it an effective neutron absorber.

3.2.3 Cooling System

The coolant used in TREAT reactor is air. Its density is approximately 0.0012 g/cm^3, and its composition is listed in Table 3-5. It is the typical air encountered in the atmosphere at INL.

Table **3-4**: Chemical Composition of Air [26]

Component	Volume (ppm)
Nitrogen (N2)	780840
Oxygen (O2)	209460
Argon (Ar)	9340
Carbon Dioxide (CO2)	380
Neon (Ne)	18.18
Helium (He)	5.24
Methane (CH4)	1.7
Krypton (Kr)	1.14
Hydrogen (H2)	0.55
Water (H2O)	10000

Chapter 4

Simulations and Calculations

In order to study the statics, kinetics and temperature feedback mechanism of TREAT, several fission matrices were calculated. The fission matrix methodology was tested for different spatial fuel temperature distributions. Due to the strong temperature feedback effect, the temperature distribution has a large effect on both the reactivity and power distribution.

It is important to study the effect of fuel temperature on the reactivity as well as the contribution of the delayed neutrons to the TREAT power. This is the key to analyzing the statics and kinetics of the reactor. Therefore, we calculated several fission matrices for both uniform and non-uniform models.

One limitation of the methodology is that although the neutronics model is in 3D, the temperature distributions and fission matrices used are only 2D. This is not realistic, but the methodology is consistent between the different models and is used as an initial test of the fission matrix method.

4.1 Core Model with Uniform Temperature

4.1.1 Fission matrix Model

The fission matrix indicates how many fission neutrons are in each cell generated by neutrons traveling from another cell. The FMTX command in Serpent was used to obtain the fission matrix for the MCC with a uniform temperature. At the same time, the detector card is used to tally the fission rate in each cell, in order to obtain reference values for comparison and

validation. For the fission matrix, a certain number of cells, 169, were selected from the ones shown in the core layout in Figure 1. This selection was made in order to exclude most of the cells that did not contain any fuel. The cells are arranged in a matrix consisting of 13 rows and 13 columns. Figure 4-1 shows the arrangement of the cells that the core was divided into when performing the Serpent simulation, as well as the fission distribution obtained from the fission matrix itself. An eigenvector was calculated in which each individual value represents the amount of fission neutrons in each cell. The fission rate tallied by Serpent is shown in Figure 4-1.

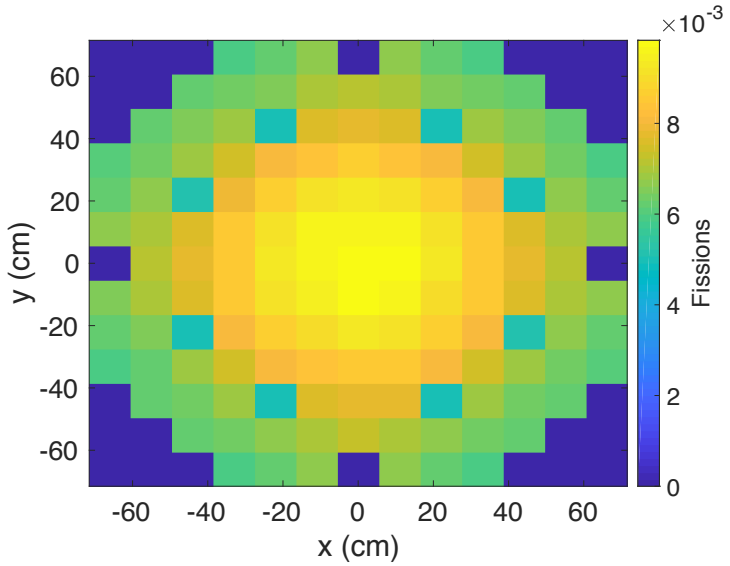

Figure **4-1**: MCC Fission Distribution Calculated by Serpent with Uniform Temperature (300K)

For the core layout in Figure 4-1, the temperature was set up to be 300 K and uniform. From this, the eigenvalue and eigenvector were calculated for a steady state condition to study the TREAT statics with space dependence. These calculations were used as an initial case study for the accuracy of the fission matrix (FM) method. The Serpent ENDF/B VII.1 cross section library was used, and thermal scattering S(α,β) treatment was used for the graphite. The Serpent criticality calculation used 100,000 particles with 100 active and 50 inactive cycles.

4.1.2 Fission Matrix Solution

Once the fission matrix was calculated, the next step was to calculate the source distribution and k-eff eigenvalue. For this project, a MATLAB script, shown in Appendix A, was written in which the eigs function was used to solve for the fundamental eigenvalue and eigenvector which represent the k-eff and fission distribution. Since there is a total of 169 spatial cells, the FM consists a total of 28561 elements.

4.1.3 Results and Analysis

Once the fission distribution was calculated, the root means square error (RMSE) was obtained. For the RMSE calculation, the FM fission distribution was compared to the detector card results. Equation 16 was used to calculate the RMSE. When performing this calculation, the detector card output was considered the reference value.

$$RMSE = \frac{\sqrt{\sum_{n=1}^{N} \frac{(\hat{x}_n - x_n)^2}{\hat{x}_n^2}}}{N} \tag{16}$$

For the equation 16 variables, x_n is the FM number of fissions in cell n, $\hat{x}_n$ is the Serpent Monte Carlo number of fissions in cell n detector, and N is the total number of cells. As mentioned before, only the cells that contained fuel or fission material were selected to test the accuracy of the fission matrix method. Both the FM and Monte Carlo fission distribution are plotted in Figure 4-2. The error bars are also plotted, which represent the uncertainties of the Monte Carlo fission distributions. In order to further test the accuracy of the data, several important parameters for the 300 K uniform temperature model, listed in Table 4-1, were calculated. Among these parameters include the fission matrix reactivity coefficient K_{FM}, the RMSE, the RMS of the relative Monte Carlo fission distribution uncertainties, the Monte Carlo

reactivity coefficient or K_{MC}, the K_{MC} uncertainty, and the K_{eff} error. The K_{eff} error was calculate using equation 17.

$$E(pcm) = \left(\frac{K_{MC}-K_{FM}}{K_{MC}}\right) * 1x10^5 \quad (17)$$

For equation 17, E is the error in pcm, K_{MC} is the Serpent eigenvalue, and K_{FM} is the fission matrix eigenvalue. K_{MC} is being treated as the reference value. The rest of the values in Table 4-1 were calculated using equations 15-16.

Table **4-1**: Comparison of Fission Matrix and Standard Monte Carlo Parameters for the MCC core at 300 K.

K_{FM}	1.02863
RMSE (%)	0.36 %
Serpent uncertainty RMS (%)	0.18 %
K_{MC}	1.02872
K_{MC} relative uncertainty (pcm)	19.1
K_{FM} error (pcm)	9.8

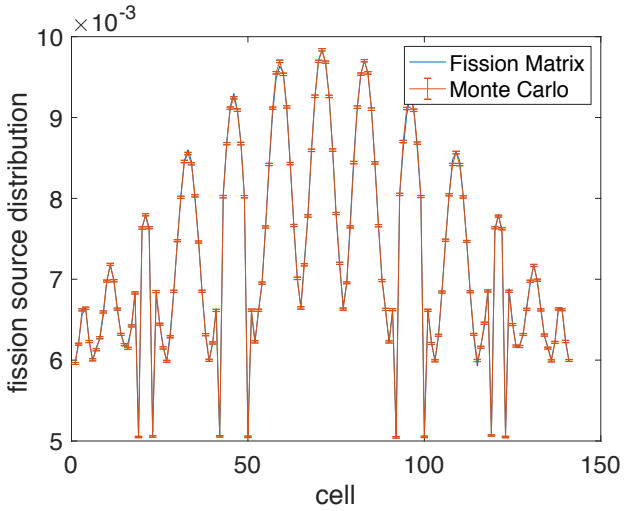

Figure **4-2**: Comparison of Fission Matrix and standard Monte Carlo Source Distribution for the MCC core at 300 K

The values for both K_{FM} and K_{MC} are what we expect given that the reactor is supercritical at 300 K. The Figure 4-2 plot, as well as the two RMSEs listed in Table 4-1, demonstrate little error in the fission matrix for a uniform temperature model. For a uniform temperature model, an RMSE and serpent uncertainty RMS less than 1% is acceptable, as is the same for a K_{FM} error lower than 45 pcm.

4.2 Static Temperature Distribution Models

4.2.1 Linear Temperature Distribution Model

To test the accuracy of a non-uniform temperature fission matrix, we began by implementing a simple linear temperature distribution using the Serpent multi-physics interface. Since, the physics interface card allows us to assign each cell its own material temperature and density, we can arrange the properties in the way we intend to simulate the core. Figure 4-3 displays a linear temperature distribution with a 600 K peak. A regular mesh-based interface was used. In this case, it was possible to bring in the temperatures and densities on the cartesian mesh that displayed the temperature distribution in Figure 4-3. This mesh-based interface was used for other temperature distributions we studied. This simple linear temperature distribution was used to initially test the fission matrix combination, as well as study the reactor statics.

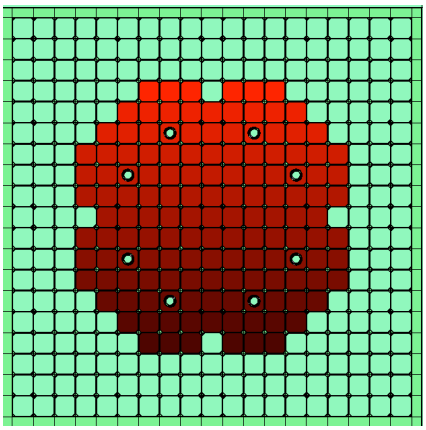

Figure **4-3**: MCC Core Linear Temperature Distribution With 600 K Peak. The temperature at the bottom row is 300 K and it rises by 25 K at every row.

In Figure 4-3, the linear temperature rises up at every row. The temperature at the bottom row, or minimum temperature, is 300 K and it rises up by 25 K at every row. Therefore, the maximum temperature in the top row is 600 K. Other temperature distributions were also used to test the fission matrix method. Another linear temperature distribution model was also calculated in which the minimum temperature is also 300 K but it rises by 75 K at every row until a peak temperature of 1200 K is reached. This temperature distribution is displayed in Figure 4-4.

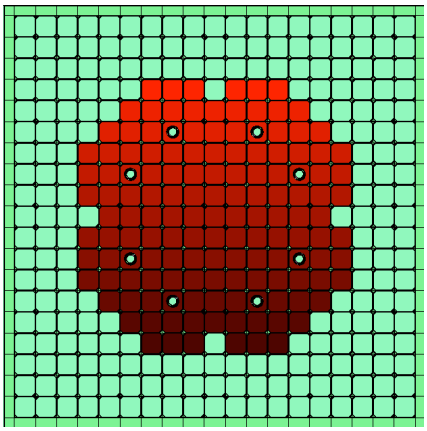

Figure **4-4**: MCC Core Linear Temperature Distribution with 1200 K Peak. The temperature at the bottom row is 300 K and it rises by 75 K at every row.

Besides linear, two realistic temperature distribution models were also tested. The first temperature distribution is shown in Figures 4-5.

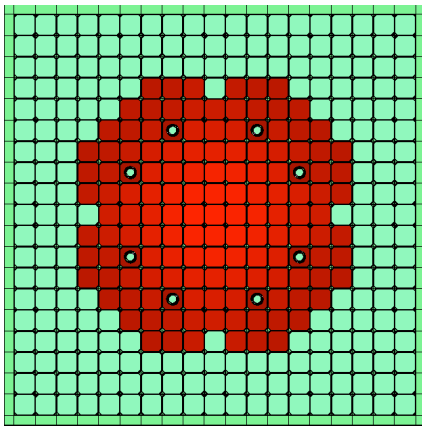

Figure **4-5**: MCC Realistic Temperature Distribution with 594.19 K Peak. The temperature distribution is proportional to the power distribution, with the minimum temperature at 300 K.

As shown in Figure 4-5 the temperature peak, 594.19 K, occurs at the center of the core. This is due to the temperature being proportional to the power and fission distribution. This temperature distribution was calculated by assuming an increase in temperature of each cell above 300 K proportional to the power distribution calculated at 300 K. Another fission matrix was calculated for a realistic temperature distribution with a higher peak. This model is displayed in Figure 4-6.

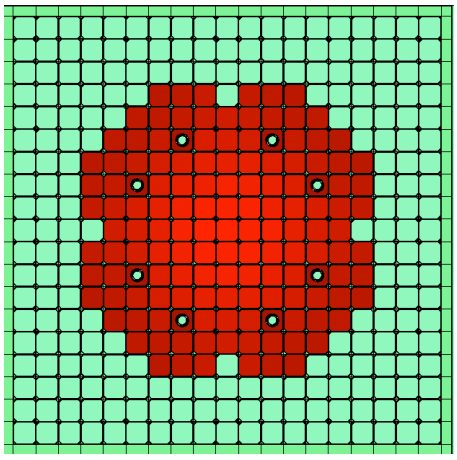

Figure **4-6**: MCC Realistic Temperature Distribution with 1035.3 K Peak. The temperature distribution is proportional to the power distribution, with the minimum temperature at 300 K.

For the temperature distribution in Figure 4-6, the peak is 1035.3 K. The overall goal of calculating fission matrices for the models shown in Figure 4-3 to 4-6 is to test the effect of

temperature magnitude and spatial distribution on the accuracy of the method of combining and interpolating fission matrices.

Since the interface card allows us to easily bring in density and temperature fields, it can automatically produce power distributions to be coupled with other codes. For this research project, the interface was coupled with both the fission matrix calculation and the Monte Carlo Serpent input script. Using the interface card required the thermal cross section data to be interpolated as a function of temperature.

4.2.2 Fission Matrix Combination Application

The fission matrix combination technique is primarily based on extracting elements from different fission matrices. In order to successfully perform the technique, we first had to calculate several individual fission matrices for uniform temperature models. These calculations required the coupling of multi-physics interface cards, fission matrix and Serpent input.

Fission matrices were calculated for uniform temperature models that range from 300 K to 1200 K, with a 100 K difference between each model. We then extracted values from these fission matrices and combined them to form a fission matrix with the Figure 4-3 linear temperature distribution model.

The fission matrix combination technique is a less computational expensive method for creating a fission matrix dealing with any temperature distribution. Once the new fission matrix is formed, we can converge the source distribution and test the accuracy of this methodology. This is performed by calculating the eigenvalue and eigenvector using the MATLAB code in Appendix B, from which we can obtain the RMSE and Keff error. Thus, we can validate and test the efficiency of the fission matrix method for applications to different types of temperature distribution models.

4.2.3 Results and Analysis

The fission distributions for the models shown in Figures 4-3 to 4-6 are represented by the fission matrix eigenvectors. A fission distribution was calculated for each of the static models that were analyzed. Figure 4-7 displays the fission distribution calculated using the linear 600 K temperature profile (as in Figure 4-3).

