MODERN ROSSI ALPHA MEASUREMENTS

BY

GEORGE E. MCKENZIE IV

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Nuclear, Plasma, and Radiological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Master's Committee:

Professor Tomasz Kozlowski, Advisor
Professor Rizwan Uddin

ABSTRACT

The Rossi-α method determines the prompt neutron decay constant in a nuclear fissioning system at or near delayed critical. Knowledge of the prompt neutron decay constant is important for a critical system as it is a major contributor to the dynamic system behavior. The classical method for the Rossi experiment used gated circuitry to track the time when a neutron was incident upon the detector. The downside of this method is that the circuitry was complex and only one single fission chain could be measured at a time. The modern method allows many chains to be measured simultaneously by a pulse time tagging system such as the LANL custom designed List-mode module.

This thesis examines the implementation of the modern Rossi-α method on the all highly enriched uranium, HEU, Zeus experiment. Measurements are taken at several subcritical configurations, at critical in the presence of a source, and at one supercritical point. During the experiment, the List-mode module generates time tags of incoming neutron pulses. After the experiment, this list of neutron pulses is complied using custom software into a histogram. This histogram is fit using off the shelf graphing software to determine the value of α.

The subcritical measurements of α are used to extrapolate α at delayed critical. The extrapolation determined the value of α at delayed critical to be $\alpha = -89910\ s^{-1}$. This value is compared to the measured value of α at delayed critical which is determined to be $\alpha = -90408.4\ s^{-1}$. These values differ by 0.55% which is remarkably good agreement. This thesis also examines the expected value of α using a Monte Carlo transport code, MCNP. MCNP determined the value of α at delayed critical to be $-100048 \pm 0.584\ s^{-1}$. This result differs by 11.3% from the extrapolated value of α determined experimentally. When compared to systems with similar neutron spectra, the measured value of α fits well in comparison to historical measurements.

ACKNOWLEDGMENTS

First, I want to thank Dr. William "Bill" Myers of Los Alamos National Laboratory and Professor Tomasz Kozlowski for making this thesis possible. In the summer of 2012, Bill took me on as his student, and encouraged me to continue my education past the undergraduate level. Since then, Bill has been a wonderful mentor to me, both in respect to this thesis and other projects designed to expand my technical knowledge. In 2014, Professor Kozlowski took me under his wing, and since then I believe Professor Kozlowski has been everything I could have asked for in an academic advisor. Both Bill and Professor Kozlowski have alleviated many of the stresses that develop during execution of an experimental thesis.

I also owe Travis Grove, Jesson Hutchinson, Rene Sanchez, Bill Myers, Joetta Goda and Dave Hayes a huge thank you for putting up with my seemingly endless questions during the process of researching and implementing my experiment. Their patience and doodles on our white-boards not only made the completion of this thesis possible, but also enjoyable. I want to thank you all for providing an excellent working environment that inspires learning.

I want to thank Rene Sanchez and John Bounds for dedicating their time to my project. Rene and John were the main operators for the critical assemblies during experimentation. Without them, none of this would be possible. I am grateful to both of you for coaching me through set-up and allowing me to be an operator in training and perform the measurements.

I also owe Travis Grove and Rene Sanchez thanks for writing the scripts in Mathematica and MathCAD to test my Rossi-α program.

Finally, I want to thank Professor Roy Axford whose guidance connected me with Bill and deepened my interests in reactor physics. Professor Axford has also provided many excellent resources upon which to build my thesis.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

DAF	Device Assembly Facility
DC	Delayed Critical
GE	General Electric
HEU	Highly Enriched Uranium
LANL	Los Alamos National Laboratory
LLD	Lower Level Discriminator
MCNP	Monte Carlo Neutron Photon (Computational Software)
mil	One-thousandth of an Inch
NCERC	National Criticality Experiments Research Center
NNSS	Nevada National Security Site
SCA	Single Channel Analyzer
SNM	Special Nuclear Material
TTL	Transistor-Transistor Logic

LIST OF SYMBOLS

α	Prompt Neutron Decay Constant
β_{eff}	Delayed Neutron Fraction
ϵ	Efficiency of the Detection System
ν_p	Number of Neutrons Emitted Promptly
υ	Velocity of Thermal Neutrons
Σ_f	Macroscopic Fission Cross Section
A	Average Count Rate of a System
B	Maximum Correlated Count Probability
D_ν	Diven's Parameter
F	Average Fission Rate
k_{eff}	Multiplication Factor
k_p	Prompt Multiplication Factor
p(t)	Probability
l	Neutron Lifetime
Δt	Time Difference Between Neutron Counts

CHAPTER 1

REACTOR PHYSICS BACKGROUND

The Rossi-α method was developed by Bruno Rossi in the 1940s to establish the mass control increment between delayed and prompt critical [1]. This methodology uses the statistical fluctuation of the measured neutron population to determine the nuclear kinetic parameters associated with the physics of neutron chain reacting systems. Rossi's work is known as the Rossi experiment, and determined the fluctuations in neutron emission rates for single neutron chains [2].

1.1 Reactivity

Valuable insight into the dynamic system behavior of a critical system can be attained through measurement of the prompt neutron decay constant. Before understanding the theory related to the determination of the prompt neutron decay constant, some understanding of the related reactor physics quantities is necessary. The best place to start when discussing reactor physics is the concept of k_{eff} which is the multiplication factor. The multiplication factor is the ratio between the previous neutron generation and the current one. This factor is not a directly measurable quantity but may be the single most important piece of information about a fissioning system. The multiplication factor determines how close or far a neutron multiplying system is from being critical. When a system is critical, the value of k_{eff} is exactly one. Subcritical configurations have values of k_{eff} less than one, and supercritical configurations have values of k_{eff} greater than one as shown by Eq. 1.1.

$$k_{eff} \begin{cases} < 1 & subcrtical \\ = 1 & critical \\ > 1 & supercritical \end{cases} \tag{1.1}$$

In every nuclear reactor, a control system is used to maintain and adjust criticality. In commercial power reactors, neutron absorbing rods are used to control the neutron population in the core and therefore regulate the criticality. The critical assemblies at the National Criticality Experiments Research Center, NCERC, have a different method of controlling reactivity. The control rods are actually fissile material or reflectors themselves and the criticality of a system is increased as more special nuclear material, SNM, or reflector is added. In either case, a measure of the criticality or a related quantity of a system is necessary. A direct measurement of the k_{eff} of a system is quite difficult, so instead a measure of the reactor power is performed using neutron detectors. The power level is proportional to the change in the count rate shown by the detectors. Although an exact value of k_{eff} cannot be determined, the time behavior of the system is used to determine whether a system is subcritical, critical or supercritical.

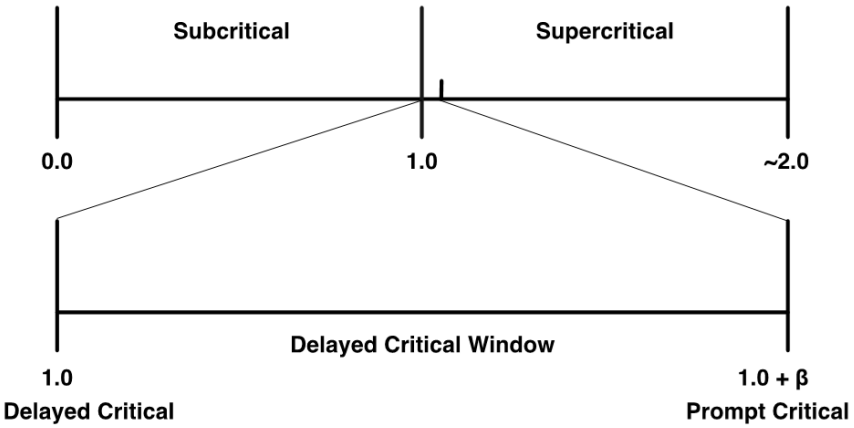

Figure 1.1: Criticality Range using k_{eff} as a guide [3].

Figure 1.1 is used by the NEN-2 group, Los Alamos National Laboratory's own Advanced Nuclear Technology Group, as a training tool when teaching about criticality. Nuclear fission is a statistically driven process. In Fig. 1.1, two critical values are shown. A system

is delayed critical when on average one neutron from each fission or its decay chain survives to produce another fission. The neutrons from the decay of fission products are imperative to sustaining a system at delayed critical. Similarly, a reactor is prompt critical when one neutron generated by the fission process survives to produce one fission. The region of criticality between delayed and prompt critical is ideal for operating power reactors because the reaction rate changes on timescales of seconds to hours. Without these long timescales, nuclear power would not be a reality.

Similar to k_{eff}, the prompt multiplication factor, k_p, is a measure of the state of a chain reacting system with respect to prompt critical. Just as a k_{eff} equal to one corresponded to delayed critical, a value of k_p equal to one corresponds to prompt critical. The relation between k_{eff} and k_p is shown in Eq. 1.2.

$$k_p \approx k_{eff} - \beta_{eff} \tag{1.2}$$

This approximation uses the quantity β_{eff} which is the delayed neutron fraction. The delayed neutron fraction is in itself an approximation based upon the probability for each fission daughter to produce a neutron as part of its decay process as well as the probability for that fission daughter to be produced during a fission event. Figure 1.1 shows an approximation where prompt critical is given as $1 + \beta_{eff}$. In reality, β_{eff} is the reactivity change between delayed critical and prompt critical and the value of k_{eff} at prompt critical is only approximated by $k_{eff} \approx 1 + \beta_{eff}$. The full derivation begins with the definition of ρ which is given by Eq. 1.3.

$$\rho = \frac{k_{eff} - 1}{k_{eff}} \tag{1.3}$$

Using a reactivity of β_{eff} and some algebraic manipulation the true value of k_{eff} at prompt critical is given by Eq. 1.4. The first two terms of the Taylor series of Eq. 1.4 make for an extremely good approximation of the value of k_{eff} at prompt critical.

$$k_{eff,\,prompt} = \frac{1}{1 - \beta_{eff}} \approx 1 + \beta_{eff} \tag{1.4}$$

The prompt neutron decay constant, α, depends on both the prompt multiplication factor,

k_p, and the neutron lifetime, l. Specifically, the prompt neutron lifetime is the average length of time a prompt neutron exists in a system before a terminating event. Termination can be caused by leakage from the system, non-fission capture, or fission capture.

The experiments for the Rossi-α method are all performed in the subcritical and delayed critical windows. Measurement of α between delayed and prompt critical is often difficult becasue the power level of the reactor is increasing which eventually saturates the detectors. Although the described Rossi experiment is not valid, measuring α above prompt critical is possible by measuring the prompt period of the reactor. Above prompt critical, α is defined as the inverse of the prompt period.

1.2 Prompt and Delayed Neutrons

Nuclear fission is an extremely useful but volatile process. When an atom fissions: daughter nuclei form, neutrons are liberated, photons are released, and massive amounts of energy are transferred to the surrounding media. Prompt neutrons are those released as a direct result of fission, and can be measured approximately 10^{-9} seconds after the start of the fission process. The daughter nuclides born in fission are highly unstable and will often produce more photons or even release neutrons themselves. Delayed neutrons are released on the order of milliseconds to seconds after the beginning of the fission process. Delayed neutrons are the main reason critical systems are safely controllable, but they provide little insight into the time-dependent behavior of the fission process. For this reason, the delayed neutrons in such systems are often ignored during Rossi-α measurements because they have little effect on the dynamic system behavior on such a short timescale. Insight into the time-dependent behavior of the fission process is attainable though measurement of prompt neutrons in a system during the implementation of the Rossi experiment. The Rossi experiment is able to determine the prompt neutron decay constant, α, of a system.