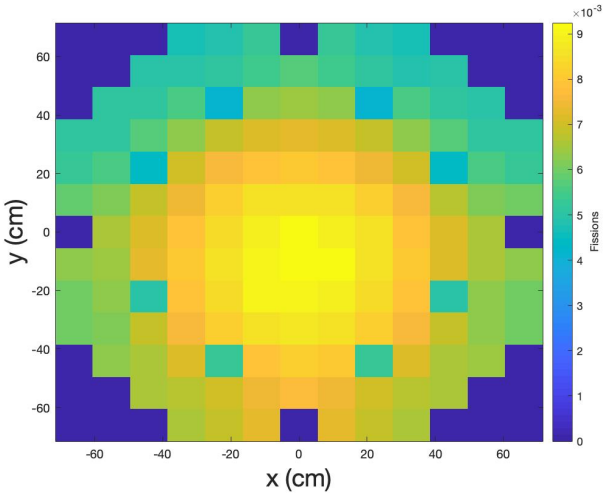

Figure **4-7**: MCC Fission Distribution with 600 K Peak Linear Temperature Distribution

As we notice in Figure 4-7, more fission occurs in the colder bottom region, as compared to the uniform temperature case (Figure 4-1). This is due to a lower temperature leading to a higher reactivity, thus higher amounts of fissions. Thus, more overestimations of the FM fission distribution occur particularly in the most central regions when compared to the Monte Carlo results.

With the eigenvector calculated, we obtained the RMSE using the MATLAB code in Appendix B. The RMSE allowed us to compare the FM to the Serpent Monte Carlo fission distribution. The fission matrix and Monte Carlo fission distributions are plotted in Figure 4-8. Just like in the case with the uniform temperature model, the error bars represent the uncertainties of the Monte Carlo fission distribution. As noticed by the plots, there is relatively small

difference between the FM and Monte Carlo fission distributions. There is, however, a small overestimation in the hot regions, and underestimation in the cold regions by the fission matrix combination method. This indicates that the fission matrices calculated using uniform temperatures without any corrections have some small errors when applied to a non-uniform case. Additional results are presented in Table 4-2.

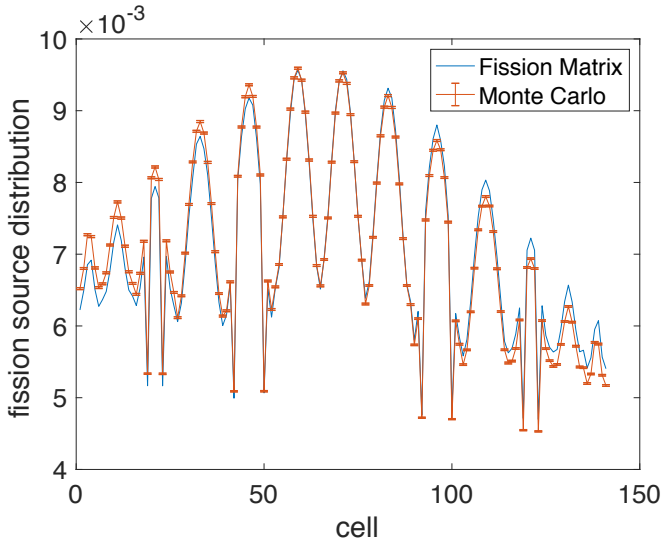

Figure **4-8**: MCC Fission Matrix and Monte Carlo Linear Fission Distribution with 600 K Peak Temperature

Table **4-2**: TREAT Linear Temperature Distribution Fission Matrix Data with Smaller 600 K Peak Temperature

K_{FM}	0.99680
RMSE	2.46 %
Serpent uncertainty RMS (%)	0.16 %
K_{MC}	0.99691
K_{MC} relative uncertainty (pcm)	20.9
K_{FM} error (pcm)	29.4

As noticed in Table 4-2, the error for the K_{FM} is 29.4 pcm, greater than the one for a uniform temperature model, 9.82 pcm. However, given that the relative uncertainty of the Serpent calculation, 20.9 pcm, is only 2 pcm greater than the one for the uniform temperature model, 19

pcm, it's not clear how large this effect of a non-uniform temperature model is. Figure 4-9, which displays the difference in fission distribution, indicates which parts of the core have largest differences in fission.

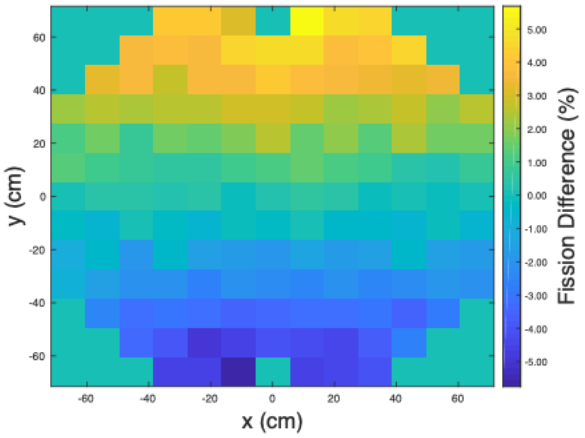

Figure **4-9**: MCC Fission Distribution difference (FM-MC) for Linear Temperature Distribution with 600 K Peak Temperature

These errors contribute to the RMSE and K_{eff} error. The largest differences occur in the bottom and top regions of the core. We can see in Figure 4-3 that the minimum and maximum temperatures occur in these areas. Both the FM and Monte Carlo fission distributions follow the same geometric pattern illustrated in Figure 4-7. However, the fission matrix underestimates the amount of fissions in the colder region, due to a higher reactivity leading to more fission production in the Monte Carlo simulation. As for the top, hotter cells, the number of fissions is overestimated by the fission matrix since lower reactivity leads to less fission production. Where the least fission difference occurs is in the central regions. They tend to receive more neutrons from every other cell in the core, which leads to large accumulations according to Figure 4-7. These accumulations bring the FM number of fissions closer to the values computed by the Serpent Monte Carlo simulation. Since the core model consists of a linear temperature

distribution with a 600 K peak, the errors are not relatively significant. This is due to the small temperature gradient of 25 K causing low difference in reactivity, thus fissions, among cells. Thus, a low temperature gradient helps to maintain the RMSE and K_{eff} errors relatively low.

The linear temperature distribution shown in Figure 4-4 was also analyzed using the fission matrix technique. This temperature distribution had a peak of 1200 K, and a larger gradient. The fission distribution obtained for that model is displayed in Figure 4-10. It is important to analyze how a higher temperature would affect the accuracy of the fission matrix method.

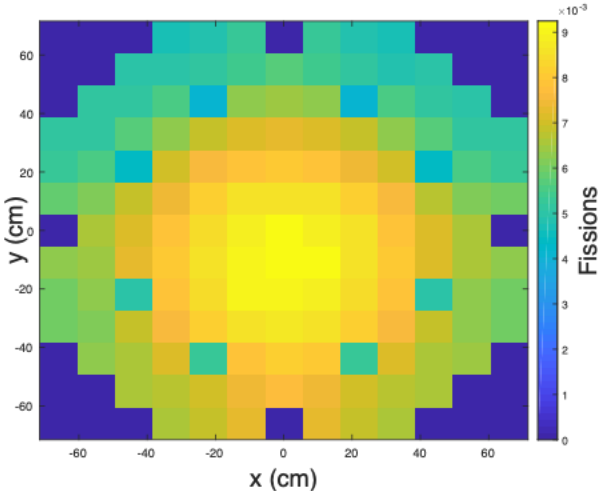

Figure **4-10**: MCC Linear Fission Distribution with 1200 K Peak Linear Temperature Distribution

As shown in Figure 4-10, the cells with the lowest temperatures consists of the most fissions, similar to Figure 4-7. The fission distribution in Figure 4-10, is shown to be directly proportional to the temperature distribution in Figure 4-4. For the Monte Carlo fission simulation, an interface card with the temperature distribution in Figure 4-4 coupled with a Serpent input file. This allowed us to calculate the number of fissions in each cell as well as the uncertainties. When comparing them to the fission matrix values, we conducted the calculations for the RMSE and Keff error; the same process as the one for the linear temperature distribution with the 600 K peak. These values are listed in Table 4-3.

Table **4-3**: TREAT Linear Temperature Distribution Fission Matrix Data with 1200 K Peak Temperature

K_{FM}	0.94931
RMSE	5.20 %
Serpent uncertainty RMS (%)	0.16 %
K_{MC}	0.94941
K_{MC} relative uncertainty (pcm)	19.8
K_{FM} error (pcm)	5.5

As noticed in Table 4-3, the K_{FM} error is still very small (5.5 pcm), compared to the uncertainty in the reference calculation (20 pcm). The RMSE, however, increases with rising temperature since a large temperature gradient leads to more significant difference in the number of fissions among each cell. These differences may be due to the linear interpolation between temperature points, or due to combining fission matrices of uniform temperature using the end-method. The number of fissions obtained from the fission matrix and Monte Carlo simulation corresponding to the model in Figure 4-4 are plotted in Figure 4-11. The spatial fission difference distributions are plotted in Figure 4-12.

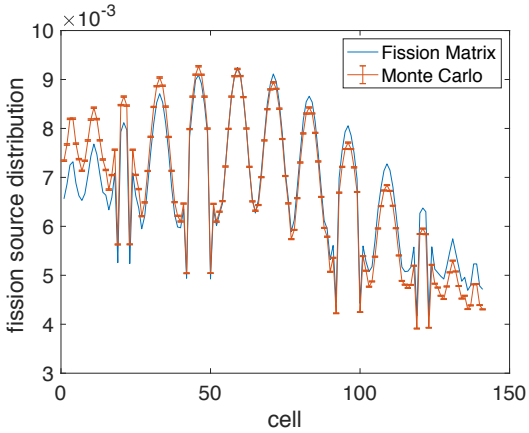

Figure **4-11**: MCC Fission Matrix and Monte Carlo Linear Fission Distribution with 1200 K Peak Temperature

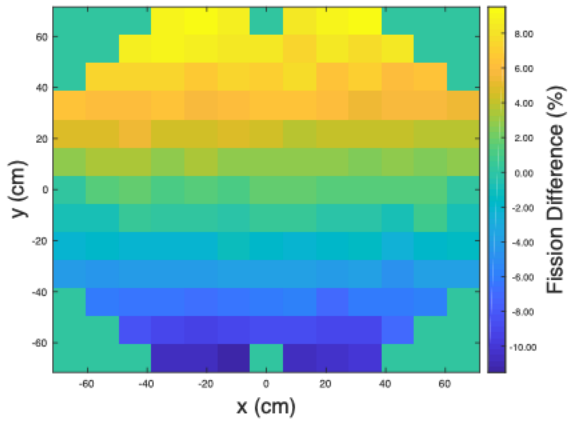

Figure **4-12**: MCC Fission Distribution difference (FM-MC) for Linear Temperature Distribution with 1200 K Peak Temperature

Figure 4-12 allows us to see that the fission difference distribution follows a similar pattern to that of the 600 K peak linear temperature distribution. One difference is that the fission matrix, when compared to the Monte Carlo simulations, overestimates and underestimates the fission distribution to a higher degree at hotter or colder temperatures. The very high temperature gradient causes an error when uniform-temperature fission matrices are combined. However, the eigenvalue is still very accurate, even for the large gradient.

In addition to linear temperature distributions, more realistic temperature distribution models were analyzed. For these cases, the temperature and power distribution were proportional. Our intention was to study the accuracy of the fission matrix in analyzing realistic temperature distributions for the TREAT reactor. Two arbitrary temperature distribution models were examined, one with a lower and the other with a higher peak temperature.

The fission distribution calculated by Serpent with a 594.19 K peak realistic temperature distribution is displayed in Figure 4-13.

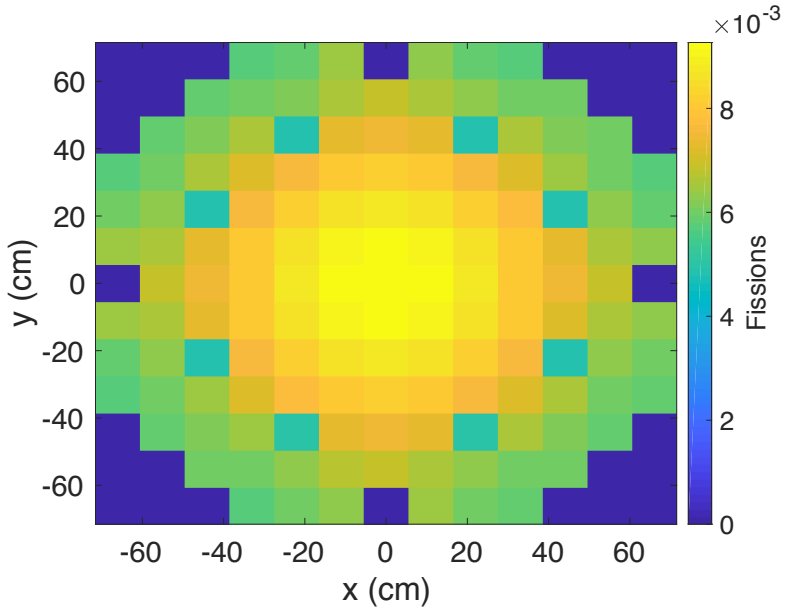

Figure **4-13**: MCC Fission Distribution with 594.19 K Peak Realistic Temperature Distribution

Since the maximum temperature is approximately 594.19 K, this core model has a relatively small temperature gradient. We can notice that in this temperature distribution; the hotter cells have the most fissions. This due to their location in the center of the core allowing them to receive the most fissions from cells surrounding them. The parameters calculated for comparison between the fission matrix and Monte Carlo simulation are listed in Table 4-4.

Table **4-4**: TREAT Arbitrary Temperature Distribution Fission Matrix Data with 594.19 Peak Temperature.

K_{FM}	0.98012
RMSE	1.00 %
Serpent uncertainty RMS (%)	0.17 %
K_{MC}	0.98021
K_{MC} relative uncertainty (pcm)	18.75
K_{FM} error (pcm)	42.72

The RMSE in Table 4-4 is about 1%, a relatively low value. This indicates that the fission matrix works well for a realistic temperature distribution with a relatively small temperature gradient. The K_{FM} error of 42.72 pcm is higher than the linear temperature profiles, but still small

since it is under 45 pcm. The difference in fission is plotted in Figure 4-14. Figure 4-15, meanwhile, displays the fission difference distribution.

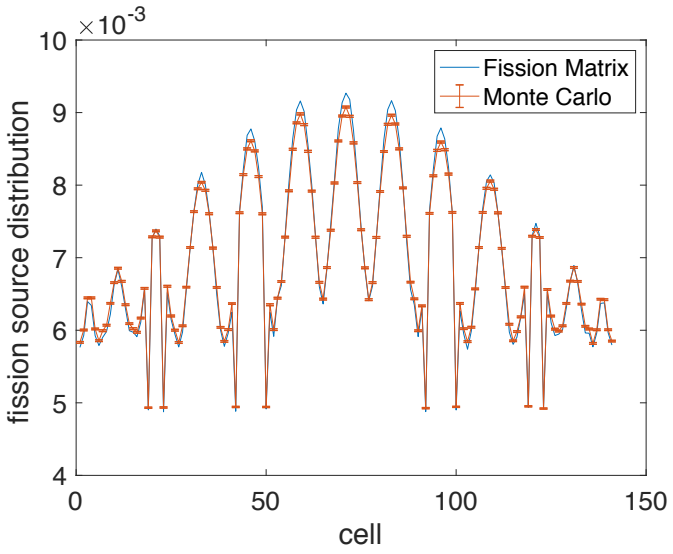

Figure **4-14**: MCC Fission Matrix and Monte Carlo Realistic Fission Distribution with 594.19 K Peak Temperature

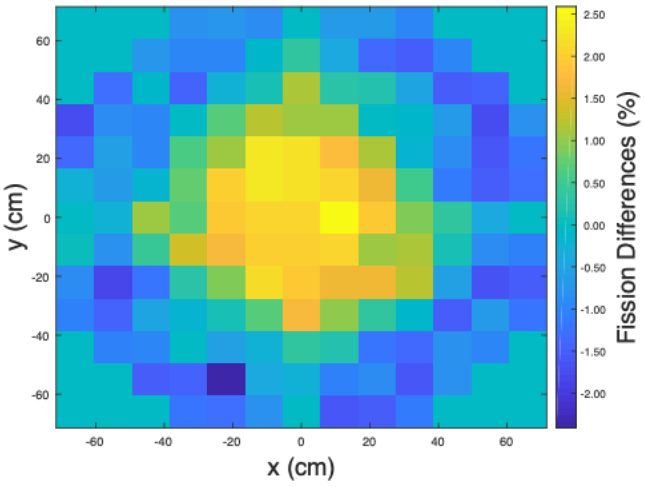

Figure **4-15**: MCC Fission Distribution difference (FM-MC) for Realistic Temperature Distribution with 594.19 K Peak Temperature

As indicated in Figures 4-14 and 4-15, overestimation and underestimation occur in cells in the center and near the edges, respectively. This is another result of the central cells receiving more fissions from the outer cells. However, the difference in fissions are relatively low. These

results further convince us that we can make accurate static calculations for a TREAT reactor using the fission matrix. Since the fissions are proportional to power, we can accurately calculate the spatial power distribution. These convincing results, however, mostly apply to cores with low temperatures. We also studied a case for an arbitrary temperature distribution with a peak of 1035.3 K to test the limits of the fission matrix. The fission distribution for this model is shown in Figure 4-16.

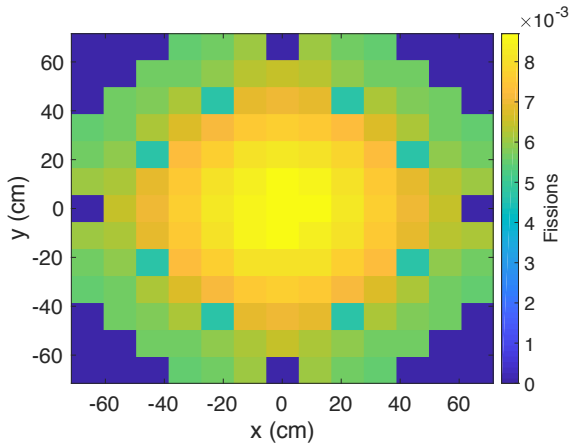

Figure **4-16**: MCC Fission Distribution with 1035.3 K Peak Realistic Temperature Distribution

Just like the model with the 594.19 K peak temperature, the core in Figure 4-16 follows a similar fission distribution pattern. Like previous models, we calculated the same set of values to validate the fission matrix. These outputs are listed in Table 4-5.

Table **4-5**: TREAT Arbitrary Temperature Distribution Fission Matrix Data with 1035.3 K Peak Temperature.

K_{FM}	0.92453
RMSE	1.60 %
Serpent uncertainty RMS (%)	0.16 %
K_{MC}	0.92413
K_{MC} relative uncertainty (pcm)	21.23
K_{FM} error (pcm)	37.56

The results show a large decrease in K_{FM} due to temperature feedback, and a relatively accurate eigenvalue calculated by the fission matrix method. The RMSE, however, does increase

to about 1.6% when raising the temperature. The fission and fission difference distribution are plotted in Figures 4-17 and 4-18, respectively.

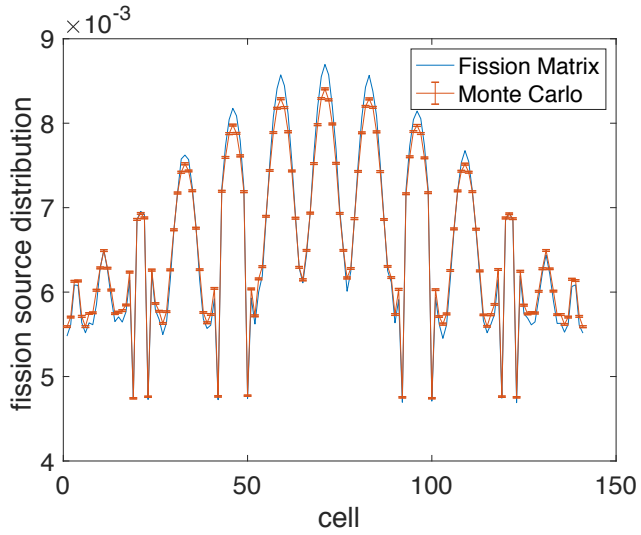

Figure **4-17**: MCC Fission Matrix and Monte Carlo Realistic Fission Distribution with 1035.3 K Peak Temperature

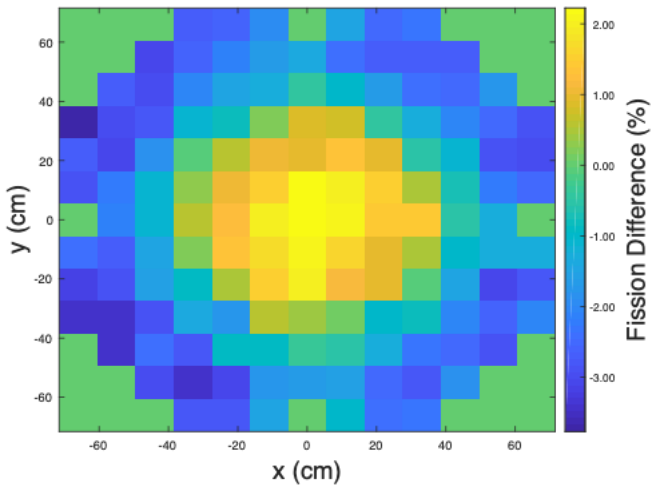

Figure **4-18**: MCC Fission Distribution difference (FM-MC) for Realistic Temperature Distribution with 1035.3 K Peak Temperature

Figures 4-17 and 4-18 once again prove that the models with the 594.19 K and 1035.3 K peak temperatures follow the same patterns for overestimations and underestimation. It is unclear

how much of the error is due to the linear interpolation, and how much is due to the combination of uniform-temperature fission matrices.

The RMSE and K_{FM} errors indicate that the fission matrix has the potential to be an accurate method for studying the statics. When large temperature gradients are present, however, some sort of correction method may need to be developed in order to get more accurate results, as has been used in other fission matrix studies [13]. Another possibility is to use more realistic temperature distributions for the fission matrix database calculations. Then, the fission matrices can be interpolated for each individual cell, using a more realistic assumption for the temperature of the surrounding cells.

4.3 Transient Calculations and Point Kinetics

4.3.1 Power and Temperature Feedback

In order to study the reactor kinetics with temperature feedback mechanism, a point kinetics as well as a quasi-static model had to be written. The point kinetics model is primarily based on reactivity insertion and it is utilized to make transient calculations. According to several uniform temperature models, the reactivity decreases as the reactor gets hotter due to temperature feedback. Point kinetics assumes that the average fuel temperature changes over time. A quasi-static model, meanwhile, takes into account the change in the spatial fuel temperature distribution shape over time. The fission matrices for point kinetics and quasi-static models were calculated by using equation 15 to interpolate, extract, and combine different uniform temperature fission matrices. We then obtained the reactivities for a quasi-static, point kinetics, and fission matrix point kinetics model. The point kinetics assumes that the uniform temperature and reactivity are linearly proportional. On the other hand, the FM point kinetics calculates the fission matrix with a

uniform average fuel temperature distribution before obtaining the reactivity. As for the quasi-static, it calculates the reactivity based on a fission matrix with a temperature distribution proportional to the power distribution. Quasi-static takes into account the energy released, according to equations 13 and 14.

For the transient operations considered here, control rods were not considered in the model. The transient simply starts at 1 W power, with the rods fully out. Additionally, only one delayed neutron group was considered, for simplicity. This limits the immediate applicability to real-world transients at TREAT, but allows for initial testing of the methodology.

Once the reactivity, at a certain time step during operation, was obtained, equations 7 and 8 were used to calculate the power distributions at that time. We plugged the values in Table 4-6, which were calculated by Serpent, into those equations in order to calculate the power distributions.

Table **4-6**: Point Kinetics Constants

Point Kinetics Constant Values	
Delayed Neutron Fraction β	0.00743
Decay Constant λ (s^{-1})	0.736
Neutron Generation Time (s)	9.2302×10^{-4}

The power distribution shape was obtained for the quasi-static model. An average nominal power was calculated for the FM point kinetics before using it to obtain the average temperature. For both the quasi-static and FM point kinetic models, an initial uniform fuel temperature distribution model of 300 K was assumed. And as for the point kinetics, the powers at all the time steps were directly calculated from the set of reactivities that were assumed to be linearly related to the average temperatures. An initial average temperature of 300 K and its corresponding Serpent reactivity of $3.90 was assumed. Equations 13 and 14 were used to calculate the temperatures as well as the temperature distribution shape for the quasi-static model. The core temperature, uniform or non-uniform, was then used to calculate new fission matrix and

reactivity at the next time step. The temperature distribution for the quasi-static model was calculated based on the fission matrix eigenvector, or fission distribution, at each time step. The fission distribution would be plugged into equation 13 in order to precisely obtain the energy released in each cell. Thus, the energy distribution causes the temperature distribution shape to change at each time step. A uniform average fission distribution, meanwhile, would be used to make similar calculations for the FM point kinetics model. The fission distribution being uniform is what leads to the temperature distribution shape remaining static over time. No eigenvector is used for the point kinetics model since the reactivities and average temperature are assumed to be linearly related. A time step of 1 x 10^{-4} seconds was used in equations 7, 8, 13 and 14. In addition, a total time interval of approximately 2 seconds was chosen for the entire transient calculation. Therefore, we had to implement 20000 timesteps, thus obtain 20000 different fission matrix point kinetics model. The quasi-static shape, however, was calculated at every 100 timesteps. Figure 4-19 shows the change in reactivity for all three models as a function of time.

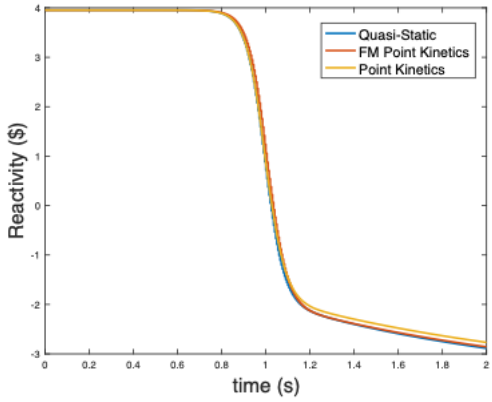

Figure **4-19**: TREAT Transient Reactivity for Quasi-Static, FM Point Kinetic, and Point Kinetic models

Figure 4-19 indicates that before operation began, a large positive reactivity of approximately $3.90 was inserted at an initial temperature of 300 K. As the temperature increases, the negative feedback causes the reactivity to decrease.

As noticed in Figure 4-19, the FM point kinetics model has the lowest reactivity. This is due the fact that it reaches its lowest reactivity at a higher average temperature than the quasi-static and FM point kinetics model. In addition, the reactivities for the quasi-static and FM point kinetic models are shown not to be decreasing in a smooth curve. These curve behaviors are a result of the reactivities being calculated at every 50 time steps for those two models, in order to save computational expense. The kinetics equations are still solved at every timestep. The reactivity for all three models are plotted as a function of temperature in Figure 4-20.

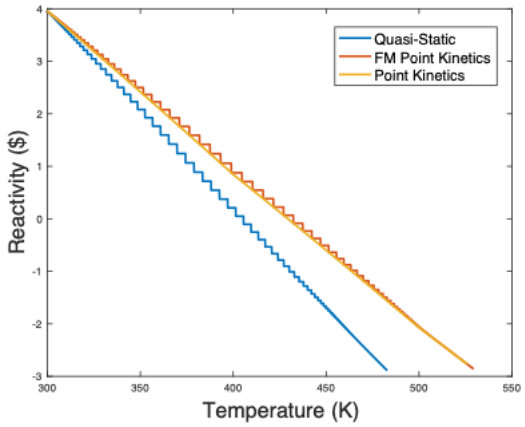

Figure **4-20**: TREAT Reactivity vs Average Temperature for Quasi-Static, FM Point Kinetic, and Point Kinetic models

As noticed in Figure 4-20, the lowest reactivities for the point kinetics and FM point kinetic models occur at a temperature above 500 K. This leads to more decreases in reactivity. Since the FM point kinetics reactivities are calculated based on a uniform temperature distribution, they follow a similar pattern to those corresponding to the point kinetics model. In the quasi-static model, the same average temperature will have a higher peak temperature in the middle, which results in a lower reactivity. At 300K, however, all models have the same temperature distribution, and the reactivities are equal.

The average power outputs during the transient are plotted in Figure 4-21 both on a regular and logarithmic scale. The total energy, pulse width, and peak power are listed in Table 4-7.

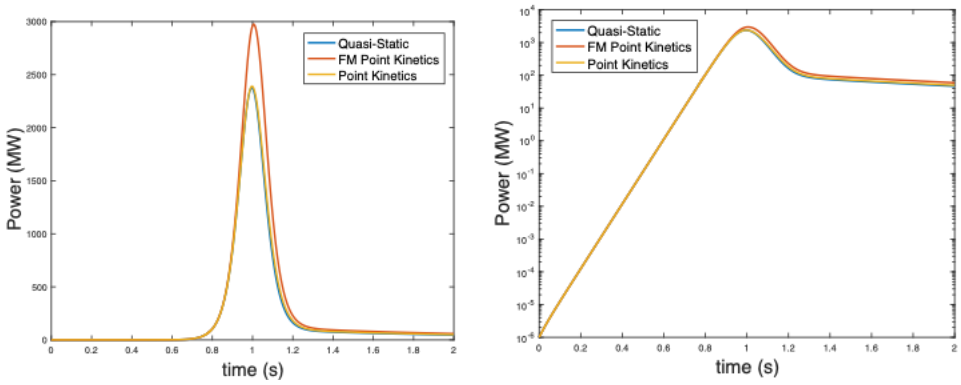

Figure **4-21**: TREAT Transient Total Power for Quasi-Static, FM Point Kinetic, and Point Kinetic models

Table **4-7**: TREAT Power Pulse Data

Model	Pulse Width (s)	Total Core Energy Deposited (MJ)	Total Power Peak (MW)
Quasi-Static	0.1516	472.25	2376.3
FM Point Kinetics	0.1523	592.39	2975.9
Point Kinetics	0.1593	486.96	2390.5

The pulse width for the point kinetics model that the power reaches its peak at a slightly later time than for the quasi-static and FM point kinetics. The power produced in Figure 4-21 is a result of an initial reactivity insertion of \$3.90, for a temperature-limited transient. This lead to a pulse width between 150-160 ms, which is similar to other TREAT transient pulse widths (~80-500 ms, depending on total energy deposited) [27]. TREAT pulse width is related to the energy deposited, with a larger reactivity insertion resulting in a decrease in pulse width.