1.3 Inhour Relation and Prompt Neutron Decay Constant

The prompt neutron decay constant, often referred to as the Rossi-α, is a dynamic variable of a chain-reacting nuclear fission system. The Rossi-α models the time behavior of the prompt neutron population. This behavior is modeled well by an equation developed by Richard Feynman congruently with Rossi's development of the experiment. The prompt neutron decay constant can be used to create a dynamic model of a fissioning system for a single state of the system. The main uses for measurements of this type are in the operation and performance of reactors. Like the Inhour relation shown by Eq. 1.5, the Rossi-α method is another useful method of calibrating reactivity[1].

$$\omega \left[\frac{l}{\beta} + \sum_{i=1}^{G} \frac{\alpha_i}{\lambda_i + \omega} \right] = k_o \tag{1.5}$$

The Inhour relation uses a measured reactor period to determine a systems reactivity which becomes difficult to measure as the reactor period decreases in length as the reactor approaches prompt critical. The Inhour relation determines the value of ω which is related to the asymptotic period, $T_{as} = 1/\omega$. The value of ω is determined for a particular reactivity, k_o. The neutron lifetime l, and the delayed neutron fraction β are properties of a given system and are static in the calculation. The remaining constants α_i and λ_i are properties of each delayed neutron group. The constant α_i is not related to the Rossi-α, but rather is a constant $\alpha_i = \beta_i/\beta$. Where β_i is the fraction of delayed neutrons in group i and β is the total delayed neutron fraction. The Rossi-α method measures the correlation in neutron counts to determine the prompt neutron decay coefficient. If the value of α at delayed critical is well defined, further values of alpha can be related to their reactivity.

Rossi proposed that active fission systems are self-modulated; meaning that the emission rate of delayed neutrons is sufficiently slow that neutrons produced directly from two separated fission events are discernible. Measurements of this type are applicable near critical. Measurements below delayed critical are simple to perform as there is no positive period. Measurements above delayed critical are possible, but low power levels are necessary [4]. Measurements performed more than a few percent above delayed critical become difficult

because of the short reactor period [5]. The Rossi experiment is technically simple which is why it is widely used to find the prompt neutron decay constant of nuclear assemblies. This thesis will determine the prompt neutron decay constant of fissioning systems through application of the Rossi-α method.

CHAPTER 2

ROSSI-α METHOD

2.1 Accidental and Correlated Neutron Pairs

An understanding of the distinction between accidental and correlated neutron pairs is crucial to the comprehension of how the Rossi-α method was developed. Much like the distinction between prompt and delayed neutrons, dividing the detected neutrons into two groups is necessary to complete the analysis required when performing the Rossi-α method. Accidental and correlated pairs are the two groups. In a single fission chain, the accidental and correlated pairs relate to the prompt and delayed neutron groups. Correlated pairs refer to the prompt neutrons generated from a common fission ancestor. Accidental pairs are defined to be neutrons originating from a random source such as the background or delayed emission, but when multiple fission chains are being analyzed neutrons originating in a different fission chain are considered as accidental pairs. One of the assumptions made when using the Rossi-α method is that the measurement is being performed at zero power so there is no significant overlapping in fission chains.

Figure 2.1 provides a visual representation of accidental and correlated neutron pairs. In Fig. 2.1, X is the common fission ancestor to correlated pairs of neutrons like C, D, and G. Correlated counts are also seen in the other chains at A and E or B and F. Any combination of neutrons from separate chains denotes accidental pairs, such as A and B.

2.2 Development of Rossi-α Equation

The prompt neutron decay constant is found by fitting an equation heuristically developed by Richard Feynman to experimental data. The data is taken using an experiment proposed

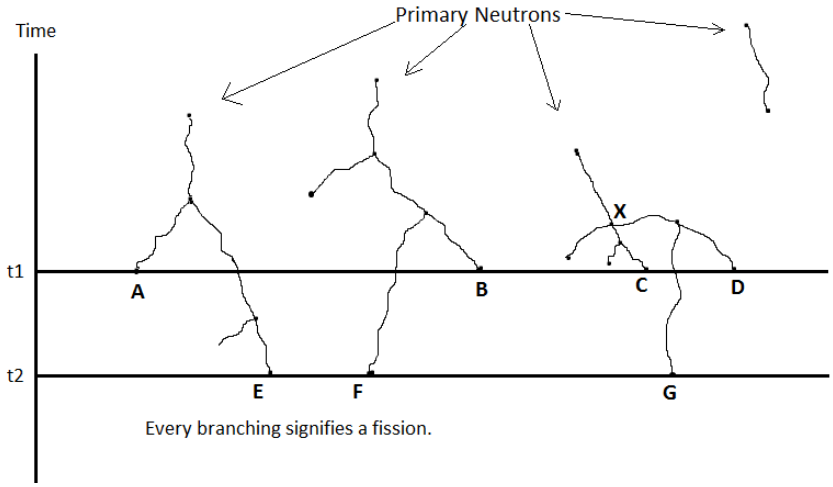

Figure 2.1: Visual representation of accidental and correlated neutron counts [6].

by Bruno Rossi. This experiment models the behavior of a single neutron chain in the experiment. Rossi's experiment works on the concept that subcritical fissile material is self-modulated [2]. This observation only holds for zero power systems. Systems operating above zero power have overlapping neutron chains making correlated counts much more difficult to distinguish. The zero power constraint is the limiting factor to the experiment above delayed critical. A few percent above delayed critical, the power level would be rapidly increasing and the system would have many overlapping fission chains [7].

In development of the Rossi-α fitting equation, first consider a fission occurring at some time t_0. The Rossi-α analysis considers the subsequent neutron counts incident on the detection system. The first neutron encountering the detector occurs at time t_1, this neutron will further be referred to as the initiating event. Then, consider the probability that another neutron will be incident on the detector in a given Δt after the initiating event. Assuming the first neutron is correlated to the fission at time t_0, the second count must either be a random neutron or a correlated neutron. $A\Delta t$ quantifies the probability that the second count is random, where A is the average count rate of a system and Δt is the time interval of the measurement [5].

The prompt neutron population must decay exponentially on average, so the probability of detecting a correlated event also decays exponentially with time. The prompt neutron decay constant measures the speed of this exponential decline; the behavior is modeled by

$e^{\alpha t}$. The exponential including α has been shown here with a positive sign. The prompt neutron decay constant has been modeled using many different sign conventions. The sign conventions used in this paper follow the sign conventions used by Orndoff [5]. Orndoff's convention defines α to be negative when below prompt critical. The probability of the count detected being a prompt neutron can then be written as $Be^{\alpha t}\Delta t$ [5].

The probability that a neutron is counted at time $t_0 = 0$ can be generalized by Eq. 2.1 to be equal to the average fission rate, F.

$$p_0(t_0)\Delta_0 = F\Delta_0 \tag{2.1}$$

Where in general p_x is the probability of detecting a neutron count number x at a time t_x. The time t_x exists within the time window Δ_x.

Now, the probability of another neutron due to fission being counted at some t_1 after the initial count, which occurred at t_0, is of interest. The probability of this second count being detected can be quantified by Eq. 2.2.

$$p_1(t_1)\Delta_1 = \epsilon\nu_p\upsilon\Sigma_f e^{\alpha(t_1-t_0)}\Delta_1 \tag{2.2}$$

Where ϵ is the efficiency of the detector in counts per fission, ν_p is the number of prompt neutrons emitted at time t_1, υ is the velocity of thermal neutrons, Σ_f is the macroscopic fission cross section, and when υ and Σ_f are combined they become the average fission rate per unit neutron density $\upsilon\Sigma_f$ [4].

Next, the probability of a neutron count occurring at time t_2 after counts occurred at both t_0 and at t_1 and from the same fission chain is of interest. The probability is quantified in Eq. 2.3.

$$p_2(t_2)\Delta_2 = \epsilon(\nu_p - 1)\upsilon\Sigma_f e^{\alpha(t_2-t_0)}\Delta_2 \tag{2.3}$$

Notice that the ν term has been modified to $(\nu - 1)$ to account for the neutron lost at t_1 to the fission chain [4].

All three of the probabilities calculated in Eq. 2.1, Eq. 2.2, and Eq. 2.3 are independent and can be combined to give the probability of occurrence of two chain-related counts

initiated by a fission at time t_0; the first subsequent count occurring at time t_1 in Δ_1 and the second happening at some time t_2 in Δ_2 [4]. The probability of the above-mentioned sequence coming to fruition can be found by integrating the product of the probabilities for events at t_1 and t_2 over all time up until t_1. This integration is shown in Eq. 2.4.

$$p_c(t_1, t_2)\Delta_1\Delta_2 = \int_{-\infty}^{t_1} p(t_1)\Delta_1 p(t_2)\Delta_2 F \, dt_0 \tag{2.4}$$

This integration is performed becasue there is no way to know that a detected count is caused directly from the fission. Instead, it is assumed that detected counts relate to the counts at time t_1 and t_2.

With a little simplification, Eq. 2.4 can be simplified into Eq. 2.5 which portrays the probability of two chain related events occurring as a result of a fission at time t_0.

$$p_c(t_1, t_2)\Delta_1\Delta_2 = F\epsilon^2 \frac{D_\nu k_p^2}{2(1-k_p)l} e^{\alpha(t_2-t_1)}\Delta_1\Delta_2 \tag{2.5}$$

Equation 2.5 is simplified from Eq. 2.4 using $\overline{\nu_p(\nu_p-1)}$ as an average of the number of prompt neutrons emitted and the identities shown in Eq. 2.6 and Eq. 2.7.

$$\overline{\nu_p} = \frac{k_p\Sigma_a}{\Sigma_f} = \frac{k_p}{\Sigma_f \upsilon l} \tag{2.6}$$

$$D_\nu = \frac{\overline{\nu_p(\nu_p-1)}}{\overline{\nu_p^2}} \tag{2.7}$$

These identities refer to the definitions of the average emission of prompt neutrons and Diven's parameter respectively [4].

The probability that the counts seen at time t_1 and t_2 are an accidental pair is the same as the product of the average fission rate and the efficiency of the detector in the time bin. This probability can be seen in Eq. 2.8 [4].

$$p_r(t_1, t_2)\Delta_1\Delta_2 = F^2\epsilon^2\Delta_1\Delta_2 \tag{2.8}$$

The total probability for observing a pair of counts in Δ_1 and Δ_2 is the aggregate of the

probabilities found above as shown in Eq. 2.9 [4].

$$p(t_1,t_2)\Delta_1\Delta_2 = F^2\epsilon^2\Delta_1\Delta_2 + F\epsilon^2\frac{D_\nu k_p^2}{2(1-k_p)l}e^{\alpha(t_2-t_1)}\Delta_1\Delta_2 \tag{2.9}$$

The Rossi experiment guarantees an interaction in the time interval Δ_1 because this is the initiating event. With some manipulation, the probability of the first count occurring in the time interval Δ_1, $F\epsilon\Delta_1$, can be separated and set to 1 as shown by Eq. 2.10 [4].

$$p(t_1,t_2)\Delta_1\Delta_2 = F\epsilon\Delta_1\left[F\epsilon\Delta_2 + \epsilon\frac{D_\nu k_p^2}{2(1-k_p)l}e^{\alpha(t_2-t_1)}\Delta_2\right] \tag{2.10}$$

The result of the generalization of this process to any time after $t_1 = 0$ can be seen in Eq. 2.11 [4].

$$p(t)\Delta = F\epsilon\Delta + \epsilon\frac{D_\nu k_p^2}{2(1-k_p)l}e^{\alpha t}\Delta \tag{2.11}$$

In Orndoff's paper [8], a correction is made to Eq. 2.11 by the consideration of the effect of detection of the fission producing the count at $t = 0$. Consider δ to be the effective number of neutrons resulting from this fission and detection process, at $t = 0$. Since detection may involve capture, scattering, or fission, δ will depend on the type and placement of the detector and must be evaluated for a particular experimental setup [8]. The correction to Eq. 2.11 modifies the $\overline{\nu_p(\nu_p-1)}$ term hidden as a part of Diven's parameter. The correction term is shown in Eq. 2.12.

$$\overline{\nu_p(\nu_p-1)} + \frac{2\nu_p(1-k_p)}{k_p}\delta \tag{2.12}$$

With the correction, the probability Eq. 2.11 becomes Eq. 2.13 [4].

$$p(t)\Delta = F\epsilon\Delta + \epsilon\frac{\epsilon[\overline{\nu_p(\nu_p-1)} + 2\nu_p(1-k_p)\delta/k_p]k_p^2}{2\overline{\nu}p^2(1-k_p)l}e^{\alpha t}\Delta \tag{2.13}$$

The correction added by δ is at most a few percent, and Orndoff suggests δ need not be evaluated precisely [8]. Uhrig suggests in Random Noise Techniques [4] that the correction itself is often neglected because of its small magnitude. Often for simplicity the total prob-

ability to detect a neutron event in some Δ_2 after detecting an event at Δ_1 is written in the general form shown in Eq. 2.14.

$$P(t) = A + Be^{\alpha t} \tag{2.14}$$

Equation 2.14 is fit to experimental data during analysis.