Figures 4-17 and 4-19 shows that while the reactivity is positive, the power increases until it reaches its peak. When the reactivity becomes zero, the reactor is at the maximum power. The flux and power, however, begin to decrease when the reactivity becomes negative due to temperature feedback. Since the power also gives us the energy depositions, which are listed in

Table 4-7, it is used to calculate the temperatures using equations 13 and 14. The TREAT temperature for all three models are plotted in Figure 4-22.

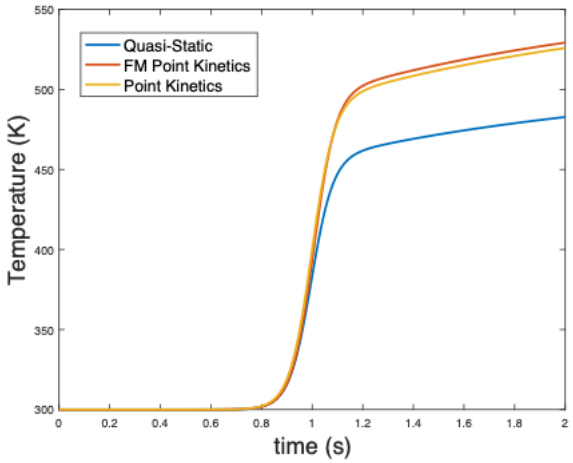

Figure **4-22**: TREAT Transient Average Temperature for Quasi-Static, FM Point Kinetic, and Point Kinetic models

As noticed in Figures 4-21 and 4-22 for all three models, the powers and average temperatures for TREAT are proportional to each other since they are directly related to the energy outputs. Also, the TREAT temperature never decreases since the process is assumed to be adiabatic (i.e., there is no heat transfer out of the system). This is realistic given the short time-scale of the pulse and the low conductivity of graphite. Once again, the temperatures are a result of such reactivity insertion that leads to small pulse width. A smaller reactivity insertion would lead to an increase in pulse width, energy and temperature [27]. Fuel temperature calculations requires the thermal properties of graphite, which were assumed to be constant, since the fuel assembly is assumed to be mostly composed of graphite. In addition, the graphite volume, according to the benchmarks, is shown to have little to no change in density or volume as a result of temperature increase [1]. Table 4-8 lists the graphite thermal properties used in equations 13 and 14 to calculate fuel temperature for all three models.

Table **4-8**: Graphite Thermal Properties [2]

Thermal Properties of Graphite	
Density (g/cm^3)	1.67
Heat Capacity (J/g*k)	0.72
Fuel Assembly Length (cm)	10.16
Fuel Assembly Width (cm)	10.16
Fuel Assembly Height (cm)	123.19
Number of Fuel Assemblies	141
Total fuel Volume (cm^3)	1,692,000

The total fuel volume in Table 4-8 was calculated by using Equation 18:

$$V_f = N_A * L * W * H \quad (18)$$

In equation 18, N_A is the total number of fuel assemblies, L is the length, W is the width, and H is the height. Note that this approximates the geometry as rectangular prism and ignores the corners as shown in Figure 3-2. The volume of all the fuel assemblies in the core can be used to calculate the reactor temperature output. However, it is important to notice that the point kinetics model calculates power without incorporating change in spatial flux distribution shape. The flux is what determines the spatial power distribution shape. Therefore, the fission matrix end-method was used to calculate the flux distributions for the FM point kinetics and quasi-static models. For the quasi-static model, these flux distributions were used to obtain the power and temperature distribution steps at certain selected time steps.

4.3.2 Power and Temperature Distribution Calculation

The power and temperature distribution are directly calculated from the fission matrix eigenvector. This allows us to validate the fission matrix end method since the eigenvector represents the fission distribution. The main benefit of the fission matrix method was that it allows us to observe the change in the fission distribution shape at every time step, which allowed

us to display certain characteristics of a quasi-static model. In order to implement the fission matrix method, the MATLAB code in Appendix B was written to calculate eigenvectors, or fission distribution, power and temperature distributions at each time step. This method involves using the end-method to interpolate, extract and combine fission matrices. In order to begin this process, we assumed that the initial TREAT MCC model consisted of a 300 K uniform temperature. We then used the initial 300 K uniform temperature model to calculate an initial fission matrix, eigenvector, and reactivity. The eigenvector was then used to get the power distribution. This process was carried out for both the quasi-static and fission matrix point kinetics model. An initial power of 1 W was assumed for all transients.

These power distributions were then used to calculate an arbitrary temperature distribution for the quasi-static model utilizing equations 13 and 14. An average fuel temperature would be calculated, and it would be used as a nominal uniform fuel temperature for the FM point kinetics model. These newly obtained uniform and arbitrary fuel temperature distributions were then used to calculate new fission matrices using equation 15 for the two models at the next time step. A separate function implementing equation 15 to obtain a fission matrix at every time step is shown in Appendix C. The fission matrices for the FM point kinetics and quasi-static models were then used to repeat the process for fission, power and temperature distribution calculations. This process was repeated at every time step during the 2 second operation time.

4.3.3 Calculation Results and Analysis

The arbitrary power and temperature distributions of the TREAT MCC quasi-static model are displayed in Figures 4-23 to 4-26. However, only certain average nominal power levels, obtained by the fission matrix, were selected to evaluate these distribution shapes. They

include the first full width half maximum (FHWM), the maximum power, the second FWHM and the final power.

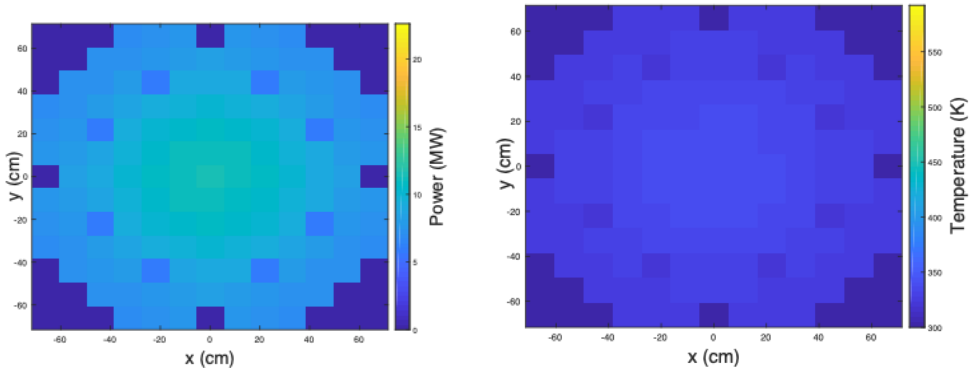

Figure **4-23**: MCC Quasi-static Power and Temperature Distribution at the First FWHM

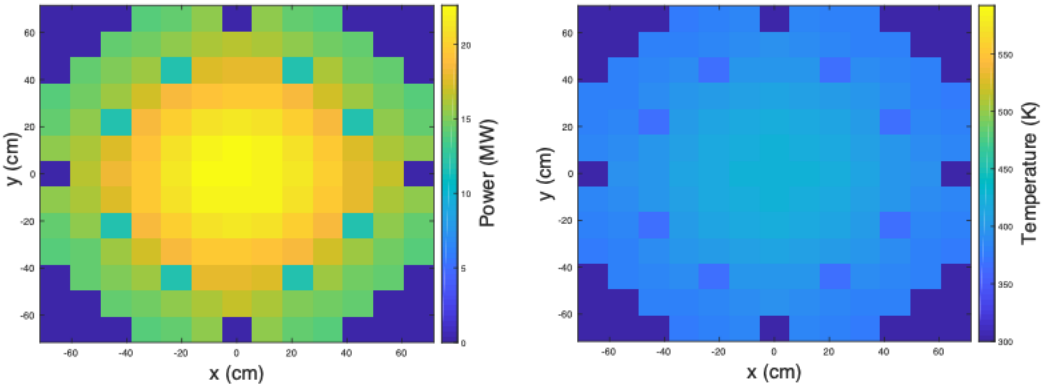

Figure **4-24**: MCC Quasi-static Power and Temperature Distribution at the Maximum Power

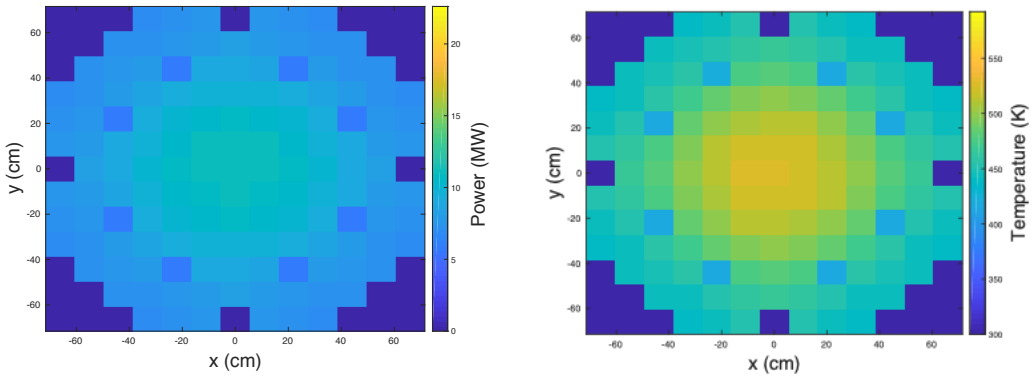

Figure **4-25**: MCC Quasi-static Power and Temperature Distribution at the Second FWHM

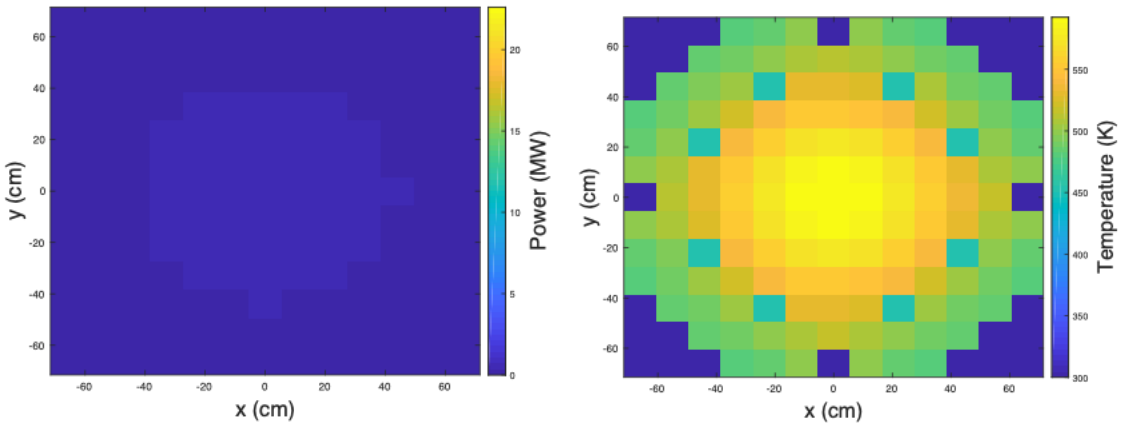

Figure **4-26**: MCC Quasi-static Final Power and Temperature Distribution

The FWHM and maximum quasi-static power and temperature distributions in Figures 4-23 and 4-24 are obtained when the average power is increasing, and reactivity is positive. Figures 4-25 and 4-26 consist of the power and temperature distributions calculated when the average power is decreased. As the average power decreases with time, the flux distribution shape changes. This affects the temperature of the edge cells. With the negative reactivity leading to a decrease in flux, the fission distribution is shaped such that the fuel cells near the edge have more fission compared to the ones in the point kinetic models.

Temperature feedback is caused by change in the spectrum of the system that leads to higher parasitic absorption and leakage as a result of lower total fissions compared to the initial state immediately after the initial insertion of positive reactivity prior to any temperature change [18]. The change in spectrum leads to lower amount of energy deposited into the cells. In addition, leakage from the center cells cause more neutrons to accumulate into the outer cells at negative reactivity.

Once these realistic temperature distributions were obtained, we were able to calculate the fission matrices at these power levels. This was conducted using the fission matrix function file in Appendix C. Once the quasi-static fission matrices at the corresponding power levels different were obtained, we calculated source distributions and the eigenvalues or effective multiplication values K_{eff}. The reactivities ρ for the fission matrices, as a function of K_{eff}, were calculated using equation 19.

$$\rho = \frac{K_{eff}-1}{K_{eff}} \tag{19}$$

The reactivities for the quasi-static arbitrary fuel temperature distribution models displayed in Figures 4-23 to 4-26 are listed in Table 4-9.

The values are presented in $ worth. In addition, in an attempt to validate the fission matrix method, we calculated reactivities for the FM point kinetic models. We then selected reactivities obtained at the power levels where the quasi-static reactivities were calculated. The FM point kinetic reactivities are listed in Table 4-9.

Table **4-9**: TREAT Quasi-static Reactivities at Different Power levels

Power Level	QS Reactivity ($)	FM PK Reactivity ($)	PK Reactivity ($)
Initial	3.91	3.91	3.91
FHWM 1	3.13	3.13	3.08
Maximum	0	0	0
FHWM 2	-1.20	-1.27	-1.16
Final	-2.89	-2.86	-2.77

According to Table 4-9, the difference in reactivity is proportional to the power level. It increases while the reactivities remain positive decreases when the reactivity becomes negative. This is due to the quasi-static model allowing the power distribution to change.

On the other hand, when talking about fission distribution difference, it is determined by local cell temperatures. For the quasi-static model, the fission differences are inversely proportional to the temperature. While the reactivity and power level determine the fission distribution shape, the temperature determines the local cell fissions. A higher temperature leads to a smaller reactivity and number of fissions. Since the FM point kinetics model treats the fuels as having a uniform temperature while the quasi-static model takes into account the actual temperature distribution, we would expect the quasi-static model cells to have less fissions, or a negative difference, in the hotter cells and more fissions, or positive difference, in the colder cells than the FM point kinetics. The main advantage of the quasi-static over point kinetics model is that it does take into account the physical temperature distribution. The difference in the fission distributions between the quasi-static and FM point kinetics model were calculated to further validate the fission matrix for studying the TREAT kinetics. These fission error distributions were calculated by subtracting the quasi-static minus the point kinetic fissions, and are displayed in Figures 4-27- to 4-30.