Using Uhrig's suggestion to neglect the δ correction, the parameters A and B are represented by Eq. 2.15 and 2.16 [4].

$$A = F\epsilon \tag{2.15}$$

$$B = \frac{\epsilon D_\nu k_p^2}{2\alpha l^2} \tag{2.16}$$

2.3 Measurement

The Rossi experiment is a technically simple measurement used to constitute a reactivity calibration without the use of the Inhour equation [1]. The measurement is especially important in reactors where the neutron lifetime is extremely short. The Rossi experiment is performed by placing a sufficiently sensitive neutron detector near a neutron multiplying, chain reacting system [2]. Although not always possible, the ideal placement of the detector is near the center of the system. The central placement alleviates any issue with room return or neutron back scatter from the floor, ceiling, or walls into the neutron detectors. Further enhancement of this experimental set-up can be realized by adding multiple detectors running on independent channels to minimize dead time.

The prompt neutron decay constant can then be calculated from a fit of the data measured by the detector system. The system collects neutron detection events and records the time that each event occurs. The measurement of the time in which a count occurred is the main improvement between the classical and modern techniques for the Rossi experiment. These differences will be discussed later in this thesis.

The prompt neutron decay constant best measures the dynamic behavior of a system near criticality through a measurement of neutron pulse correlation. The value of the prompt

neutron decay constant, as previously stated, relies on the prompt multiplication factor, k_p, and the prompt neutron lifetime, l, as shown by Eq. 2.17.

$$\alpha = \frac{k_p - 1}{l} \tag{2.17}$$

By definition, the prompt neutron decay constant is zero when the system is prompt critical, negative below prompt critical, and positive above prompt critical [5]. The value of α at delayed critical can be estimated by plotting the subcritical values of α vs. the inverse of the count rate and generating a linear fit. The y-intercept is the value of α at delayed critical because at delayed critical the count rate approaches infinity, so the inverse of the count rate approaches zero. This allows interpolation in the region where power is increasing too rapidly to perform the Rossi experiment. This analysis allows for interpolation between the data taken and the defined value of α at prompt critical. The data could then be extrapolated beyond prompt critical in a similar fashion. The prompt neutron decay constant is defined by Eq. 2.17, but can also be defined to be the inverse of the prompt period. For systems that can be measured in a prompt critical state, the value of α can be measured in this way.

Another notable point is that at delayed critical α can be approximated by Eq. 2.18.

$$\alpha_{DC} = \frac{-\beta_{eff}}{l} \tag{2.18}$$

The result comes from Eq. 1.2 that $k_p = 1 - \beta_{eff}$ when $k_{eff} = 1$. Using this approximation and Eq. 2.17 the result shown in 2.18 is attained. In this analysis, β_{eff} denotes the effective delayed neutron fraction of the system.

2.4 Data Analysis

Binning Data

The first step in data analysis is binning. The data will eventually be binned into a histogram based on the time difference, Δt, between an initiating event and a subsequent count. In the

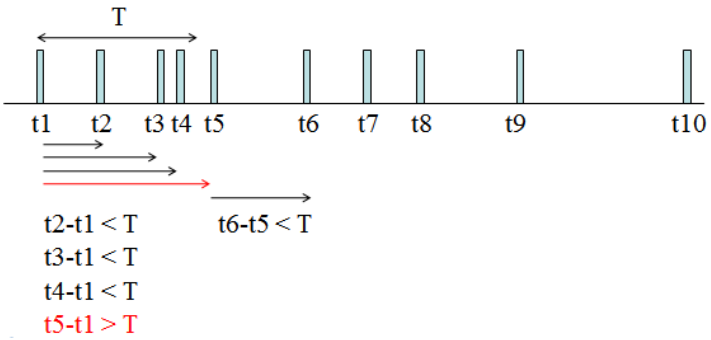

Figure 2.2: The binning process used to generate the histogram used in Rossi-α analysis.

classical approach to the Rossi experiment, bin size was predetermined, and all of the binning was done by the analog circuitry. The modern approach allows binning to be performed post-measurement removing the need for re-measurement to modify the bin size. The modern approach allows for more sophisticated analysis of the data because different size time binning can be performed on a single set of data to find the ideal bin size.

The binning process is performed using the first computer program included in Appendix A. The program inputs are the time window which is the total time for all bins, and the length of each bin. First, the program sets the initiating event to the first count chronologically. Then, it finds the final time in the file. The program uses the final time to make sure the time window for binning is smaller than the length of measurement. Next, the current event is set to be the count occurring directly after the initiating event. In relation to the theoretical derivation of the Rossi-α equation, the initiating event is the event occurring at t_1, and the current event is the event occurring at t_2. The time difference between the initiating event and the current event is calculated. So long as the calculated time difference is less than previously set time window, the count is binned. The current event moves to the next chronological count, and the time difference is measured again. If the time difference is greater than the time window, the initiating event is moved to the next chronological event, and the current event is set to the event directly after the initiating event. The binning process is well described by Fig. 2.2 and the flowchart shown in Fig. 2.3. In the figure, T is the time window and $t1$ is the initiating event. Next, the algorithm moves the initiating event to the second event or t_2 in Fig. 2.2.

Testing of this algorithm was performed using specially modified data inputs, and an

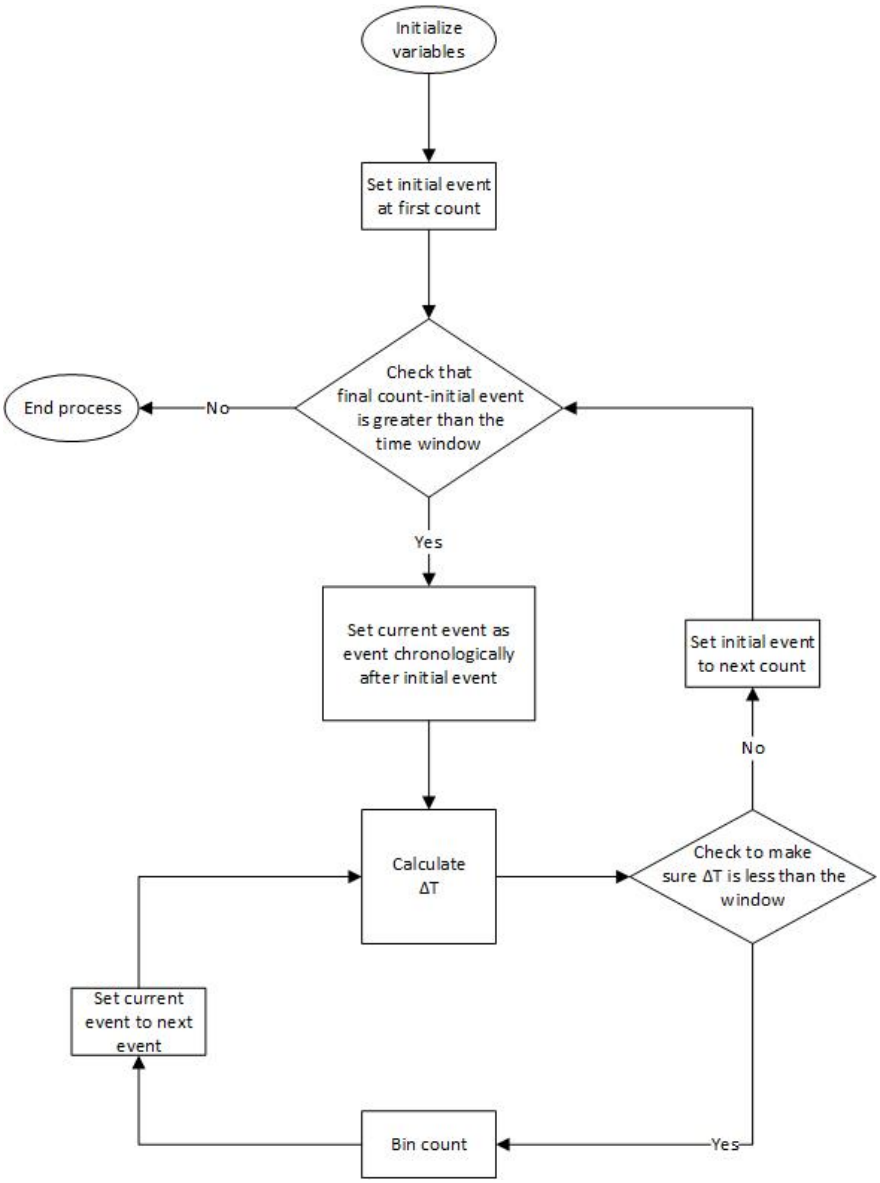

Figure 2.3: Flow chart of the binning code.

independently written Mathematica code for comparison.

Fitting Data

Once the data has been recorded and compiled appropriately into a histogram, equation 2.14 is fit to the histogram using a nonlinear least squares fit [9] to find the constants A, B, and α. The fit was performed using a plotting software from Origin Labs. The histogram created has inherent dead time related to the detector and counting circuit. As shown in Fig. 2.4, the first few channels under-report the number of counts. If unmodified, the value of Rossi-α would be incorrectly determined by the fit. To correct the dead time issue, the first few channels are removed by the user. At most, the first five channels are removed. The number of channels removed varies based on the width of the bins, and the hardness of the neutron spectra (average speed of the neutron population). An example of the behavior that is removed is shown in Fig. 2.4; for this example the first five channels are removed. The constants resolved from fitting, A, B, and α, can be further manipulated to generate additional information about the system. The fit will give values for A, B and α. A and B represent a collection of five independent system parameters as demonstrated in Eq. 2.15 and 2.16. The five independent system parameters include: F, ϵ, D_ν, k_p, and l. So, A and B can be used to find one parameter if the other three parameters are already known or independently measured [4]. Although finding these additional parameters is beyond the scope of this thesis, determination of these reactor physics parameters may be beneficial extension of this work.

Interpretation of Results

Each fit produces one value of Ross-α. A typical Rossi-α experiment measures data at many different configurations (subcritical, critical, and supercritical). For this thesis, the different configurations of the Zeus experiment are measured. The Zeus experiment uses mass separation of the fuel to produce different configurations. Using the values of α determined from multiple subcritical measurements, the data points are plotted on a linear scale. The

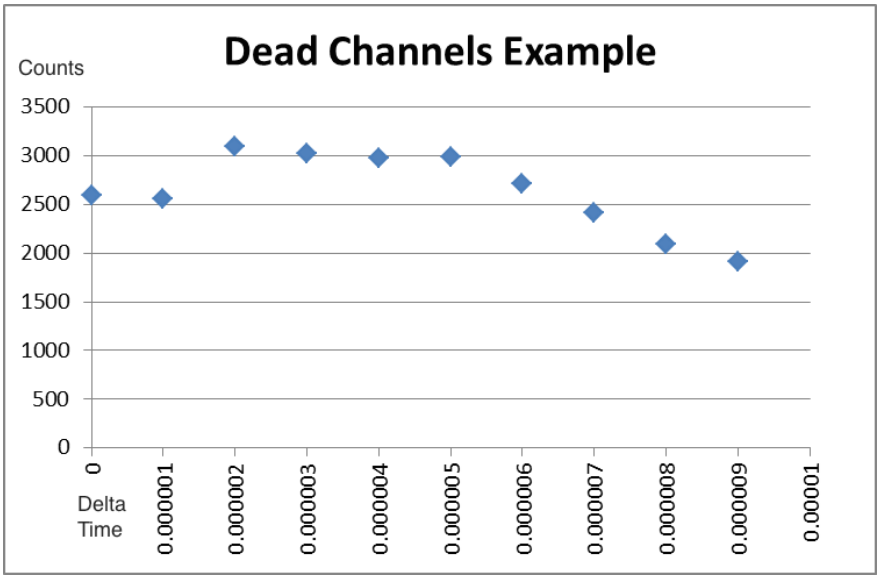

Figure 2.4: An example of the dead time behavior by showing the first ten channels in the histogram.

data forms a straight line and can be used to predict the locations of delayed and prompt critical. Plots of α vs. inverse count rate and α vs. reactivity, ρ, are created. These plots predict the value of α at delayed critical. When the reactivity is zero or the inverse of the count rate approaches zero, the system is approaching delayed critical. A line is fit to the data for α vs. inverse count rate. This y-intercept of this line is the value of α at delayed critical. Similarly, a line is fit the graph of α vs. ρ. this line is used to calibrate the reactivity of the system based on the value of α. are linearly fit the value of α at delayed critical is the y-intercept of that fit.

The reactivity of a system can be calibrated based on the value of α at prompt critical, and the value of α at delayed critical. Using similar triangles such as the one shown in Fig. 2.5 the reactivity of each measurement can be determined using Eq. 2.19; where X is the reactivity of the measured point in dollars.

$$\frac{\alpha_{DC}}{1\$} = \frac{\alpha_{measured}}{1\$ + X} \tag{2.19}$$

Equation 2.20 is an algebraically manipulated form of Eq. 2.19 to make analysis easier.

$$X = \frac{\alpha_{measured} - \alpha_{DC}}{\alpha_{DC}} \tag{2.20}$$

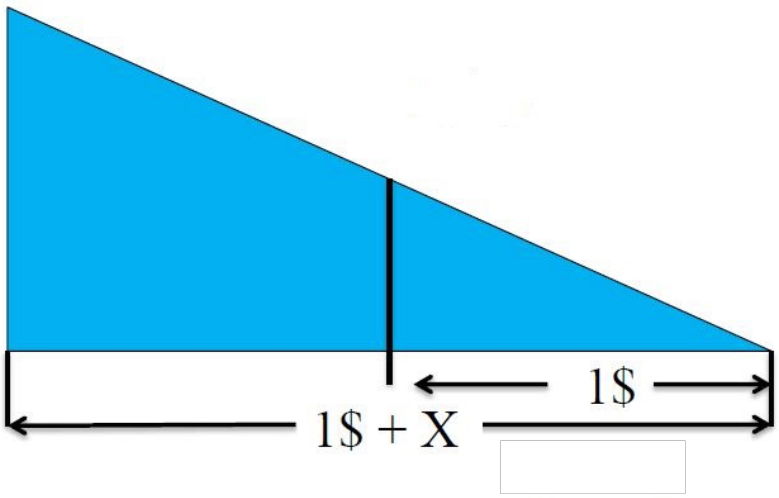

Figure 2.5: Similar triangle used to calibrate the reactivity of a system using the value of α determined at delayed critical [10].

2.5 Classical Measurements

The Rossi-α method has been used many times to determine the prompt neutron decay constant of a nuclear fissioning system. The most notable implementation of the Rossi experiment was a system designed by John Orndoff. Orndoff's system consisted of a ten channel time analyzer which used gating hardware to bin counts during experimentation. The bin size on the system were determined prior to experimentation, and bins were restricted to be either 0.25 or 0.5 μs wide [1].

A block diagram of Orndoff's apparatus is shown in Fig. 2.6. The system was attached to a U^{235} fission chamber placed inside the assembly of interest. The detector was attached to two separate circuits. The first of these circuits collected a long, square gating pulse which fed through a series of 0.25 μs delay lines successively opening all ten channels [1]. The second circuit collected very narrow pulses which are counted in one of the ten channels.

The gating worked such that after the gating pulse each counter would be opened for a preset length of time, either 0.25 or 0.5 μs, to collect counts and then trigger the next counter [1].

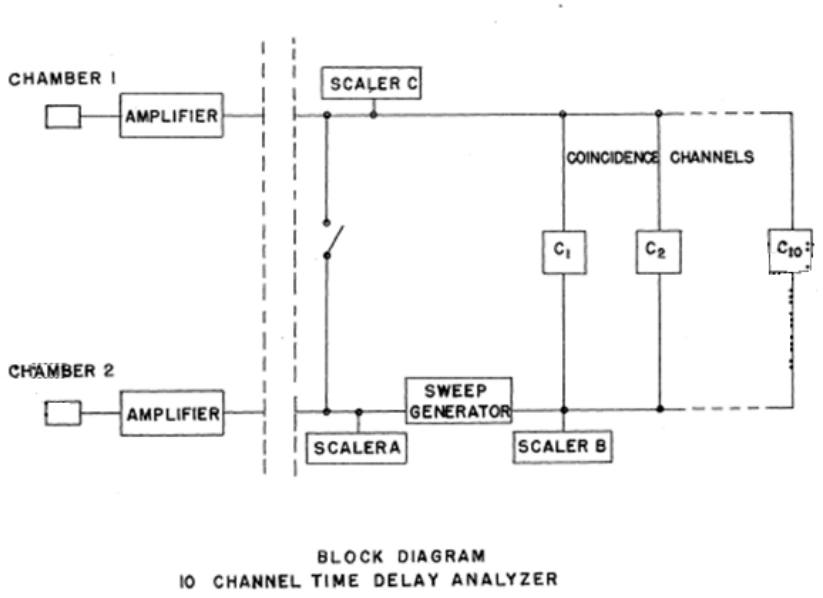

Figure 2.6: Block diagram of Orndoff's coincidence counting circuit [5].

Orndoff's time analyzer circuit did all the binning, but made customized measurements more complicated. Becasue the data was binned during experimentation, each measurement was only useful for those system parameters. Any changes would require repetition of data acquisition. This method is discussed in detail in a Los Alamos report written by Orndoff [5].

2.6 Modern Measurements

The modern measurement of the prompt neutron constant using the Rossi-α method no longer utilizes complex multi-circuit designs. Instead neutrons are time tagged by a custom designed piece hardware called a List-mode. For this experiment, a custom LANL designed List-mode module is used. The measurement using the List-mode is incredibly simple; all that is needed is a basic pulse shaping circuit.

A block diagram of the circuit used during these measurements is provided in Fig. 2.7.

The pulse shaping circuit includes: a preamplifier, amplifier, high voltage source, and single channel analyzer. In the modern method, the detector counts incident neutron pulses. These pulses undergo pulse shaping and are sent to the List-mode module which time tags the pulse. The List-mode is controlled by a computer, and the software is extremely user friendly.

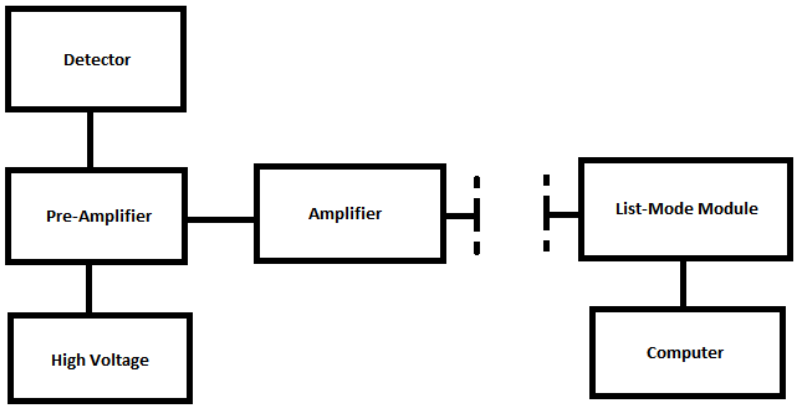

Figure 2.7: List-mode data acquisition circuit.

Unlike the classical method, the modern experiment requires extra analytical steps to bin the data, but the modern method's digitization of part of the data acquisition system, through use of the List-mode, excels in its simplicity and ability to analyze a single data set in many different ways.

CHAPTER 3

ROSSI-α EXPERIMENT

3.1 Equipment

Critical Assemblies

Rossi-α measurements are performed on the NCERC critical experiments at the Device Assembly Facility, DAF, in the Nevada National Security Site, NNSS. Details of each assembly are outlined in this section.

Comet

Comet is a general purpose, vertical lift critical assembly. The Comet assembly is a large platform capable of hosting ten tons. The assembly also includes a hydraulic lift capable of finely adjusting one ton of material to one mil (one thousandth of an inch). The comet assembly controls criticality through mass separation between the top and bottom fuel which is why the accuracy of the hydraulic lift is important. The assembly itself is very versatile because any combination of fuel and reflectors can be combined to fit the needs of the experiment. The particular configuration used in this thesis is the Zeus experiment reflector and the Jemima plates.

The Zeus experiment refers to the massive copper reflector placed on the top of the machine. The fuel used is comprised of about 100 kg of HEU in a short fat cylinder. The fuel consists of 93% enriched uranium in thin 0.125" thick plates nicknamed the Jemima plates because of their similarity to pancakes. The fuel has an outer diameter of about 21" and has an inner annulus of about 2.5" in the bottom half of the fuel as shown by the green

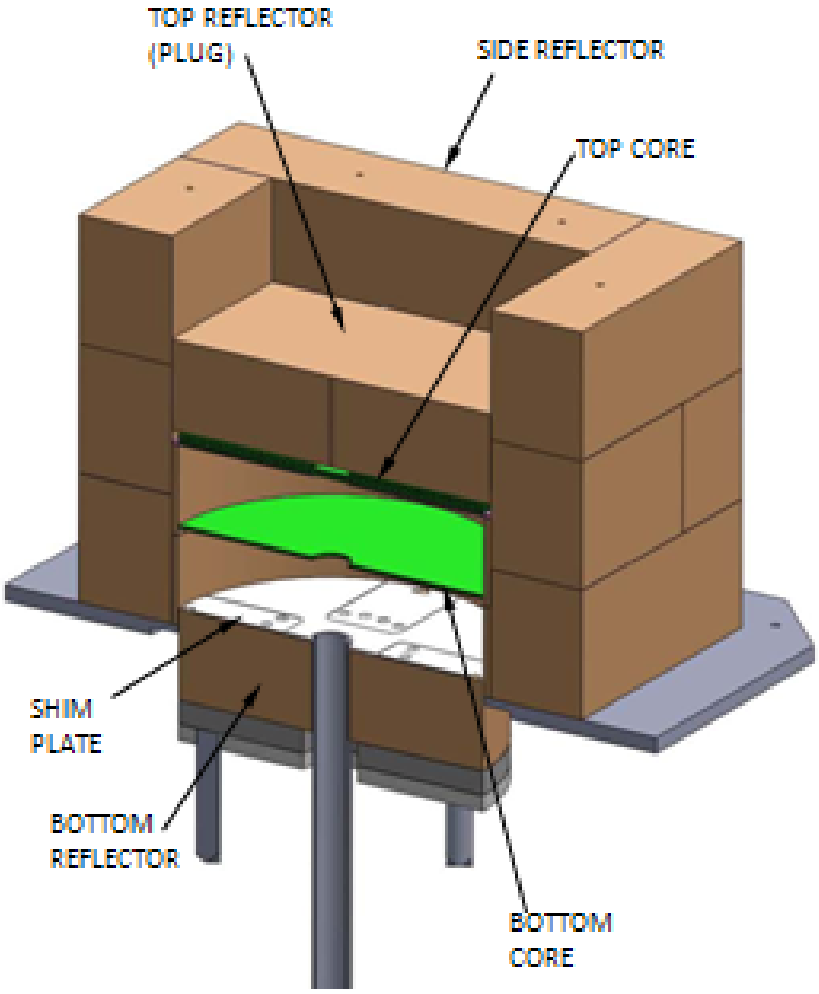

Figure 3.1: Cut away of the Zeus configuration to show interior details.