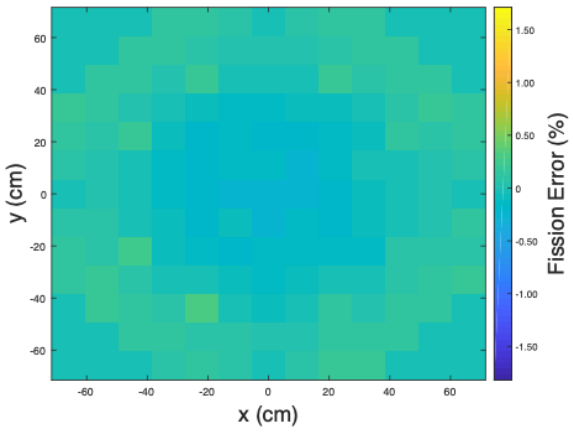

Figure **4-27**: Fission Error % between fission rate for quasi-static and point kinetic methods at the First FWHM

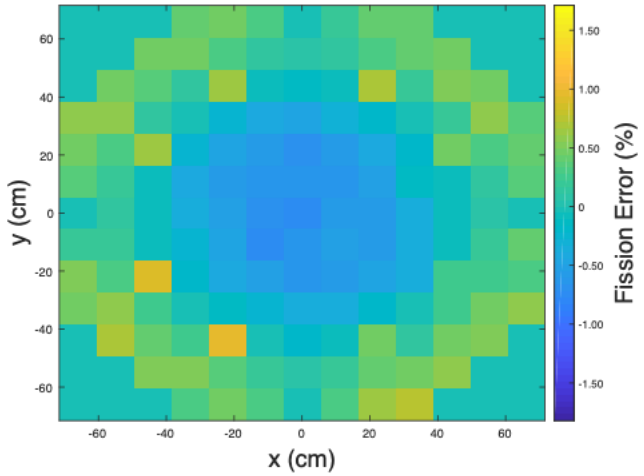

Figure **4-28**: Fission Error % between fission rate for quasi-static and point kinetic methods at the Maximum Power

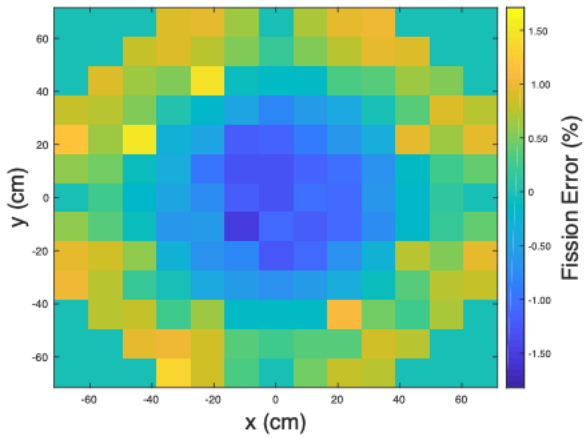

Figure **4-29**: Fission Error % between fission rate for quasi-static and point kinetic methods at the Second FWHM

Figures 4-27 to 4-29, show a more significant spread in fission difference. This is due to the re-shaping of the fission distribution due to the changing temperature distribution. If we were

to calculate the average flux of the quasi-static model, its value would be close if not equal to that of the FM point kinetics. The same can be said for the temperatures. However, the temperature distribution shape indicates that there is a major difference in some regions. In the center of the reactor, the higher temperature causes less fissions to occur (correctly) in the quasi-static model. In the PK model, the power distribution is fixed, so the power does not redistribute itself to the outside.

As noticed in Figure 4-30, the quasi-static and FM point kinetic nominal peak powers differ. This difference indicates that the changing spatial power distribution has an effect on the reactivity, and thus the accuracy of the PK approximation. Temperature feedback and smaller reactivity causes the quasi-static power pulse to be lower than the FM point kinetics. This does not guarantee, however, that the end method will precisely calculate the spatial temperature distribution peak at every time step since the flux shape changes. Furthermore, the peak power differences and fission error distributions indicate that the error will always be within at least 2%. This further demonstrated in Figure 4-30 when the fission error distribution is calculated at the final power.

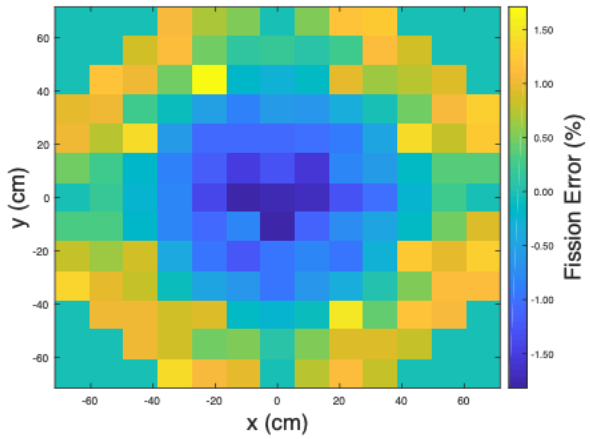

Figure **4-30**: Fission Error % between fission rate for quasi-static and point kinetic methods at the Final Power

The fission differences don't return to their original values before operation began since the reactor is not experiencing shutdown. Further comparison between the quasi-static and point kinetics data are listed in Table 4-10. These values were obtained treating the parameters for the FM point kinetics solution as the true values.

Table **4-10**: TREAT Transient Fission Matrix Data

Power Level	K_{QS}	RMS (%)	RMS Uncertainty (pcm)	K_{FMPK}	K_{FMPK} Uncertainty (pcm)	K_{QS} Error (pcm)
Initial	1.02861	0	179.8	1.02861	19.1	0
1st FWHM	1.02243	0.12	180.1	1.02263	20.2	19.5
Maximum	1.00000	0.38	169.9	1.00000	17.9	0
2nd FWHM	0.99127	0.67	159.5	0.99159	23.4	40.3
Final	0.98014	0.81	158.7	0.98019	16.6	10.2

As noticed in Table 4-10, the Keff Error increases as the power level does. The fission distribution RMSE rises continuously during the operation, also demonstrated in the fission difference distribution plots. Further displays of the quasi-static and FM point kinetic fission distributions are displayed in Figures 4-31 to 4-35.

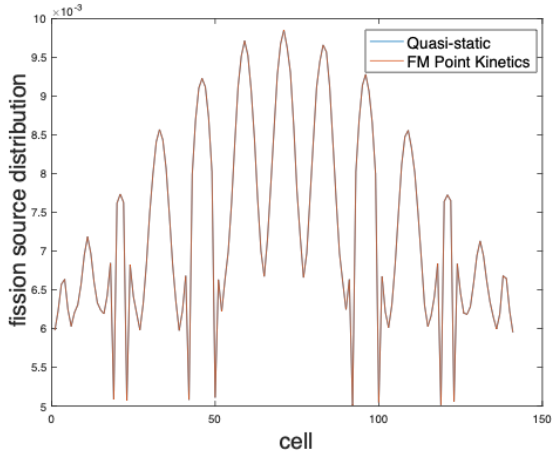

Figure **4-31**: Quasi-static and FM Point Kinetic Fission Distributions at the Initial Power

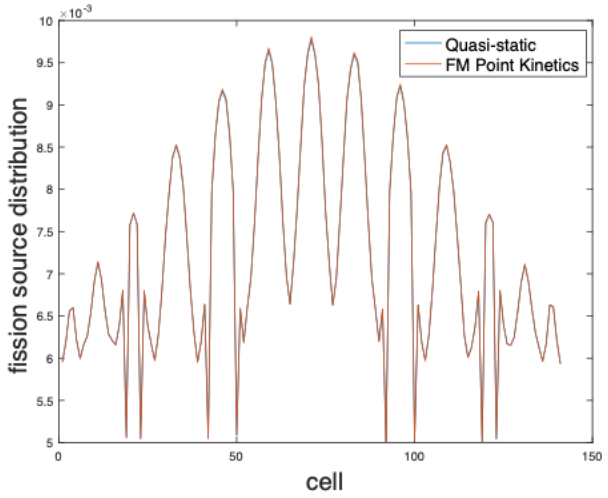

Figure **4-32**: Quasi-static and FM Point Kinetic Fission Distributions at the First FWHM

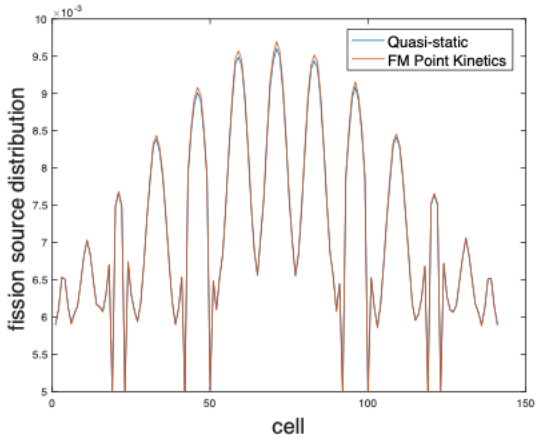

Figure **4-33**: Quasi-static and FM Point Kinetics Fission Distributions at the Maximum Power

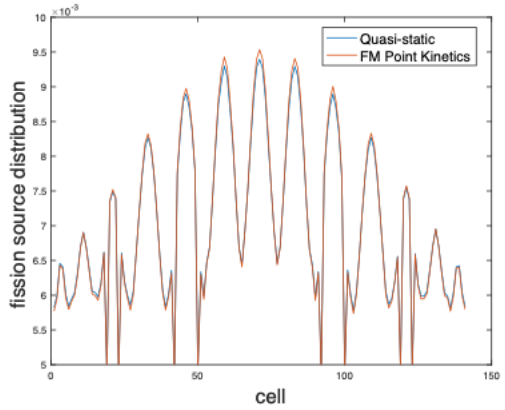

Figure **4-34**: Quasi-static and FM Point Kinetics Fission Distributions at the Second FWHM

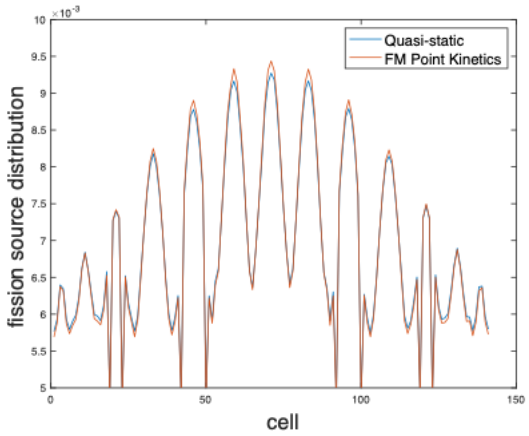

Figure **4-35**: Quasi-static and FM Point Kinetics Fission Distributions at the Final Power

The plots displayed in Figures 4-31 to 4-35 further confirm increases in error during transient operation as the reactor gets hotter. They are consistent with the fission difference distributions. Besides the plot in Figure 4-31, most of the eigenvectors obtained by quasi-static calculations differ slightly from the FM point kinetic values, most noticeably in the middle hottest cells and colder fuel cells near the edges. This is expected since the quasistatic temperature will be higher in the middle, thus lowering local reactivity, and the opposite in the outside.

Based on the results listed above, we can conclude that the fission matrix method can be used for making quasi-static calculations during a transient operation. The quasistatic approach should be more accurate that a point kinetics approach due to the ability to adjust the reactivity based on the actual temperature distribution, as opposed to a simple average. For the cases examined, the effect up to about 2% for local temperature distributions, and a change of 20% percent in the total energy deposited.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, the fission matrix combination and interpolation technique is a fast and efficient method for calculating the eigenvalue and power distribution in the TREAT reactor with temperature feedback. There has been past work done to further prove its effectiveness and validity for a variety of reactors and applications. However, there are certain factors that can limit the reliability of this methodology. An arbitrary temperature distribution can lead to some errors, as was the case for TREAT. Models with non-uniform properties normally present themselves with error in the certain areas of the core where there the power and temperature may be above or below the average value. Interpolations in such core regions can lead to overestimation or underestimation of fission matrix elements due to linear interpolation being used in the end-method. For a realistic temperature distribution (with a maximum temperature of 594 K), a difference of 42.72 pcm and 1.00% RMS was noted between the interpolated fission matrix model and the standard Serpent Monte Carlo model.

In addition to studying the reactor statics, the fission matrix method was investigated analyzing the TREAT kinetics with temperature feedback mechanism. More specifically, it showed to be effective in making quasi-static calculations for changes in fission distribution shape for non-uniform temperature models. Differences between point kinetics and quasi-static for a simple transient were 19.49% in total energy, 14.88% in maximum power, and 8.33% in average fuel temperature. Like in the case for studying reactor statics, temperature distributions obtained during a transient calculation lead to some small errors during the interpolation.

Nevertheless, the fission matrix also proved itself to have high value in examining certain aspects of a transient operation that the point kinetics would not be capable. This was the case with the change in the flux distribution shape.

5.2 Future Works

There are a few things that can be done to make the fission matrix calculations more realistic. One is the use of 3D temperature distributions and fission matrices, as opposed to the 2D ones used in this work. Another is the use of more delayed energy groups, as opposed to the single group used here. The addition of control rods to the fission matrix database would also allow comparison with experimental data such as the M8CAL series.

Among some future works include performing a Monte Carlo transient simulation for the TREAT reactor to compare results. This will help to further validate the fission matrix method for studying TREAT kinetics with temperature feedback. However, Monte Carlo transient simulation faces the obstacle of being very computationally challenging, as a consequence of preventing the requirement to conduct multiple simulations. Nevertheless, transient Monte Carlo simulations have the potential to take into account spatial dependence, a capability that the point kinetics method lacks. Besides a standard Monte Carlo transient, comparisons could be made to a time-dependent Monte Carlo code T-Rex, that uses the Improved Quasi-Static method with KEVOa. and KEVO-VI models.

References

[1] J. D. (Idaho N. L. Bess, M. D. (Argonne N. L. DeHart, T. (University of M. Downar, and V. (University of M. Seker, "Benchmark Specifications for the TREAT Minimum Critical and M8CAL Cores," 2016.

[2] J. Bess and M. D. DeHart, "Baseline Assessment of TREAT for Modeling and Analysis Needs," Idaho National Laboratory. Nuclear Systems Design and Analysis., Idaho Falls, Idaho 83415, 2015.

[3] Z. Haining, "Uncertainty and Quantification and Sensitivity Analysis of the TREAT Minimum Critical Core," ANS Int. M&C Conference, Korea, 2017.

[4] T. J. Topham, A. Rau, and W. J. Walters, "An Iterative Fission Matrix Scheme for Calculating Steady-State Power and Critical Control Rod Position in a TRIGA Reactor." Manuscript submitted for publication., Department of Mechanical and Nuclear Engineering, Penn. State Univeristy, 132 Reber Bldg., State College, PA 16802, USA, pp. 1–12, 2019.

[5] T. J. Topham and W. J. Walters, "Temperature Feedback Using the Fission Matrix for a TRIGA Reactor," *Reactor Analysis Methods III*, vol. 119. Transactions of the American Nuclear Society, Orlando, Florida, pp. 1233–1235, 2018.

[6] T. Kitada, Takanori, Takeda, "Effective Convergence of Fission Source Distribution in Monte Carlo Simulation," *J. Nucl. Sci. Technol. 385*, pp. 324–329, 2001.

[7] A. Laureau, M. Aufiero, P. R. Rubiolo, P. R. Merle-Lucotte, and D. Heuer, "Transient Fission Matrix: Kinetic calculation and kinetic parameters beff and Keff calculation," *Annals of Nuclear Energy*. Elsevier Ltd., LPSC, Université Grenoble-Alpes, CNRS/IN2P3, 53 rue des Martyrs, F-38026 Grenoble Cedex, France, pp. 1035–1044, 2015.

[8] J. Dufek and W. Gudowski, “Fission matrix based Monte Carlo criticality calculations,” vol. 36. Elsevier Ltd., Royal Institute of Technology, Department of Reactor Physics, AlbaNova University Center, 10691 Stockholm, Sweden, pp. 1270–1275, 2009.

[9] J. Dufek and W. Gudowski, “Stability and convergence problems of the Monte Carlo fission matrix acceleration methods,” *Annals of Nuclear Energy*, vol. 36. Elsevier Ltd., pp. 1648–1651, 2009.