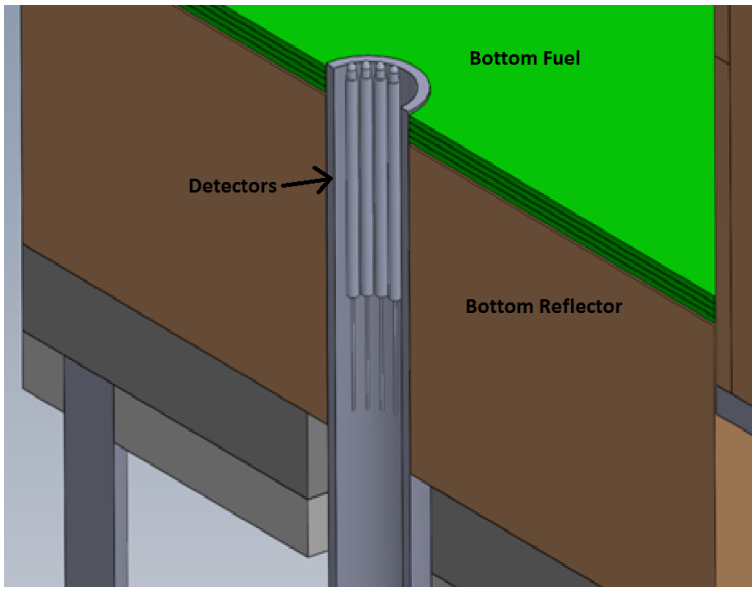

Figure 3.2: The detector positioning when measuring α in the Zeus configuration on Comet.

in Fig. 3.1 or in the hands of the scientists in Fig. 3.5. The inner annulus provides space for an aluminum cylinder which provides structural support to the fuel and keeps it from moving. For the Rossi experiment this inner cylinder also provides the ideal place to put the detectors as shown in Fig. 3.2. The upper half of the fuel has a 21" outer diameter and has no inner annulus. An extremely thin stainless steel diaphragm holds the upper half of the fuel inside the copper reflector. The diaphragm is shown in Fig. 3.3 and is located at the bottom of the top half of the core in Fig. 3.1. While the bottom half of the fuel sits on top of a large hydraulic cylinder made of copper. Although Fig. 3.1 shows the bottom of the core floating inside the reflector, in reality this fuel sits directly on top of the bottom reflector. This hydraulic cylinder controls the mass separation of the fuel helping produce different configurations.

The Zeus experiment also includes an enormous copper reflector weighing about 1 ton visualized in Fig. 3.3, 3.4, and 3.5. Figure 3.3 is a picture looking down at the diaphragm from above. Figure 3.4 is a picture of the top of the reflector if the diaphragm were removed and figure 3.5 is a picture of the whole assembly. The copper reflector moderates the fast neutrons produced in fission into an intermediate neutron energy spectrum. More information on the Zeus experiment can be found in HEU-MET-FAST-073 in the International

Figure 3.3: Zeus experiment fuel cavity looking down at the diaphragm from the top.

Figure 3.4: The top of the interior of the Zeus experiment reflector looking up if there was no diaphragm.

Figure 3.5: The Zeus experiment.

Handbook of Evaluated Criticality Safety Benchmark Experiments [11].

He^3 Detectors

Much like Geiger counters, He^3 neutron detectors are excellent proportional counters for neutrons. Like Geiger counters, He^3 detectors can not detect the energy of an incident neutron pulse. The Rossi experiment is mainly interested in the time a neutron count occurred, not the amount of energy deposited by the neutron into the detector. The detectors used in this experiment are 40 atm Reuter Stokes detectors from GE. These detectors are 5" long with an active length of 3", and have a diameter of 0.25" shown in Fig. 3.6. The detector is good at time of flight measurements and has short dead time. The quick recovery speed is ideal for the Rossi experiment on fast critical assemblies where correlated counts can be extremely collocated in time. The small dimensions of the detector are also beneficial because the best results are generated when the detector is placed near the center of the assembly, and space inside these assemblies is extremely limited. To further reduce the dead time during measurements, multiple detectors on independent channels are used during data collection.

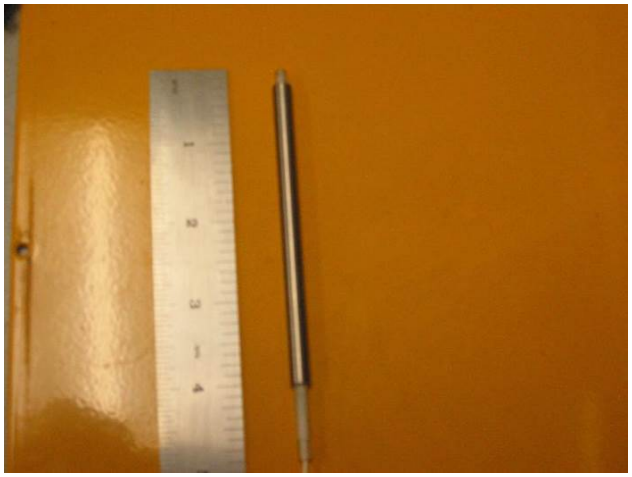

Figure 3.6: The He^3 detectors used to measure Rossi-α.

List-mode Module

The combined advances in computer processor miniaturization, data processing speeds, data recording speeds, and data storage capacity have created new ways of data acquisition [12]. For instance, the LANL List-mode module is capable of recording the incident time of a transistor-transistor logic pulse, TTL. The List-mode is able to distinguish between pulses that are more than 100 nanoseconds apart. When recording data, the List-mode records the time a pulse was encountered in seconds as well as the channel a count occurred in. The current List-mode design is able to accommodate 32 channels. This type of data acquisition is extremely versatile because the same set of output data can be analyzed using multiple techniques [12].

3.2 Procedure

The results presented below are acquired from the Comet assembly configured with the Zeus experiment. Each channel of the experimental set-up includes a 40 atm Reuter Stokes He^3 detector placed inside the spindle (center of fuel) of the assembly. Each detector is connected to an Ortec preamplifier. The preamplifier is coupled to an amplifier and high voltage source set to +2100 V and the amplifier is adjusted such that the neutron peak is centered around 6 V. The amplifier is connected to a single channel analyzer, SCA. The SCA's lower level

discriminator, LLD, is adjusted to 4 V so that the gamma noise is removed without disturbing the neutron peak. The output of the SCA is routed to the data acquisition lines which are approximately one quarter mile long. The data acquisition lines are composed of large gage coaxial cables. The other end of these data acquisition cables is in the control room where the remote operations are performed. The data feed from these wires is connected to a 50 Ω terminator as well as to the List-mode module. (The 50 Ω terminators were added after the series of experiments discussed in this thesis to reduce the effect of noise.) The List-mode module records the data in conjunction with a data acquisition computer.

The detectors are then placed in or near the experiment to optimize the neutron flux detected. For the Zeus experiment, all four He^3 detectors are placed inside the spindle as depicted in Fig. 3.2. It is important to note that the neutron source helping sustain the subcritical measurements is also placed inside the spindle as far away from the detectors as possible.

Once in active remote operations, delayed critical in the presence of a source is found. Finding delayed critical is beneficial becasue the worth of the detectors in the spindle can be calculated. At this point, Comet is auto run-out. An auto run-out is similar to a SCRAM in which the experiment is returned to an extremely subcritical state. The auto run-out is performed to allow the neutron population to decay away. Once enough time has passed, the neutron population will return to our background level. Now, the neutron population has decayed sufficiently to perform the measurement, the experiment is reassembled to the configuration necessary to perform the measurement. In the Zeus experiment, mass separation controls the reactivity. For this thesis, measurements are taken at separations of 100, 65, 57, 51, and 48 mils. The first three measurements are all subcritical. The measurement at 51 mils being is delayed critical with the source inside the assembly, and the measurement at 48 mils is supercritical. Due to time restrictions, only one measurement is completed at each separation, so there is no experimental determination of the error in the measurement. Ideally, a further measurement would be taken of the system at critical without the source present.

CHAPTER 4

RESULTS

4.1 Experimental Analysis

Subcritical, critical, and supercritical measurements were performed on the Zeus experiment on the Comet assembly using the Rossi-α method. The subcritical measurements were performed at separations of 100, 65, and 57 mils. These measurements relate to a reactivity of -38.24, -10.24, and -3.84 cents respectively. The critical measurement was performed at the critical separation of 51 mils which was measured to be approximately 0.25 cents. This measurement was the closest to critical the encoders on the Comet assembly were able to get to a critical configuration. In reality 51 mils, was slightly supercritical and 52 mils was slightly subcritical. It is also important to note that this measurement was performed in the presence of a source. One supercritical measurement was performed at a separation of 48 mils which corresponds to 3.36 cents.

The measurements for the supercritical and critical separations are compared to the results obtained from the linear fit of the subcritical data points. The graphical results of binning and fitting the data taken at separations of 100, 65, 57, 51 and 48 mils are visualized in Fig. 4.1, 4.2, 4.3, 4.4, and 4.5 respectively.

The fit for the supercritical value, shown in Fig. 4.5, has a single channel spikes which do not follow the rest of the trend. This channel records many more than the expected or even the maximum number of counts. Becasue these outliers are from a single channel it does not have a significant impact on the results, but should be corrected for future experiments. These outliers are most likely the result of signal reflections on the extremely long BNC cables used to transmit the data from the point of measurement to the control room leading to double pulsing. These refections are eliminated in later experiments by adding a 50Ω

terminator to the end of the BNC cables.

The values for A, B, and α found through implementation of these fits using Origin Pro can be found in Table 4.1. Table 4.1 also shows the information that is used to further analyze these values and obtain the value of α at delayed critical. The values of α recorded in table 4.1 come from the non-linear least squares fits performed using Origin Pro graphing software. The histograms built by the binning program in Appendix A as well as the fits produced by Origin Pro are shown in Fig. 4.1, 4.2, 4.3, 4.4, and 4.5. These figures show the histogram in blue and the fit line in red. Further analysis includes plotting the value of α vs. the inverse of the count rate measured during that measurement, and then fitting a line through these points. Figure 4.6 provides a visualization of this linear fitting used to determine the value of α at delayed critical. The y-intercept of the linear fit is the value of α at delayed critical which as shown in Fig. 4.6 is $-89910\ s^{-1}$.

The value of α at delayed critical is both experimentally measured and extrapolated from the subcritical data points. The measured value of α is found to be $\alpha_{measured} = -90408.4\ s^{-1}$, and the value of α extrapolated from subcritical data is found to be $\alpha_{fit} = -89910\ s^{-1}$. Percent difference between the measured and extrapolated values of α is about 0.55%.

$$\%_{difference} = \frac{\alpha_{fit} - \alpha_{measured}}{\alpha_{fit}} * 100 = 0.55\% \tag{4.1}$$

Without knowledge of the uncertainty in the value of α it is hard to determine if there is any significant difference between these values, but the origin of this difference potentially stems from the slight super-criticality of the delayed critical measurement (approximately 0.25¢), or the fact that the measurements at are performed using a source. Another potential source of this difference is the way the data is fit including disregarding channels that contained dead time.

The neutron lifetime of the system is calculated using the value of α determined by the linear fit of the subcritical data points and the known value of β_{eff} for all-HEU systems, 0.0065 pcm. The neutron lifetime for the all-HEU Zeus experiment is calculated using Eq. 2.18 to be $7.23 * 10^{-8}$ s.

Using the reactivity calibration developed using similar triangles, Eq. 2.20 gives the value

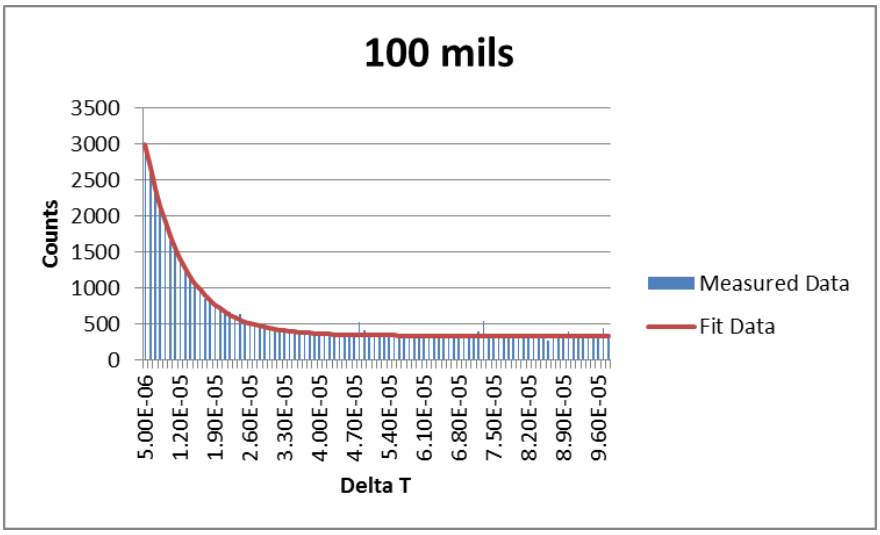

Figure 4.1: Data taken on the all-HEU Zeus experiment at a separation of 100 mils. During experimentation the software's best guess for the reactivity of this configuration is 38.24¢ below critical.

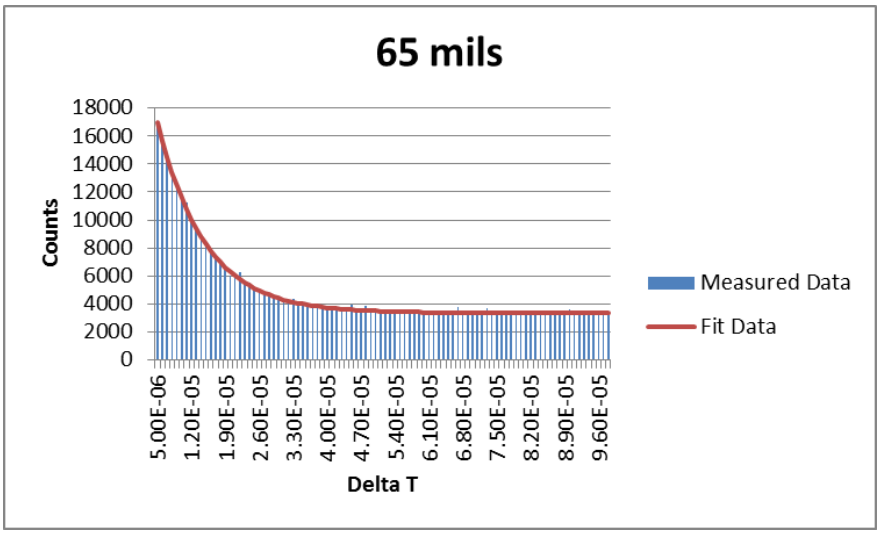

Figure 4.2: Data taken on the all-HEU Zeus experiment at a separation of 65 mils. During experimentation the software's best guess for the reactivity of this configuration is 10.24¢ below critical.

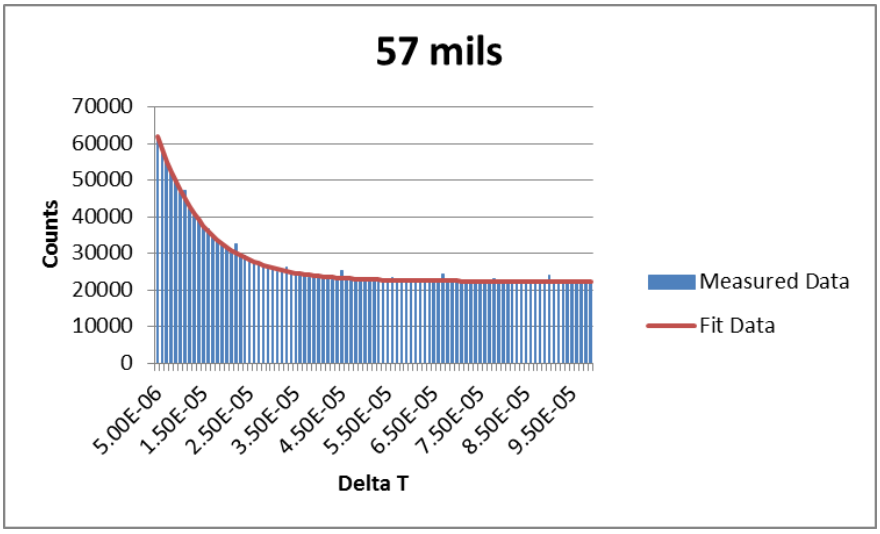

Figure 4.3: Data taken on the all-HEU Zeus experiment at a separation of 57 mils. During experimentation the software's best guess for the reactivity of this configuration is 3.84¢ below critical.

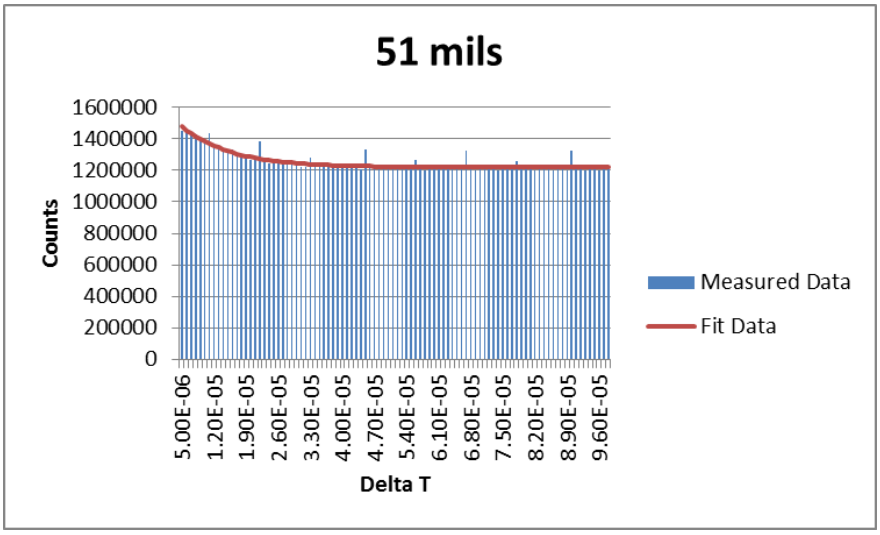

Figure 4.4: Data taken on the all-HEU Zeus experiment at the critical separation of 51 mils.

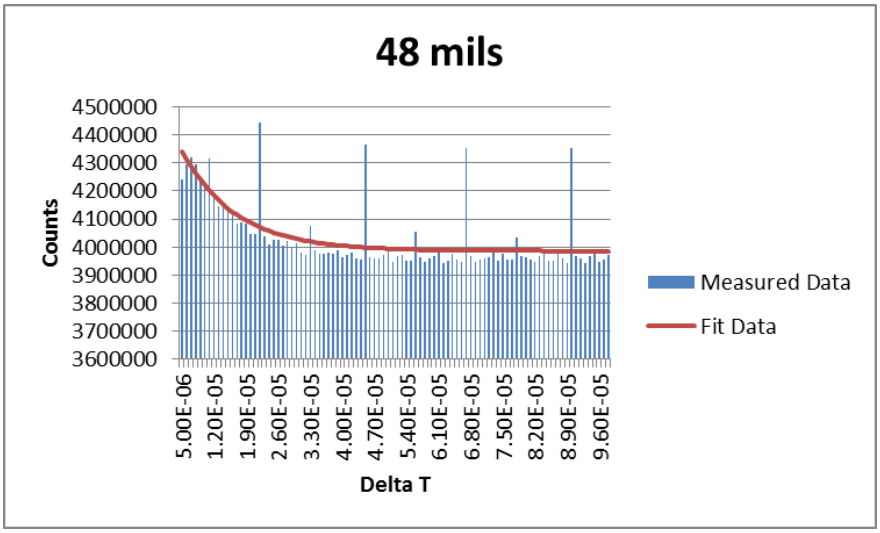

Figure 4.5: Data taken on the all-HEU Zeus assembly at a separation of 48 mils. During experimentation the software's best guess on the reactivity of this configuration is 3.36¢ above critical.

Table 4.1: The values of A, B, and α found in the fitting of the data taken on Comet at separations of 100, 65, 57, and 51 mils. For these data points, an all-HEU core was used with the Zeus experiment reflector. The reactivities in this chart refer to those obtained from the machine software during execution.

Fitting Information					
Separation (in.)	ρ (¢)	inverse $\dot{c}$	α s^{-1}	A	B
0.100	-38.24	$1.233*10^{-3}$	-130207	334.2319	5102.356
0.065	-10.24	$3.876*10^{-4}$	-101846	3344.9132	22703.75
0.057	-3.84	$1.496*10^{-4}$	-95310.6	22389.441	63762.12
0.051	0.25	$2.026*10^{-5}$	-90408.4	$1.22*10^{6}$	404413.3
0.048	3.36		-83926.4	$3.99*10^{6}$	540737.6

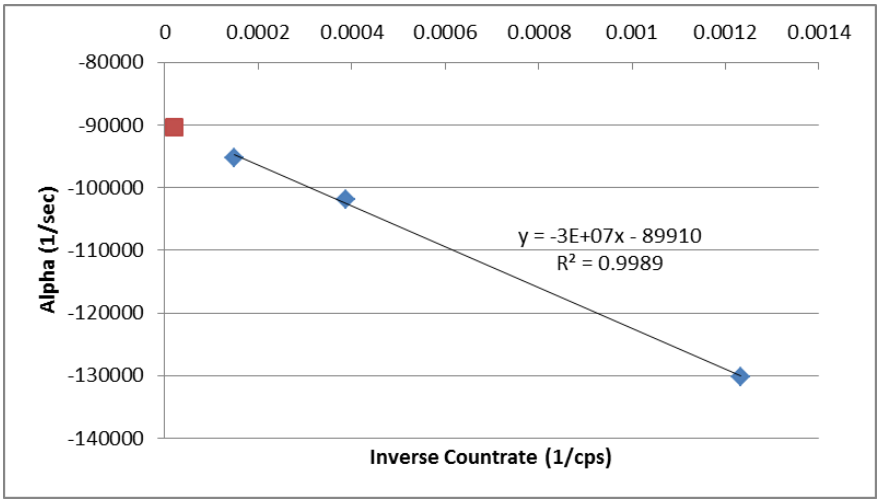

Figure 4.6: Values of α graphed against the inverse countrate, $\dot{c}$. Including the linear fit used to extrapolate the value of α at delayed critical.

Table 4.2: Reactivity of measurements made on the all-HEU Zeus assembly calculated using the linear fit value of α at delayed critical and the method of similar triangles developed in Eq. 2.20.

Reactivity Determined by α		
Separation (in.)	ρ (¢)	$\alpha\ s^{-1}$
0.100	-44.8195	-130207.2
0.065	-13.2758	-101846.2
0.057	-6.0067	-95310.6
0.048	6.6551	-83926.4

of the reactivity of the system at each subcritical data point. The calculated values of these reactivity levels are shown in Table 4.2. These calculated values of reactivity are also plotted against the value of α as shown in Fig. 4.7.

4.2 Computational Analysis

The all-HEU Zeus experiment is part of the International Handbook of Evaluated Criticality Safety Benchmark Experiments book listed as HEU-MET-FAST-073 [11]. Using the MCNP deck from the benchmark, the value of α is calculated. This deck tracks $10,000$ particles for 650 histories of which the last 600 are used to determine quantities of interest. The model

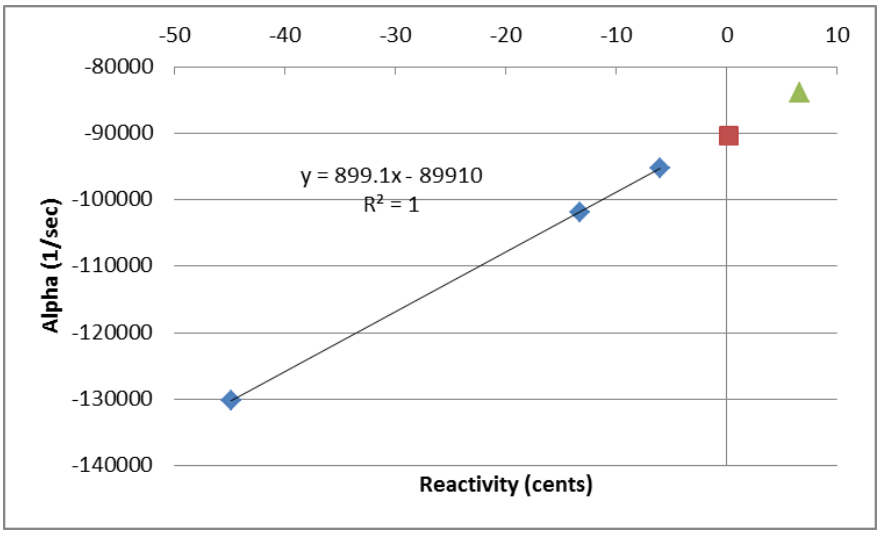

Figure 4.7: α values versus the reactivity level of each measurement.

returns $k_{eff} = 1.0082 \pm 0.0003$ which is the stated k_{eff} in the benchmark. The value of Rossi-α determined by MCNP during this run is $-107586 \pm 0.608\ s^{-1}$. In comparison to the value determined experimentally for Rossi-α, the value calculated by MCNP differs by 19.6%. This difference is found using Eq. 4.1 and the experimentally calculated value of $\alpha = -89910\ s^{-1}$. The large discrepancy can be attributed to the cross section set used in the calculation or imperfections in the physics of the Rossi-α package. In either case, it is beneficial to modify the input deck in such a way that the k_{eff} is nearly one. Although the value of k_{eff} may not seem that far from one, the system is prompt critical assuming the value of $\beta_{eff} = 0.0065$. It seems imperative that the input deck be modified in such a way that k_{eff} is closer to one.