[10] A. Laureau, D. Heuer, E. Merle-Lucotte, P. . Rubiolo, M. Allibert, and M. Aufiero, “Transient coupled calculations of the Molten Salt Fast Reactor using the Transient Fission Matrix approach,” *Nuclear Engineering and Design*. Elsevier Ltd., LPSC, Université Grenoble-Alpes, CNRS/IN2P3, 53 rue des Martyrs, F-38026 Grenoble Cedex, France, pp. 112–124, 2017.

[11] J. F. Freund, G. A., Elias, P., MacFarlane, D. R., Geier, J. D., Boland, “Design Summary Report on the Transient Reactor Test Facility (treat).” Argonne National Laboratory, Reactor Engineering Division, p. ANL-6034, 1960.

[12] W. J. Walters, “Application of the RAPID Fission Matrix Methodology to 3-D Whole-core Reactor Transport,” *Proc. M&C 2017*. Jeju, Korea, 2017.

[13] D. He and W. J. Walters, “A New Fission Matrix Correction Method to Estimate the Source Distribution in Nuclear Reactor Core,” *Proc. PHYSOR 2018*. American Nuclear Society (2018). (Submitted), 2018.

[14] A. Rau, D. He, and W. J. Walters, “Application of Fission Matrix Correction Method to TRIGA Reactor Core,” *Reactor Physics Design, Validation, and Operational Experience*, vol. 118. Transactions of the American Nuclear Society, Philadelphia, Pennsylvania, pp. 1014–1017, 2018.

[15] J. W. Nielsen, D. W. Nigg, and A. W. LaPorta, “A fission matrix based validation protocol for computed powerdistributions in the advanced test reactor,” *Nuclear*

Engineering and Design. Elsevier Ltd., pp. 615–624, 2015.

[16] K. Ott and D. A. Meneley, "Accuracy of the Quasistatic Treatment of Spatial Reactor Kinetics," *Nuclear Science and Engineering*, vol. 3, no. 36. Argonne National Laboratory, Argonne, Illinois, pp. 402–411, 1969.

[17] Z. Mausolff, M. DeHart, and S. Goluoglu, "Enhanced geometric capabilities for the transient analysis code T-ReX and its application to simulating TREAT experiments," *Progress in Nuclear Energy*, no. 105. Elsevier Ltd., pp. 236–246, 2018.

[18] S. Goluoglu and M. D. DeHart, "On the effects of pre- and post-transient rod positions for TREAT temperature-limited transient powers," *Nuclear Engineering and Design*, vol. 331. Elsevier B.V., Nuclear Engineering Program, University of Floridam 549 Gale Lemerand Dr, Gainsville, FL 32611-6400, United States. Reactor Physics Design and Analysis Department, Idaho National Laboratory, 2525 North Freemont St, Idaho Falls, ID 83415, United States, pp. 97–102, 2018.

[19] J. H. Beam, C.H., McCuaig, F.D., Hanwreck, "The Fabrication of the Fuel Elements for the Transient Reactor Test." Argonne National Laboratory, 1959.

[20] E. Meijeting, "A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing," *Proc. IEEE*, vol. 90, no. 3, pp. 319–342, 2002.

[21] J. D. Bess and M. D. Dehart, "TREAT Fuel Assembly Characterization for Modern Neutronics Validation Methods," Trans. American Nuclear Society, Idaho Falls, Idaho 83415, 2015.

[22] R. E. Nightingale, *Nuclear Graphite*. New York, NY: Academic Press Inc., 1962.

[23] "USG GLEDCO - Engineered Carbon, Graphite, Manufacturer." U.S. Graphite Inc., 2003.

[24] H. M. Connaway *et al.*, "Analysis of the TREAT LEU Conceptual Design." Argonne National Laboratory, Nuclear Engineering Division, Argonne, Illinois, 2016.

[25] H. Ayktekin and Y. Akcin, "Characterization of borided Incoloy 825 alloy." Elsevier Ltd.,

Afyon Kocatepe University, Faculty of Technology, Metallurgy and Materials Engineering Department, Turkey, pp. 515–523, 2013.

[26] J. Wisniak, “The Composition of Air. Discovery of Argon,” *Educ. Química*, vol. 18, no. 1, p. 69, 2018.

[27] J. Bess *et al.*, “Narrowing transient testing pulse widths to enhance LWR RIA experiment design in the TREAT facility,” *Annas of Nuclear Energy*, no. 124. Elsevier Ltd., Idaho National Laboratory, Idaho Falls, ID, pp. 548–571, 2018.

Appendix A

Matlab source distribution convergence input file

```
clc;clear
close all

treatmc_DT_det0
TREAT_Gr
%treatBenchsm_fission matrixtx0
treatmc_DT_res

e=eigs(F_new,1); %eigenvalue
[V,D]=eigs(F_new,1); %eigenvector (V) and diagonal eigenvalue (D)
Vnorm0=(V./sum(V)).*e; %normalized eigenvector such that the sum of the eigenvectors is equal to the eigenvalue, with zeros
Vref0=DETxy(:,11);

Vnorm=nonzeros(Vnorm0); %normalized eigenvector matrix without zeros
Vref=nonzeros(Vref0); %reference eigenvector without zeros (calculate by zeros)

A=((Vnorm-Vref)./Vref).^2; %term used in RMS equation
RMS_V=((1./length(Vnorm0)).*sum(A)).^(1/2); %root mean square of relative difference between normalized and reference eigenvectors

U0=DETxy(:,12); %uncertainty matrix with zeros
U=nonzeros(U0); %uncertainty matrix without zeros
U2=U.^2; %squared uncertainties
RMS_U=((1./length(U)).*sum(U2)).^(1/2); %root mean square of uncertainties

N=1:length(Vnorm);
N=N(:);
figure
plot(N,Vnorm)
title('eigenvector')
xlabel('value number');ylabel('eigenvector')
legend('normalized','reference')
hold on
err=U.*Vref;
errorbar(N,Vref,err)
title('eigenvector')
xlabel('spatial location');ylabel('fission source')
legend('normalized','reference')

DiffD=abs(IMP_KEFF(1)-D)/IMP_KEFF(1); %fission matrix eigenvalue error
DiffUD=IMP_KEFF(2); %eigenvalue uncertaint
```

Appendix B

Matlab Point Kinetics and fission matrix combination input file

```
clear all; clc; close all;
TREATpointkineticscomplete %matlab code for obtaining Serpent data
temperature
treatT1DT_out %serpent data output at the initial data temperature

%% Serpent Data fission matrix
%EigenV=ones(169,1);
% zetaQS(:,1)=[betaT/lambda1G;1];
% zetaFMPK(:,1)=[betaT/lambda1G; 1];

aUp=zeros(length(TdataOfficial),n,m); %Serpent Data Fission Matrices

for j=1:n;
    for k=1:m;
        aUp(1,j,k)=fmtx1(j,k);
        aUp(2,j,k)=fmtx2(j,k);
        aUp(3,j,k)=fmtx3(j,k);
        aUp(4,j,k)=fmtx4(j,k);
        aUp(5,j,k)=fmtx5(j,k);
        aUp(6,j,k)=fmtx6(j,k);
        aUp(7,j,k)=fmtx7(j,k);
        aUp(8,j,k)=fmtx8(j,k);
        aUp(9,j,k)=fmtx9(j,k);
        aUp(10,j,k)=fmtx10(j,k);
    end
end

%% Linear temperature distribution with higher peak
TlinearSGTD=linspace(300,1200,13); %linear temperature distribution
with small gradient
TlinSG=ones(13,1)*TlinearSGTD;  %Linear temperature distribution shape
Tlin=reshape(TlinSG,n,1);
FM_lin=zeros(n,m);

%Linear temperature distribution fission matrix combination
for j=1:n
    for k=1:m
        FM_lin(j,k)=interp1(TdataOfficial, aUp(:,j,k), Tlin(j));
    end
end

treatLinearSGTD_det0 %small gradient linear temperature distribution
Serpent detector data
treatLinearSGTD_fmtx0 %small gradient linear temperature distribution
Serpent fission matrix
```

```
treatLinearSGTD_res %small gradient linear temperature distribution
Serpent main output

[Vlin,Dlin]=eigs(FM_lin,1); %fission matrix eigenvector/fission
distribution and eigenvalue/effective multiplication factor (Keff)
Vnorm0lin=(Vlin./sum(Vlin)).*Dlin; %normalized eigenvector such that
the sum of the eigenvectors is equal to the eigenvalue, with zeros
Vref0lin=(DETxy(:,11)./sum(DETxy(:,11))).*Dlin;
Vnormlin=nonzeros(Vnorm0lin); %normalized eigenvector matrix without
zeros
Vreflin=Vref0lin(Vref0lin>1E-8);
A=((Vnormlin-Vreflin)./Vreflin).^2; %term used in RMS equation
RMS_Vlin=((1./length(Vnorm0lin)).*sum(A)).^(1/2); %root mean square of
relative difference between normalized and reference eigenvectors
U0lin=DETxy(:,12); %uncertainty matrix with zeros
Ulin=U0lin(Vref0lin>1E-8);
U2lin=Ulin.^2; %squared uncertainties
RMS_Ulin=((1./length(Ulin)).*sum(U2lin)).^(1/2); %root mean square of
uncertainties
figure
N=1:length(Vnormlin); %length of eigenvector without zeros
N=N(:);
plot(N,Vnormlin) %eigenvector/fission distribution plot
xlabel('cell');ylabel('fission source distribution')
legend('Fission Matrix','Monte Carlo')
hold on
err=Ulin.*Vreflin; %error
errorbar(N,Vreflin,err) %errorbar
xlabel('cell');ylabel('fission source distribution')
legend('Fission Matrix','Monte Carlo') %fission matrix and monte carlo
fission distribution
DiffDlin=abs(IMP_KEFF(end,1)-Dlin)/IMP_KEFF(end,1); %difference between
Serpent and fission matrix Keff
DiffUDlin=IMP_KEFF(end,2); %Serpent uncertainties
fissionlin=reshape(Vnorm0lin,[13,13]); %fission matrix fission
distribution
fissionlinMC=reshape(Vref0lin,[13,13]); %monte carlo fission
distribution
figure
image(xlattice,ylattice,fissionlin','CDataMapping','scaled') %fission
difference distribution
c=colorbar;
c.Label.String = 'Fissions';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
fissionlinerror=100*(fissionlin-fissionlinMC)./fissionlinMC; %fission
error
figure
image(xlattice,ylattice,fissionlinerror','CDataMapping','scaled') %fiss
ion error distribution
c=colorbar;
c.Label.String = 'Fission Difference (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
```

```
%% Linear temperature distribution with smaller peak
T_SPTD=linspace(300,600,13); %linear temperature distribution with
large gradient
TlinSPTD=ones(13,1)*T_SPTD;  %linear temperature distribution shape
TlinSPTD=reshape(TlinSPTD,n,1);
FM_SPTD=zeros(n,m);

%fission matrix combination
for j=1:n
    for k=1:m
        FM_SPTD(j,k)=interp1(TdataOfficial, aUp(:,j,k), TlinSPTD(j));
    end
end

treatSPTD_det0 %Serpent detector data
treatSPTD_fmtx0 %Serpent fission matrix
treatSPTD_res %Serpent main output file


[V_SPTD,D_SPTD]=eigs(FM_SPTD,1); %fission matrix eigenvector/fission
distribution and eigenvalue/Keff
Vnorm0SPTD=(V_SPTD./sum(V_SPTD)).*D_SPTD; %normalized eigenvector such
that the sum of the eigenvectors is equal to the eigenvalue, with zeros
Vref0SPTD=(DETxy(:,11)./sum(DETxy(:,11))).*D_SPTD;
VnormSPTD=nonzeros(Vnorm0SPTD); %normalized fission matrix eigenvector
matrix without zeros
VrefSPTD=Vref0SPTD(Vref0SPTD>1E-8); %normalized Serpent eigenvector
matrix without zeros
A=((VnormSPTD-VrefSPTD)./VrefSPTD).^2; %term used in RMS equation
RMS_VSPTD=((1./length(Vnorm0SPTD)).*sum(A)).^(1/2); %root mean square
of relative difference between normalized and reference eigenvectors
U0SPTD=DETxy(:,12); %serpent uncertainties
USPTD=U0SPTD(U0SPTD>1E-8); %serpent uncertainties without zeros
U2SPTD=USPTD.^2; %squared serpent uncertainties
RMS_USPTD=((1./length(U0SPTD)).*sum(U2SPTD)).^(1/2); %root mean square
difference of serpent uncertainties
figure
N=1:length(VnormSPTD); %number of cells with fissions
N=N(:);
plot(N,VnormSPTD) %fission distribution plot without zeros
xlabel('cell');ylabel('fission source distribution')
legend('Fission Matrix','Monte Carlo')
hold on
err=USPTD.*VrefSPTD; %fission distribution error
errorbar(N,VrefSPTD,err) %fission distribution errorbar
xlabel('cell');ylabel('fission source distribution')
legend('Fission Matrix','Monte Carlo')
DiffDSPTD=abs(IMP_KEFF(end,1)-D_SPTD)/IMP_KEFF(end,1); %difference
between fission matrix and Serpent Keff
DiffUDSPTD=IMP_KEFF(end,2); %Serpent Keff uncertainties
fissionSPTD=reshape(Vnorm0SPTD,[13,13]); %fission matrix fission
distribution shape
fissionSPTDMC=reshape(Vref0SPTD,[13,13]); %Serpent fission distribution
shape
```

```
figure
image(xlattice,ylattice,fissionSPTD','CDataMapping','scaled') %Serpent
fission distribution image
c=colorbar;
c.Label.String = 'Fissions';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
fissionSPTDdiff=100*(fissionSPTD-
fissionSPTDMC)./fissionSPTDMC; %fission error distribution
figure
image(xlattice,ylattice,fissionSPTDdiff','CDataMapping','scaled') %fiss
ion error distribution image
c=colorbar;
c.Label.String = 'Fission Difference (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')



%% Serpent Reactivities

keff=[keff1 keff2 keff3 keff4 keff5 keff6 keff7 keff8 keff9
keff10]; %serpent data reactivities

%% Quasi-static, FM Point Kinetics, and Point Kinetics transient
calculations

DT=1E-4; %timestep
Length=2/DT; %number of timesteps
tL=linspace(0,2,Length); %total transient calculation time
TemperatureQS=300*ones(n,1); %initial quasi-static temperature
distribution
PowerQS=zeros(n,Length); %pre-allocated quasi-static power distribution
PowerFMPK=zeros(n,Length); %pre-allocated FM point kinetics power
distribution
rhoqs=zeros(1,Length); %pre-allocated quasi-static reactivity
TempMaxQS=zeros(1,Length); %pre-allocated quasi-static maximum
temperatures
TempAvgQS=zeros(1,Length); %pre-allocated quasi-static average
temperatures
TempDisQS=zeros(n,Length); %pre-allocated quasi-static temperatures
distributions

TemperatureFMPK=300*ones(n,1); %initial FM point kinetics temperature
distribution
rhofmpk=zeros(1,Length); %pre-allocated FM point kinetics reactivity
TempMaxFMPK=zeros(1,Length); %pre-allocated FM point kinetics maximum
temperatures
TempAvgFMPK=zeros(1,Length); %pre-allocated FM point kinetics average
temperatures
TempDisFMPK=zeros(n,Length); %pre-allocated FM point kinetics
temperatures distributions
```

```
VQS=zeros(n,Length); %pre-allocated quasi-static fission distribution
VFMPK=zeros(n,Length); %pre-allocated FM point kinetics fission
distribution

EnergyQS=zeros(n,Length); %pre-allocated quasi-static energy
distribution
DeltaTQS=zeros(n,Length); %pre-allocated quasi-static temperature
change distribution
PowerQSAvg=zeros(1,Length); %pre-allocated quasi-static average powers
EnergyQSAvg=zeros(1,Length); %pre-allocated quasi-static average energy
DeltaTQSAvg=zeros(1,Length); %pre-allocated quasi-static average
temperature change

PowerFMPKAvg=zeros(1,Length); %pre-allocated FM point kinetics average
power
EnergyFMPK=zeros(n,Length); %pre-allocated FM point kinetics energy
distribution
DeltaTFMPK=zeros(n,Length); %pre-allocated FM point kinetics
temperature change distribution
EnergyFMPKAvg=zeros(1,Length); %pre-allocated FM point kinetics average
energy
DeltaTFMPKAvg=zeros(1,Length); %pre-allocated FM point kinetics average
temperature change

PowerQSMax=zeros(1,Length); %pre-allocated quasi-static maximum powers
PowerFMPKMax=zeros(1,Length); %pre-allocated FM point kinetics maximum
powers
den=1.67; %density in g/cm^3
Vf=141*12000; %volume in cm^3
Cp=0.72; %heat capacity in J/g*K

VFMPKAvg=zeros(n,Length); %pre-allocated FM point kinetics average
fission distribution
VPK=zeros(1,Length); %pre-allocated point kinetics fission distribution
Vpk=ones(n,1); %point kinetics eigenvector multiplication matrix (ones)
rhopk=zeros(1,Length); %pre-allocated point kinetics reactivity

zetaQS(:,1)=[betaT/lambda1G;1]; %quasi-static initial power
zetaFMPK(:,1)=[betaT/lambda1G; 1]; %FM point kinetics initial power

NumFM=100;
tic
for i=1:Length
fprintf('Processing %d of %d...',i,Length);
if (mod(i,NumFM)==1)
    [kQS, Pqs,
fmtxQS]=fmtxInterp(TdataOfficial,aUp,TemperatureQS); %eigenvalue,
eigenvector, and FM
end

if (mod(i,NumFM)==1)
    [kFMPK, Pfmpk,
fmtxPK]=fmtxInterp(TdataOfficial,aUp,TemperatureFMPK); %eigenvalue,
eigenvector, and FM
end
```

```
rhoQS=(kQS-1)/kQS; %quasi-static reactivity
rhoFMPK=(kFMPK-1)/kFMPK; %point kinetics reactivity
MQS=[-lambda1G betaT; lambda1G/gen (rhoQS-betaT)/gen];
MFMPK=[-lambda1G betaT; lambda1G/gen (rhoFMPK-betaT)/gen];

if i>1&&i<3
zetaQS(:,i)=zetaQS(:,i-1)+MQS*zetaQS(:,i-1)*DT; %quasi-static power
zetaFMPK(:,i)=zetaFMPK(:,i-1)+MFMPK*zetaFMPK(:,i-1)*DT; %FM Point
Kinetics Power

elseif i>=3
zetaQS(:,i)=zetaQS(:,i-1)+(3/2)*MQS*zetaQS(:,i-1)*DT-
(1/2)*MQS*zetaQS(:,i-2)*DT; %quasi-static power
zetaFMPK(:,i)=zetaFMPK(:,i-1)+(3/2)*MFMPK*zetaFMPK(:,i-1)*DT-
(1/2)*MFMPK*zetaFMPK(:,i-2)*DT; %FM Point Kinetics Power

else
zetaQS(:,i)=zetaQS(:,i); %quasi-static Power
zetaFMPK(:,i)=zetaFMPK(:,i); %FM Point Kinetics Power
end

PowerQS(:,i)=Pqs*zetaQS(2,i); %quasi-static Power distribution
EnergyQS(:,i)=PowerQS(:,i).*DT; %quasi-static energy distribution
DeltaTQS(:,i)=PowerQS(:,i).*DT/(den*Vf*Cp); %quasi-static Delta T
distribution

PowerQSAvg(i)=mean(PowerQS(:,i));
EnergyQSAvg(i)=PowerQSAvg(i).*DT; %quasi-static average energy
DeltaTQSAvg(i)=PowerQSAvg(i).*DT/(den*Vf*Cp); %quasi-static average
Delta T

PowerQSMax(i)=max(Pqs)*zetaQS(2,i); %quasi-static maximum power

PFMpkAvg=mean(Pfmpk); %average FM point kinetics eigenvector
VQS(:,i)=Pqs; %quasi-static eigenvectors
VQSAvg=mean(Pfmpk); %quasi-static average fission
VQSAvg=VQSAvg*Vpk;
VFMPK(:,i)=Pfmpk; %FM point kinetics eigenvector
VFMpkAvg=PFMpkAvg.*Vpk; %FM point kinetics uniform eigenvector
VFMPKAvg(:,i)=VFMpkAvg; %FM point kinetics uniform eigenvectors

PowerFMPK(:,i)=VQSAvg*zetaFMPK(2,i); %FM point kinetics power
distribution
EnergyFMPK(:,i)=PowerFMPK(:,i).*DT; %FM point kinetics energy
distribution
DeltaTFMPK(:,i)=PowerFMPK(:,i).*DT/(den*Vf*Cp); %FM point kinetics
Delta T distribution

PowerFMPKAvg(i)=mean(PowerFMPK(:,i));
EnergyFMPKAvg(i)=PowerFMPKAvg(i).*DT; %FM point kinetics average energy
DeltaTFMPKAvg(i)=PowerFMPKAvg(i).*DT/(den*Vf*Cp); %FM point kinetics
average Delta T
```

```
PowerFMPKMax(i)=max(Pfmpk)*zetaFMPK(2,i); %FM point kinetics maximum
power
TemperatureQS=TemperatureQS+EnergyQS(:,i)./(den*Vf*Cp); %quasi-static
temperature calculation

TemperatureFMPK=TemperatureFMPK+EnergyFMPK(:,i)./(den*Vf*Cp); %FM point
kinetics temperature
calculationTemperatureFMPKAvg=TemperatureFMPKAvg+mean(nonzeros(Pfmpk))*
zetaFMPK(2,i)./(den*Vf*Cp); %quasi-static temperature calculation

TempMaxQS(i)=max(TemperatureQS); %quasi-static maximum temperature
TempMaxFMPK(i)=max(TemperatureFMPK); %FM point kinetics maximum
temperature
TempAvgQS(i)=mean(TemperatureQS); %quasi-static average temperature
TempAvgFMPK(i)=mean(TemperatureFMPK); %FM point kinetics average
temperature
TempDisQS(:,i)=TemperatureQS; %quasi-static temperature distribution
TempDisFMPK(:,i)=TemperatureFMPK; %FM point kinetics temperature
distribution
rhoqs(i)=rhoQS; %quasi-static reactivity
rhofmpk(i)=rhoFMPK; %FM point kinetics reactivity
fprintf('Done.\n');
end
toc

TempAvgPK=zeros(1,Length); %pre-allocated point kinetics average
temperatures
zetaPK(:,1)=[betaT/lambda1G;1]; %point kinetics initial power
PowerPK=zeros(1,Length); %pre-allocated point kinetics power
EnergyPK=zeros(1,Length); %pre-allocated point kinetics energy
distribution
DeltaTPK=zeros(1,Length); %pre-allocated point kinetics temperature
change distribution
PowerPKMax=zeros(1,Length); %pre-allocated point kinetics maximum power
TemperaturePK=300; %point kinetics initial temperature
for i=1:Length
    e=interp1(TdataOfficial,keff,TemperaturePK); %point kinetics k_eff
    rhoPK=(e-1)/e; %point kinetics reactivity
    VPK(i)=mean(VQS(:,i)); %point kinetics eigenvector

    MF=[-lambda1G betaT; lambda1G/gen (rhoPK-betaT)/gen];

    if i>1&&i<3
        zetaPK(:,i)=zetaPK(:,i-1)+MF*zetaPK(:,i-1)*DT; %point kinetics
power
    elseif i>=3
        zetaPK(:,i)=zetaPK(:,i-1)+(3/2)*MF*zetaPK(:,i-1)*DT-
(1/2)*MF*zetaPK(:,i-2)*DT; %point kinetics power
    else
        zetaPK(:,i)=zetaPK(:,i); %point kinetics power
    end

TemperaturePK=TemperaturePK+VPK(i)*zetaPK(2,i)*DT/(den*Vf*Cp); %point
kinetics temperature calculation
    TempAvgPK(i)=TemperaturePK; %point kinetics average temperature
```

```
    rhopk(i)=rhoPK; %point kinetics reactivity
    PowerPK(i)=VPK(i)*zetaPK(2,i); %point kinetics power
    EnergyPK(i)=PowerPK(i)*DT; %point kinetics energy
    DeltaTPK(i)=PowerPK(i)*DT/(den*Vf*Cp); %point kinetics Delta T
    PowerPKMax(i)=max(VFMPK(:,i))*zetaPK(2,i); %point kinetics maximum
power
end


%power vs time plot
figure
plot(tL,zetaQS(2,:)*1E-6)
hold on
plot(tL,zetaFMPK(2,:)*1E-6)
hold on
plot(tL,zetaPK(2,:)*1E-6)
xlabel('time (s)')
ylabel('Power (MW)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')

%average power vs time plot
figure
plot(tL,PowerQSAvg*1E-6)
hold on
plot(tL,PowerFMPKAvg*1E-6)
hold on
plot(tL,PowerPK*1E-6)
xlabel('time (s)')
ylabel('Power (MW)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')

%Maximum power vs time plot
figure
plot(tL,PowerQSMax*1E-6)
hold on
plot(tL,PowerFMPKMax*1E-6)
hold on
plot(tL,PowerPKMax*1E-6)
xlabel('time (s)')
ylabel('Power (MW)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')


%average power vs time plot log scale
figure
semilogy(tL,PowerQSAvg*1E-6)
hold on
semilogy(tL,PowerFMPKAvg*1E-6)
hold on
semilogy(tL,PowerPK*1E-6)
xlabel('time (s)')
ylabel('log( Power (MW))')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')

% reactivity vs time plot
```

```
figure
plot(tL,rhoqs/betaT)
hold on
plot(tL,rhofmpk/betaT)
hold on
plot(tL,rhopk/betaT)
xlabel('time (s)')
ylabel('Reactivity ($)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')

%Reactivity vs Average temperature plot
figure
plot(TempAvgQS,rhoqs/betaT)
hold on
plot(TempAvgFMPK,rhofmpk/betaT)
hold on
plot(TempAvgPK,rhopk/betaT)
xlabel('Temperature (K)');ylabel('Reactivity ($)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')

%Average Temperature vs time plot
figure
plot(tL,TempAvgQS)
hold on
plot(tL,TempAvgFMPK)
hold on
plot(tL,TempAvgPK)
xlabel('time (s)');ylabel('Temperature (K)')
legend('Quasi-Static','FM Point Kinetics','Point Kinetics')


%% power distributions

IrealfirstQS=1; %initial time/ quasi-static average power
Irealfwhm1QS=find(zetaQS(2,:)>=max(zetaQS(2,:))/2,1,'first'); %time
when quasi-static average power reaches its first full width half
maximum
IrealmaxQS=find(zetaQS(2,:)==max(zetaQS(2,:))); %time when quasi-static
average power reaches its maximum
Irealfwhm2QS=find(zetaQS(2,:)>=max(zetaQS(2,:))/2,1,'last'); %time when
quasi-static average power reaches its second full width half maximum
IreallastQS=Length; %initial time/average power

%power distribution at initial average power
figure
PrealFirst=reshape(PowerQS(:,IrealfirstQS)*1e-6,x,y);
image(xlattice,ylattice,PrealFirst,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Power (MW)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%power distribution when average power reaches first full width half
maximum
```

```
figure
Prealfwhm1=reshape(PowerQS(:,Irealfwhm1QS)*1e-6,x,y);
image(xlattice,ylattice,Prealfwhm1,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Power (MW)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%power distribution when average power reaches maximum
figure
PrealMax=reshape(PowerQS(:,IrealmaxQS)*1e-6,x,y);
image(xlattice,ylattice,PrealMax,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Power (MW)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%power distribution when average power reaches second full width half
maximum
figure
Prealfwhm2=reshape(PowerQS(:,Irealfwhm2QS)*1e-6,x,y);
image(xlattice,ylattice,Prealfwhm2,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Power (MW)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%power distribution at final average power
figure
PrealLast=reshape(PowerQS(:,IreallastQS)*1e-6,x,y);
image(xlattice,ylattice,PrealLast,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Power (MW)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%% Temperature distributions
%temperature distribution at initial average power
figure
TrealFirst=reshape(TempDisQS(:,IrealfirstQS),x,y);
image(xlattice,ylattice,round(TrealFirst,2),'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Temperature (K)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%temperature distribution when average power reaches first full width
half maximum
figure
Trealfwhm1=reshape(TempDisQS(:,Irealfwhm1QS),x,y);
```

```
image(xlattice,ylattice,Trealfwhm1,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Temperature (K)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%temperature distribution when average power reaches maximum
figure
TrealMax=reshape(TempDisQS(:,IrealmaxQS),x,y);
image(xlattice,ylattice,TrealMax,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Temperature (K)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%temperature distribution when average power reaches second full width
half maximum
figure
Trealfwhm2=reshape(TempDisQS(:,Irealfwhm2QS),x,y);
image(xlattice,ylattice,Trealfwhm2,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Temperature (K)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%temperature distribution at final average power
figure
Treallast=reshape(TempDisQS(:,IreallastQS),x,y);
image(xlattice,ylattice,Treallast,'CDataMapping','scaled')
c=colorbar;
c.Label.String = 'Temperature (K)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')

%% error distributions

% error distribution at initial average power
figure
VFMerrorFirst=100*(VQS(:,IrealfirstQS)-
VFMPK(:,IrealfirstQS))./VFMPK(:,IrealfirstQS);
VFMerrorFirst=reshape(VFMerrorFirst,x,y);
VFMerrorFirst(isnan(VFMerrorFirst))=0; %all NaN=0
image(xlattice,ylattice,VFMerrorFirst,'CDataMapping','scaled')
c=colorbar('XTickLabel',{'-1.50','-1.00','-
0.50','0','0.50','1.00','1.50'});
c.Label.String = 'Fission Error (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
tix=get(c,'xtick')';
set(c,'xticklabel',num2str(tix,'%.2f'))
```

```
caxis([-1.82, 1.71]);

% error distribution when average power reaches first full width half
maximum
figure
VFMerrorFWHM1=100*(VQS(:,Irealfwhm1QS)-
VFMPK(:,Irealfwhm1QS))./VFMPK(:,Irealfwhm1QS);
VFMerrorFWHM1=reshape(VFMerrorFWHM1,x,y);
VFMerrorFWHM1(isnan(VFMerrorFWHM1))=0; %all NaN=0
image(xlattice,ylattice,VFMerrorFWHM1,'CDataMapping','scaled')
c=colorbar('XTickLabel',{'-1.50','-1.00','-
0.50','0','0.50','1.00','1.50'});
c.Label.String = 'Fission Error (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
tix=get(c,'xtick')';
set(c,'xticklabel',num2str(tix,'%.2f'))
caxis([-1.82, 1.71]);

% error distribution when average power reaches half maximum
figure
VFMerrorMax=100*(VQS(:,IrealmaxQS)-
VFMPK(:,IrealmaxQS))./VFMPK(:,IrealmaxQS);
VFMerrorMax=reshape(VFMerrorMax,x,y);
VFMerrorMax(isnan(VFMerrorMax))=0; %all NaN=0
image(xlattice,ylattice,VFMerrorMax,'CDataMapping','scaled')
c=colorbar('XTickLabel',{'-1.50','-1.00','-
0.50','0','0.50','1.00','1.50'});
c.Label.String = 'Fission Error (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
tix=get(c,'xtick')';
set(c,'xticklabel',num2str(tix,'%.2f'))
caxis([-1.82, 1.71]);

% error distribution when average power reaches second full width half
maximum
figure
VFMerrorFWHM2=100*(VQS(:,Irealfwhm2QS)-
VFMPK(:,Irealfwhm2QS))./VFMPK(:,Irealfwhm2QS);
VFMerrorFWHM2=reshape(VFMerrorFWHM2,x,y);
VFMerrorFWHM2(isnan(VFMerrorFWHM2))=0; %all NaN=0
image(xlattice,ylattice,VFMerrorFWHM2,'CDataMapping','scaled')
c=colorbar('XTickLabel',{'-1.50','-1.00','-
0.50','0','0.50','1.00','1.50'});
c.Label.String = 'Fission Error (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
tix=get(c,'xtick')';
set(c,'xticklabel',num2str(tix,'%.2f'))
caxis([-1.82, 1.71]);
```

```
% error distribution at final average power
figure
VFMerrorLast=100*(VQS(:,IreallastQS)-
VFMPK(:,IreallastQS))./VFMPK(:,IreallastQS);
VFMerrorLast=reshape(VFMerrorLast,x,y);
VFMerrorLast(isnan(VFMerrorLast))=0; %all NaN=0
image(xlattice,ylattice,VFMerrorLast,'CDataMapping','scaled')
c=colorbar('XTickLabel',{'-1.50','-1.00','-
0.50','0','0.50','1.00','1.50'});
c.Label.String = 'Fission Error (%)';
set(gca,'YDir','normal')
xlabel('x (cm)')
ylabel('y (cm)')
tix=get(c,'xtick')';
set(c,'xticklabel',num2str(tix,'%.2f'))
caxis([-1.82, 1.71]);

%% Differences in fissions

%fission difference distribution at initial average power
DFirst=1/(1-rhoqs(IrealfirstQS));
VFirst0=(VQS(:,IrealfirstQS)./sum(VQS(:,IrealfirstQS))).*DFirst;
DPKFirst=1/(1-rhofmpk(IrealfirstQS));
VFirstPK0=(VFMPK(:,IrealfirstQS)./sum(VFMPK(:,IrealfirstQS))).*DPKFirst
;
VFirst=nonzeros(VFirst0);
VFirstPK=nonzeros(VFirstPK0);
AFirst=((VFirst-VFirstPK)./VFirstPK).^2; %term used in RMS equation
RMS_VFirst=((1./length(VFirst0)).*sum(AFirst)).^(1/2); %root mean
square of relative difference between normalized and reference
eigenvectors
UFirst0=DETxy(:,12); %uncertainty matrix with zeros
UFirst=nonzeros(UFirst0);
UFirst2=UFirst.^2; %squared uncertainties
RMS_UFirst=((1./length(UFirst)).*sum(UFirst2)).^(1/2); %root mean
square of uncertainties
N=1:length(VFirst);
N=N(:);
figure
plot(N,VFirst)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
hold on
err=UFirst.*VFirstPK;
errorbar(N,VFirstPK,err)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')

%fission difference distribution when average power reaches first full
width half maximum
Dfwhm1=1/(1-rhoqs(Irealfwhm1QS));
Vfwhm10=(VQS(:,Irealfwhm1QS)./sum(VQS(:,Irealfwhm1QS))).*Dfwhm1;
DPKfwhm1=1/(1-rhofmpk(Irealfwhm1QS));
Vfwhm1PK0=(VFMPK(:,Irealfwhm1QS)./sum(VFMPK(:,Irealfwhm1QS))).*DPKfwhm1
;
Vfwhm1=nonzeros(Vfwhm10);
```

```
Vfwhm1PK=nonzeros(Vfwhm1PK0);
Afwhm1=((Vfwhm1-Vfwhm1PK)./Vfwhm1PK).^2; %term used in RMS equation
RMS_Vfwhm1=((1./length(Vfwhm10)).*sum(Afwhm1)).^(1/2); %root mean
square of relative difference between normalized and reference
eigenvectors
Ufwhm10=DETxy(:,12); %uncertainty matrix with zeros
Ufwhm1=nonzeros(Ufwhm10);
Ufwhm12=Ufwhm1.^2; %squared uncertainties
RMS_Ufwhm1=((1./length(Ufwhm1)).*sum(Ufwhm12)).^(1/2); %root mean
square of uncertainties
N=1:length(Vfwhm1);
N=N(:);
figure
plot(N,Vfwhm1)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
hold on
err=Ufwhm1.*Vfwhm1PK;
errorbar(N,Vfwhm1PK,err)
%title('Fission Source Distribution'))
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')

%fission difference distribution when average power reaches maximum
DMax=1/(1-rhoqs(IrealmaxQS));
VMax0=(VQS(:,IrealmaxQS)./sum(VQS(:,IrealmaxQS))).*DMax;
DPKMax=1/(1-rhofmpk(IrealmaxQS));
VMaxPK0=(VFMPK(:,IrealmaxQS)./sum(VFMPK(:,IrealmaxQS))).*DPKMax;
VMax=nonzeros(VMax0);
VMaxPK=nonzeros(VMaxPK0);
AMax=((VMax-VMaxPK)./VMaxPK).^2; %term used in RMS equation
RMS_VMax=((1./length(VMax0)).*sum(AMax)).^(1/2); %root mean square of
relative difference between normalized and reference eigenvectors
UMax0=DETxy(:,12); %uncertainty matrix with zeros
UMax=nonzeros(UMax0);
UMax2=UMax.^2; %squared uncertainties
RMS_UMax=((1./length(UMax)).*sum(UMax2)).^(1/2); %root mean square of
uncertainties
N=1:length(VMax);
N=N(:);
figure
plot(N,VMax)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
hold on
err=UMax.*VMaxPK;
errorbar(N,VMaxPK,err)
%title('Fission Source Distribution'))
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')

%fission difference distribution when average power reaches second full
width half maximum
Dfwhm2=1/(1-rhoqs(Irealfwhm2QS));
Vfwhm20=(VQS(:,Irealfwhm2QS)./sum(VQS(:,Irealfwhm2QS))).*Dfwhm2;
DPKfwhm2=1/(1-rhofmpk(Irealfwhm2QS));
```

```
Vfwhm2PK0=(VFMPK(:,Irealfwhm2QS)./sum(VFMPK(:,Irealfwhm2QS))).*DPKfwhm2
;
Vfwhm2=nonzeros(Vfwhm20);
Vfwhm2PK=nonzeros(Vfwhm2PK0);
Afwhm2=((Vfwhm2-Vfwhm2PK)./Vfwhm2PK).^2; %term used in RMS equation
RMS_Vfwhm2=((1./length(Vfwhm20)).*sum(Afwhm2)).^(1/2); %root mean
square of relative difference between normalized and reference
eigenvectors
Ufwhm20=DETxy(:,12); %uncertainty matrix with zeros
Ufwhm2=nonzeros(Ufwhm20);
Ufwhm22=Ufwhm2.^2; %squared uncertainties
RMS_Ufwhm2=((1./length(Ufwhm2)).*sum(Ufwhm22)).^(1/2); %root mean
square of uncertainties
N=1:length(Vfwhm2);
N=N(:);
figure
plot(N,Vfwhm2)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
hold on
err=Ufwhm2.*Vfwhm2PK;
errorbar(N,Vfwhm2PK,err)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')

%fission difference distribution at final average power
DLast=1/(1-rhoqs(IreallastQS));
VLast0=(VQS(:,IreallastQS)./sum(VQS(:,IreallastQS))).*DLast;
DPKLast=1/(1-rhofmpk(IreallastQS));
VLastPK0=(VFMPK(:,IreallastQS)./sum(VFMPK(:,IreallastQS))).*DPKLast;
VLast=nonzeros(VLast0);
VLastPK=nonzeros(VLastPK0);
ALast=((VLast-VLastPK)./VLastPK).^2; %term used in RMS equation
RMS_VLast=((1./length(VLast0)).*sum(ALast)).^(1/2); %root mean square
of relative difference between normalized and reference eigenvectors
ULast0=DETxy(:,12); %uncertainty matrix with zeros
ULast=nonzeros(ULast0);
ULast2=ULast.^2; %squared uncertainties
RMS_ULast=((1./length(ULast)).*sum(ULast2)).^(1/2); %root mean square
of uncertainties
N=1:length(VLast);
N=N(:);
figure
plot(N,VLast)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
hold on
err=ULast.*VLastPK;
errorbar(N,VLastPK,err)
xlabel('cell');ylabel('fission source distribution')
legend('Quasi-static','FM Point Kinetics')
```