Modification of k_{eff} can be accomplished either by changing the density or the geometry. In this case, modification of the density is preferred becasue the geometry is already the same as the experimental geometry. Although the density has also been measured and correctly input into the code, some quantity must give be modified or the code cannot properly calculate the value of α. The density change is achieved by systematically decreasing the density of the fuel until the k_{eff} is near one. This process is shown by Table 4.3.

Table 4.3 shows how the value of k_{eff} responds to a change in the density. Particularly, this table shows the density modification of the fuel. The Zeus fuel is comprised of an

Table 4.3: Density modification of the fuel in the Zeus experiment benchmark. All densities are given in $\frac{atoms}{barn-cm}$.

Reactor Parameters for Various Assemblies		
Inner Plate Density	Outer Plate Density	k_{eff}
0.048622	0.047810	1.00820 ± 0.00027
0.047000	0.046000	0.98918 ± 0.00028
0.047500	0.047500	0.99907 ± 0.00026
0.047600	0.047600	0.99945 ± 0.00028
0.047620	0.047620	1.00005 ± 0.00027

outer ring and either an inner disk or ring as discussed previously [11]. For the density modification, first all fuel is assumed to be the same density and then the density of the fuel is systematically decreased until a more reasonable value of k_{eff} is achieved.

After the density modification, the value of k_{eff} is very close to one $k_{eff} = 1.00005 \pm 0.0003$. The value of Rossi-$\alpha$ determined from the MCNP calculation becomes $-100048 \pm 0.584\ s^{-1}$. When compared to the experimental value determined for Rossi-α, the value calculated for the Rossi-α differs by 11.3%. The difference between the experimental and MCNP calculated value of Rossi-α is similar to the results presented in the Rossi Alpha MCNP Validation Suite [13].

CHAPTER 5

CONCLUSIONS

This thesis completed the experimental determination of Rossi-α measurements at several different configurations of the Zeus experiment including the subcritical, delayed critical, and supercritical regimes. The values taken at subcritical configurations of Zeus were successfully used to calculate the value of α while at delayed critical. Using the delayed critical value of α along with the definition of α at prompt critical (α=0), a reactivity calibration was determined for the all HEU Zeus experiment.

The values determined for α at delayed critical fit well when compared to historical data from assemblies with varying hardness of neutron spectrum. The measured values of $\alpha_{DC} = -89910$ and $l = 7.23\text{x}10^{-8}$ agree well with historical data from other fast systems. The comparison in Table 5.1 provides the values of α at delayed critical and the neutron lifetime for five different critical assemblies. Each assembly has a slightly different neutron spectrum and they are listed from the hardest to the softest. Lady Godiva has the hardest neutron spectrum because it was bare HEU 94% enriched. Godiva IV has a slightly softer spectrum because it consists of bare HEU enriched to 93% and alloyed with 1.5 wt% Mo. Topsy had an HEU core enriched to 94% reflected by thick natural uranium. Zeus as previously described is an HEU assembly enriched to 93% reflected by copper. SHEBA stands for solution high energy burst assembly which consists of 4.9% enriched uranyl fluoride. The value of α at delayed critical and therefore the neutron lifetime of the Zeus assembly fit into Table 5.1 where expected based on the materials and spectrum involved [10].

Another way to determine the validity of the results obtained through the Rossi-α measurement is through the comparison to the value of α calculated using neutron transport codes like MCNP. When comparing the experimentally determined value of α to the calculated one found in MCNP, the values differ by 11.3%. The difference of 11.3% is similar to

Table 5.1: The neutron lifetime and α at delayed critical for the measurements completed and historical data [10].

Reactor Parameters for Various Assemblies		
Assembly	$\alpha\ s^{-1}$	Neutron Lifetime (l)
Lady Godiva	$-1.1\text{x}10^{6}$	$5.9\text{x}10^{-9}$
Godiva IV	$-8.4\text{x}10^{5}$	$7.7\text{x}10^{-9}$
Topsy	$-3.7\text{x}10^{5}$	$1.75\text{x}10^{-8}$
Zeus	$-8.9\text{x}10^{4}$	$7.23\text{x}10^{-8}$
SHEBA	-200	$4.0\text{x}10^{-5}$

Table 5.2: Comparison of the experimental value of α at delayed critical to the value of α measured by MCNP using the ENDF/B-VI cross section set [13].

Experimental vs. Calculated Values of the Rossi-α				
Assembly	Moderator	$\alpha_{ex}\ s^{-1}$	$\alpha_{calc}\ s^{-1}$	Difference
Lady Godiva	None	-1.11×10^{6}	-1.14×10^{6}	2.70%
Flattop	U^{238}	-3.82×10^{5}	-4.09×10^{5}	7.06%
Zeus	Graphite	-3.38×10^{3}	-3.73×10^{3}	10.36%
Zeus	Iron	-3.73×10^{4}	-4.12×10^{4}	10.46%
Zeus	None	-8.99×10^{4}	-10.0×10^{4}	11.27%

the differences seen in the Rossi Alpha Validation Suite for MCNP as shown on Table 5.2 [13].

CHAPTER 6

FUTURE WORK

Overall, the experiment and subsequent calculations were successful. To improve the experiment in the future a few changes should be made. The easiest improvement is to add terminators on the long data lines to reduce double pulsing. Another hardware change that can be made to reduce dead time and improve measurements on fast systems would be to include additional channels. Other observations made to improve the data acquisition include taking multiple measurements at each data point to determine uncertainty in each value. Also, measurements should not be made when the count rate is too close to the saturation rate for the detection system. The final addition should be to take one measurement with no source of the Rossi-α at delayed critical.