Appendix C

Matlab Serpent Fission Matrix and Temperature Database

TREAT_ Gr

```
%% fission matrices for uniform temperature models

% 300 K
treatT1DT_fmtx0
fmtx1=fmtx_t;

% 400 K
treatT2DT_fmtx0
fmtx2=fmtx_t;

% 500 K
treatT3DT_fmtx0
fmtx3=fmtx_t;

% 600 K
treatT4DT_fmtx0
fmtx4=fmtx_t;

% 700 K
treatT5DT_fmtx0
fmtx5=fmtx_t;

% 800 K
treatT6DT_fmtx0
fmtx6=fmtx_t;

% 900 K
treatT7DT_fmtx0
fmtx7=fmtx_t;

% 1000 K
treatT8DT_fmtx0
fmtx8=fmtx_t;

% 1100 K
treatT9DT_fmtx0
fmtx9=fmtx_t;

% 1200 K
treatT10DT_fmtx0
fmtx10=fmtx_t;

%% Serpent Main Output Files for all Uniform Temperature Models
treatT1DT_res % 300 K
treatT2DT_res % 400 K
treatT3DT_res % 500 K
```

```
treatT4DT_res % 600 K
treatT5DT_res % 700 K
treatT6DT_res % 800 K
treatT7DT_res % 900 K
treatT8DT_res % 1000 K
treatT9DT_res % 1100 K
treatT10DT_res % 1200 K


%% Serpent Delayed Neutron fraction, decay constant, neutron generation
time, and effective multiplication factor

% 300 K
betaT1=ADJ_IFP_ANA_BETA_EFF(1,1); % delayed neutron fraction
Lambda1=ADJ_IFP_ANA_LAMBDA(1,1); % decay constant
ngt1=ADJ_IFP_GEN_TIME(1,1); % neutron generation time
keff1=IMP_KEFF(1,1); % effective multiplication factor

% 400 K
betaT2=ADJ_IFP_ANA_BETA_EFF(2,1); % delayed neutron fraction
Lambda2=ADJ_IFP_ANA_LAMBDA(2,1); % decay constant
ngt2=ADJ_IFP_GEN_TIME(2,1); % neutron generation time
keff2=IMP_KEFF(2,1); % effective multiplication factor

% 500 K
betaT3=ADJ_IFP_ANA_BETA_EFF(3,1); % delayed neutron fraction
Lambda3=ADJ_IFP_ANA_LAMBDA(3,1); % decay constant
ngt3=ADJ_IFP_GEN_TIME(3,1); % neutron generation time
keff3=IMP_KEFF(3,1); % effective multiplication factor

% 600 K
betaT4=ADJ_IFP_ANA_BETA_EFF(4,1); % delayed neutron fraction
Lambda4=ADJ_IFP_ANA_LAMBDA(4,1); % decay constant
ngt4=ADJ_IFP_GEN_TIME(4,1); % neutron generation time
keff4=IMP_KEFF(4,1); % effective multiplication factor

% 700 K
betaT5=ADJ_IFP_ANA_BETA_EFF(5,1); % delayed neutron fraction
Lambda5=ADJ_IFP_ANA_LAMBDA(5,1); % decay constant
ngt5=ADJ_IFP_GEN_TIME(5,1); % neutron generation time
keff5=IMP_KEFF(5,1); % effective multiplication factor

% 800 K
betaT6=ADJ_IFP_ANA_BETA_EFF(6,1); % delayed neutron fraction
Lambda6=ADJ_IFP_ANA_LAMBDA(6,1); % decay constant
ngt6=ADJ_IFP_GEN_TIME(6,1); % neutron generation time
keff6=IMP_KEFF(6,1); % effective multiplication factor

% 900 K
betaT7=ADJ_IFP_ANA_BETA_EFF(7,1); % delayed neutron fraction
Lambda7=ADJ_IFP_ANA_LAMBDA(7,1); % decay constant
ngt7=ADJ_IFP_GEN_TIME(7,1); % neutron generation time
keff7=IMP_KEFF(7,1); % effective multiplication factor
```

```
% 1000 K
betaT8=ADJ_IFP_ANA_BETA_EFF(8,1); % delayed neutron fraction
Lambda8=ADJ_IFP_ANA_LAMBDA(8,1); % decay constant
ngt8=ADJ_IFP_GEN_TIME(8,1); % neutron generation time
keff8=IMP_KEFF(8,1); % effective multiplication factor

% 1100 K
betaT9=ADJ_IFP_ANA_BETA_EFF(9,1); % delayed neutron fraction
Lambda9=ADJ_IFP_ANA_LAMBDA(9,1); % decay constant
ngt9=ADJ_IFP_GEN_TIME(9,1); % neutron generation time
keff9=IMP_KEFF(9,1); % effective multiplication factor

% 1200 K
betaT10=ADJ_IFP_ANA_BETA_EFF(10,1); % delayed neutron fraction
Lambda10=ADJ_IFP_ANA_LAMBDA(10,1); % decay constant
ngt10=ADJ_IFP_GEN_TIME(10,1); % neutron generation time
keff10=IMP_KEFF(10,1); % effective multiplication factor
```

Appendix D

Matlab Fission Matrix Combination Function File

fmtxInterp

```
function [k,fis,fmtx]=fmtxInterp(TData,fmtxDB,Tdist)
%INPUT
%TData - the set of uniform temperatures at which the fmtx DBs were
calculated
%fmtxDB - database of fission matrices, each evaluated at TData
%Tdist - 1D array of fuel temperatures for all cells

%OUTPUT
%fmtx - interpolated fission matrix based on Tdist
%k - eigenvalue of fmtx
%fis - normalized eigenvector of fmtx

eps=1e-10;
nCell=length(Tdist);
fmtx=zeros(nCell,nCell);

for i=1:nCell
    for j=1:nCell
        %note - this process is slow, may need a faster method
        fmtx(i,j)=interp1(TData(:), fmtxDB(:,i,j), Tdist(i));
    end
end

%get the principle eigenvalue/vector
[fis,k]=eigs(fmtx,1);

%normalize the eigenvector
fis=fis/sum(fis);

%set non-fueled regions to 0
fis(fis<eps)=0;

end
```