APPENDIX A

C++ CODE

The following C++ code bins the data taken using a 32 channel listmode. This code uses methodology from the book Numerical Recipes in C [14] and the website cplusplus.com was used as a general C++ reference [15].

```
1  /*
2   * main.cpp
3   *
4   *  Created on: Jun 18, 2014
5   *      Author: George McKenzie
6   *
7   *      Program bins and fits list-mode data for the Rossi Alpha Method
8   */
9
10 #include <iostream>
11 #include <fstream>
12 #include <string>
13 #include <sstream>
14 #include <stdlib.h>
15 #include <stdio.h>
16 #include <math.h>
17 #include <cmath>
18
19 using namespace std;
20
21 int lines (string file){
22   // this function counts the number of lines in the file
23   // open data file
```

```
24      ifstream inputFile;
25      inputFile.open (file.c_str(), ios::in);
26    //moves through rows of file until end is reached keeping track with count
27      int count=0;
28      if (inputFile.is_open()){
29          string data;
30
31          while (!inputFile.eof()){
32
33            getline(inputFile,data);
34            count++;
35          }
36          inputFile.close();
37        }
38      else
39        cout << "lines failed";
40
41    // last line is a zero so subtract 1
42      count=count-1;
43
44    return count;
45  }
46
47  void buildArray(ifstream& file, double data[], int matrixLength, int stopPoint
        , int count){
48
49    // this loop only occurs if buildArray is called multiple times
50      double dummyArray[matrixLength];
51
52      for(int q=0; q<matrixLength; q++)
53        dummyArray[q]=0.0;
54
55      if(count>100000 && stopPoint<matrixLength){
56
57        // save data that has not yet been used in the array
58          for(int p=stopPoint; p<matrixLength; p++){
```

```
59              dummyArray[p-stopPoint]=data[p];
60            }
61        }
62
63    // zero the data array
64        for(int j=0; j<matrixLength; j++){
65          data[j]=0.0;
66        }
67    string line;
68    for(int i=0; i<matrixLength; i++){
69        if(i<(matrixLength-stopPoint))
70          data[i]=dummyArray[i];
71        else{
72          getline(file,line);
73          data[i]=strtod(line.c_str(), NULL);
74        }
75
76    }
77
78 }
79
80 void timeInitialize(double timearr[], double bin, int numBins){
81
82    // time of the beginning of the bin
83    double timeStart=0.0;
84    // fill first column with the time the bin begins and fill the second column
          with zeros
85    for(int i=0;i<=(numBins+1);i++){
86        timearr[i]=timeStart;
87        timeStart=timeStart+bin;
88    }
89 }
90
91 void binInitialize(int binarr[], int numBins){
92
93    // fill first column with the time the bin begins and fill the second column
```

```
        with zeros
94    for(int i=0;i<=numBins;i++){
95      binarr[i]=0;
96    }
97  }
98
99  int binning1(double data[], double timearr[], int binarr[], double window, int
        numBins, int matrixLength, int stopPoint){
100
101   // this method bins the counts assuming that every count starts a new time
        window
102
103   // set the final time
104     double tEnd=data[0];
105     for(int q=1; q<matrixLength; q++){
106       if(tEnd<data[q])
107         tEnd=data[q];
108     }
109
110     cout<<tEnd<<endl;
111
112   //counts through each point  in the data
113     for(int i=0; i<matrixLength; i++){
114
115       double tstart=data[i];
116
117       // makes sure the window does not go past the end of the data set
118         if((tstart+window)<tEnd){
119
120           // moves through the time window until the end is reached
121             for(int j=i+1; j<matrixLength; j++){
122               double tcurrent=data[j];
123               double timeDifference=tcurrent-tstart;
124
125               // ensures the point is inside the window
126                 if(tcurrent<=(tstart+window) && data[j]!=0.0){
```

```
127
128                     // perform the binning
129                       for(int z=0; z<numBins; z++){
130                         if(timearr[z]<=timeDifference && timearr[z+1]>
                                timeDifference)
131                           binarr[z]=binarr[z]+1;
132                       }
133                   }
134                   else
135                     j=matrixLength+10;
136
137               }
138           }
139           else{
140             stopPoint=i;
141             i=matrixLength+10;
142           }
143
144       }
145
146     return stopPoint;
147 }
148
149 /*
150 void binning2(double data[], double timearr[], int binarr[], double window,
        int numBins, int count){
151
152     // this method bins the counts assuming that a new bin starts immediately
           after the previous bin
153
154     // set the final time
155       double tEnd= data[count-1];
156
157     //counts through each point  in the data
158       for(int i=0; i<count; i++){
159
```

```
160         double tstart=data[i];
161
162         // makes sure the window does not go past the end of the data set
163           if((tstart+window)<tEnd){
164
165             // moves through the time window until the end is reached
166               for(int j=i+1; j<count; j++){
167                 double tcurrent=data[j];
168                 double timeDifference=tcurrent-tstart;
169
170                 // ensures the point is inside the window
171                   if(tcurrent<=(tstart+window)){
172
173                     // perform the binning
174                       for(int z=0; z<numBins; z++){
175                         if(timearr[z]<=timeDifference && timearr[z+1]>
                              timeDifference)
176                           binarr[z]=binarr[z]+1;
177                       }
178                   }
179                   else{
180                     i=j+1;
181                     j=count+10;
182                   }
183               }
184           }
185           else
186             i=count+10;
187
188       }
189
190 }
191
192 void binning3(double data[], double timearr[], int binarr[], double window,
      int numBins, int count){
193
```

```
194   // this method bins the counts with random initiating events
195     int r;
196     int totalWindows=count*0.9;
197
198   // set the final time
199     double tEnd= data[count-1];
200
201   //counts through each point  in the data
202     for(int i=0; i<totalWindows; i++){
203
204       // get random number
205         r=rand() %(count-1);
206         double tstart=data[r];
207
208       // makes sure the window does not go past the end of the data set
209         if((tstart+window)<tEnd){
210
211           // moves through the time window until the end is reached
212             for(int j=r+1; j<count; j++){
213               double tcurrent=data[j];
214               double timeDifference=tcurrent-tstart;
215
216               // ensures the point is inside the window
217                 if(tcurrent<=(tstart+window)){
218
219                   // perform the binning
220                     for(int z=0; z<numBins; z++){
221                       if(timearr[z]<=timeDifference && timearr[z+1]>
                              timeDifference)
222                         binarr[z]=binarr[z]+1;
223                     }
224                 }
225                 else
226                   j=count+10;
227
228             }
```

```
229           }
230           else
231             i=count+10;
232
233       }
234
235 }
236 */
237
238 double findA(int binsarr[],int numBins){
239
240   // the constant term for the Rossi alpha fit
241     double A;
242   // use the last half of the bins but not the last 10%
243     int unusedBins1=numBins*0.5;
244     int unusedBins2=numBins*0.9;
245     int sum=0;
246   // sums up all the bins
247     for (int i=unusedBins1; i<=unusedBins2; i++){
248       sum=sum+binsarr[i];
249     }
250   // create the average B
251     A=sum/(unusedBins2-unusedBins1);
252
253   return A;
254 }
255
256 void fitLine(int binsarr[], double time[], int numBins, double A){
257
258   // the function linearizes the data and fits a line to it
259
260   // find the max value of the data so that we do not fit the dead time
261     int max=0;
262     int dataStart=0;
263     for (int i=0; i<numBins;i++){
264       if (binsarr[i]>=max){
```

```
265           max=binsarr[i];
266           dataStart=i;
267         }
268       }
269     // create dummy array to manipulate data
270       double dataBins[numBins];
271       for(int j=0; j<numBins; j++){
272         dataBins[j]=binsarr[j]-A;
273
274         if(dataBins[j]>0)
275           dataBins[j]=log(dataBins[j]);
276         else
277           dataBins[j]=0;
278       }
279
280     // iterates through the data points so the user can choose the best fit
281       for(int g=5; g<26; g++){
282
283       // initialize the start and end of the data to be fit
284         int dataLength=g;
285         int dataEnd= dataStart+dataLength;
286
287       // create the sums for the fitting
288         double sumN=0.0;
289         double sumX=0.0;
290         double sumY=0.0;
291         double sumXX=0.0;
292         double sumXY=0.0;
293         double chisquare=0.0;
294         double residuals=0.0;
295         double total=0.0;
296         double yBar=0.0;
297         double rSquared=0.0;
298
299         for (int z=dataStart; z<=dataEnd; z++){
300           sumN+=1;
```

```
301           sumX+=time[z];
302           sumY+=dataBins[z];
303           sumXX+=(time[z]*time[z]);
304           sumXY+=(time[z]*dataBins[z]);
305           yBar+=dataBins[z];
306         }
307
308       // calculate the values
309         double denom=sumN*sumXX-(sumX*sumX);
310         double intercept=sumXX*sumY-sumX*sumXY;
311         double slope=sumN*sumXY-sumX*sumY;
312         intercept=intercept/denom;
313         slope=slope/denom;
314         yBar=yBar/dataLength;
315
316       // calculates chi squared
317         for (int i=dataStart; i <= dataEnd; i++){
318           chisquare+=(dataBins[i]-intercept-slope*time[i])*(dataBins[i]-
                intercept-slope*time[i]);
319           residuals+=(dataBins[i]-intercept-slope*time[i])*(dataBins[i]-
                intercept-slope*time[i]);
320           total+=(dataBins[i]-yBar)*(dataBins[i]-yBar);
321         }
322
323       // set outputs
324         rSquared=1-(residuals/total);
325         intercept=exp(intercept);
326         //cout << "B= " << intercept << endl;
327         cout << g << " " << A << " " << intercept << " " << slope << " " <<
              rSquared << " " << chisquare  << endl;
328
329
330       }
331
332 }
333
```

```
334 int main(){
335
336   // reminds user to remove header and footer
337     cout << "Warning only for 32 Channel Listmode."<< endl << endl;
338
339   // initialize binning variables
340     cout<< "What is the time window we want to look at? (seconds)"<< endl;
341     string Window;
342     getline(cin,Window);
343     double window=atof(Window.c_str());
344
345     cout<< "What is the bin width we want to look at? (seconds)"<< endl;
346     string Bin;
347     getline(cin,Bin);
348     double bin=atof(Bin.c_str());
349
350   // kills program if bin size is greater than the time window
351     if (bin>window)
352       return 0;
353   // begins calculations necessary for binning
354     int numBins=window/bin;
355     cout<< numBins << " bins created."<<endl;
356     double timeForbins[numBins+1];
357     int bins1[numBins];
358
359   // initialize the bin array with zeros
360     binInitialize(bins1, numBins);
361
362   // initialize fitting parameters
363     double A=0.0;
364
365
366   // initialize the time array
367     timeInitialize(timeForbins, bin, numBins);
368
369   // create filename to open
```

```
370     string filename="output.txt";
371
372   // build data matrix
373     int count = lines(filename);
374     int matrixLength;
375
376     if(count>100000)
377       matrixLength=100000;
378     else
379       matrixLength=count;
380
381     double data[matrixLength];
382     int stopPoint=matrixLength;
383
384   // open file with data in it
385     ifstream file;
386     file.open (filename.c_str(), ios::in);
387
388     if(file.is_open()){
389
390       // loops through the output file until end is reached
391         while(!file.eof()){
392
393           //  make a matrix with the first 100,000 counts or less
394             buildArray(file, data, matrixLength, stopPoint, count);
395
396           // bin the data both ways
397             stopPoint=binning1(data, timeForbins, bins1, window, numBins,
                    matrixLength, stopPoint);
398         }
399
400         file.close();
401     }
402
403   // find the constant A
404     A=findA(bins1, numBins);
```

```
405
406   // fit the line and find B and alpha
407     fitLine(bins1, timeForbins, numBins, A);
408
409   // bin output to results file
410     ofstream resultsFile;
411     resultsFile.open("results.txt", ios::out | ios::trunc);
412
413     if(resultsFile.is_open()){
414         for(int i=0; i<numBins; i++){
415         resultsFile <<  timeForbins[i] <<  " "  << bins1[i] << endl;
416       }
417         resultsFile.close();
418     }
419
420     cout << "Done" << endl;
421
422   return 0;
423 }
```

This code cuts down the size of the original data file so that the computer is not overwhelmed during execution of the previous code.

```
1  /* main.cpp
2   *
3   * Created on: Jun 26, 2014
4   *       Author: George McKenzie
5   *
6   */
7
8  # include <iostream>
9  # include <fstream>
10 # include <string>
11 # include <sstream>
12 # include <stdlib.h>
13 # include <stdio.h>
14 # include <math.h>
15
16 using namespace std;
17
18 void arrayChannels (int Channels[]){
19
20   // opens file where the used channel numbers is
21        ifstream inputFile;
22        inputFile.open("channels.txt", ios::in);
23
24   // this function builds an array that has ones in every channel spot that
         the user cares about
25      for(int i=0; i<=32; i++){
26        Channels[i]=0;
27      }
28
29
30
31   // ensures this portion of code only runs when the file is open
```

```
32      if (inputFile.is_open()){
33
34          string line;
35
36          while(!inputFile.eof()){
37            getline(inputFile,line);
38            int channel= atoi(line.c_str());
39
40            for(int i=1; i<=32;i++){
41              if(channel==i)
42                Channels[i]=1;
43            }
44
45          }
46          inputFile.close();
47      }
48      else
49        cout << "channels.txt did not open." << endl;
50  }
51
52  int getlines (string filename){
53    // this function counts the number of lines in the file
54    // open data file
55      ifstream inputFile;
56      inputFile.open (filename.c_str(), ios::in);
57    //moves through rows of file until end is reached keeping track with count
58      int count=0;
59      if (inputFile.is_open()){
60          string data;
61
62          while (!inputFile.eof()){
63
64            getline(inputFile,data);
65            count++;
66          }
67          inputFile.close();
```

```
68        }
69      else
70        cout << "lines failed";
71
72    // last line is a zero so subtract 1
73      count=count-1;
74
75    return count;
76  }
77
78  int main(){
79    // matrix of the channels used in the output file
80        int Channels[32];
81
82    // gets the name of the data file from the user
83      string filename;
84      cout << "What is the filename? (Include .txt)"<<endl;
85      getline (cin, filename);
86
87    // calls channels.txt and builds an array that holds the importance of
          individual channels
88      arrayChannels(Channels);
89
90    // creates an output file that will only include times
91      ifstream inputFile;
92      inputFile.open(filename.c_str(), ios::in);
93      string currentLine;
94    // create constants used to generate output file
95      string currentTime;
96      int currentValue;
97      int totalLines=getlines(filename);
98      string outputfilename;
99      int position=0;
100     int linesWritten=0;
101
102     if(inputFile.is_open()){
```

```
103
104         // creates a file to output the information to
105           outputfilename="output.txt";
106           ofstream writeFile;
107           writeFile.open(outputfilename.c_str(), ios::out| ios::trunc);
108
109         // ensures file is open
110           if(writeFile.is_open()){
111
112             // skips the header lines
113               for(int i=0; i<5; i++){
114                   getline(inputFile,currentLine);
115                   position++;
116                 }
117             // loops until it hits the text at the bottom
118               for(int i=0; i<totalLines;i++){
119                 for(int j=0; j<32; j++){
120
121                   // get the line from the file
122                     getline(inputFile, currentLine,',');
123
124                   // records the time for this line
125                     if(j==0){
126                       currentTime= currentLine;
127                       position++;
128                     }
129                     else
130                       currentValue= strtod(currentLine.c_str(), NULL);
131                   // records the time if the value is 1
132                     if(currentValue==1 && Channels[j]==1){
133                       writeFile << currentTime << endl;
134                       linesWritten++;
135                     }
136                 }
137                 getline(inputFile, currentLine);
138                 if(currentValue==1 && Channels[32]==1){
```

```
139                     writeFile << currentTime << endl;
140                     linesWritten++;
141                 }
142               }
143
144               writeFile.close();
145             }
146
147         }
148
149         inputFile.close();
150
151
152     cout<< linesWritten <<endl;
153     cout << "Done" << endl;
154
155     return 0;
156 }
```

REFERENCES

[1] H. C. Paxon and G. E. Hansen, "Critical assemblies and associated measurements," unpublished.

[2] C. P. Baker, "Time scale measurements by the Rossi method," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-617, January 1947.

[3] J. L. Alwin and S. P. Monahan, "United States Department of Energy (NCSP) hands-on experimental tranining," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-13-21945, 2013.

[4] R. E. Uhrig, *Random Noise Techniques in Nuclear Reactor Systems*. New York, NY: Ronald.

[5] J. Orndoff and C. Johnstone, "Time scale measurements by the Rossi method," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-744, November 1949.

[6] F. de Hoffman, "Statistical fluctuations in the water boiler and time dispersion of neutrons emitted per fission," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-101, June 1944.

[7] G. E. Hansen, "The Rossi alpha method," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-85-4176, August 1985.

[8] J. D. Orndoff, "Prompt neutron periods of metal critical assemblies," in *Nuclear Science and Engineering*, 1957, vol. 2, pp. 450–460.

[9] R. Feynman, "Statistical behavior of neutron chains," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-591(DEL), July 1946.

[10] R. G. Sanchez, G. E. McKenzie IV, T. J. Grove, and J. A. Bounds, "Measurement of the prompt neutron decay constant in a highly enriched uranium copper reflected system," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-14-21819, March 2014.

[11] *International Handbook of Evaluated Criticality Safety Benchmark Experiments*, Nuclear Energy Agency Std. HEU-MET-INTER-006, 2007.

[12] W. L. Myers, G. J. Arnone, and S. G. Melton, "Use of list-mode data acquisition systems for performing benchmark subcritical neutron measurements," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-09-00231, June 2009.

[13] R. D. Mostellar and B. C. Kiedrowski, "The Rossi Alpha validation suite for MCNP," Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep. LA-UR-11-5077, September 2011.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992.

[15] "C++ Tutorials." [Online]. Available: cplusplus.com