LA-14220-T
Thesis

Nuclear Forensics: Attributing the Source of Spent Fuel Used in an RDD Event

Los Alamos
NATIONAL LABORATORY
EST. 1943

The World's Greatest Science Protecting America

Funding provided by the Defense Threat Reduction Agency.

This thesis was accepted by the Department of Nuclear Engineering, Texas A&M University, College Station, TX, in partial fulfillment of the requirements for the degree of Master of Science. The text and illustrations are the independent work of the author and only the front matter has been edited by the IM-1 Writing and Editing Staff to conform with Department of Energy and Los Alamos National Laboratory publication policies.

Los Alamos National Laboratory, an affirmative action/ equal opportunity employer, is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

LA-14220-T
Thesis
Issued: June 2005

Nuclear Forensics: Attributing the Source of Spent Fuel Used in an RDD Event

Mark Robert Scott

Los Alamos
NATIONAL LABORATORY
EST. 1943

The World's Greatest Science Protecting America

NUCLEAR FORENSICS: ATTRIBUTING THE SOURCE OF SPENT FUEL USED IN AN RDD EVENT.

A Thesis

by

MARK ROBERT SCOTT

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

William S. Charlton
(Chair of Committee)

Ron R. Hart
(Member)

Ali Beskok
(Member)

William E. Burchill
(Head of Department)

May 2005

Major Subject: Nuclear Engineering

ABSTRACT

Nuclear Forensics: Attributing the Source of Spent Fuel Used in an RDD Event.

(May 2005)

Mark Robert Scott, B.S., Brigham Young University

Chair of Advisory Committee: Dr. William S. Charlton

An RDD attack against the U.S. is something America needs to prepare against. If such an event occurs the ability to quickly identify the source of the radiological material used in an RDD would aid investigators in identifying the perpetrators. Spent fuel is one of the most dangerous possible radiological sources for an RDD. In this work, a forensics methodology was developed and implemented to attribute spent fuel to a source reactor. The specific attributes determined are the spent fuel burnup, age from discharge, reactor type, and initial fuel enrichment. It is shown that by analyzing the post-event material, these attributes can be determined with enough accuracy to be useful for investigators. The burnup can be found within a 5% accuracy, enrichment with a 2% accuracy, and age with a 10% accuracy. Reactor type can be determined if specific nuclides are measured. The methodology developed was implemented into a code call NEMASYS. NEMASYS is easy to use and it takes a minimum amount of time to learn its basic functions. It will process data within a few minutes and provide detailed information about the results and conclusions.

ACKNOWLEDGMENTS

My thesis is the accomplishment and effort of many people. I would especially like to thank Dr. William Charlton for his expertise and friendship during my effort to complete this work. Dr. Charlton sacrificed a great amount of time in my behalf and always left the door open in case I needed help. I appreciate the effort by those at Los Alamos and Texas A&M who were involved in this project, which includes: David Burk, Kristen Epresi, Don Giannangeli, Ariane Eberhardt, Erika Leibrecht, and Jessie Ross. In particular, I would like to thank Paula Knepper who went beyond the call of duty to help me with anything I ask for, no matter how small or big.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

A. ATTRIBUTION OF RADIOLOGICAL DISPERSAL DEVICES

The reality of terrorism in the United States has increased the need to seriously consider terrorists using nuclear material. Weapons that use nuclear material include: nuclear weapons, improvised nuclear devices (IND) and radiological dispersal devices (RDD). Nuclear weapons and INDs would cause extensive damage to the U.S. if used, but they are unlikely scenarios because they are probably out of the technical reach of non-state sponsored terrorists and even most state-sponsored terrorists [1]. RDDs have technical hurdles that must be overcome but are within the reach of most terrorist organizations.

An RDD is any device that uses conventional explosives to spread radiological material with the intent to cause damage or injury. One of the more hazardous types of material that could be used in an RDD is spent nuclear fuel. Due to the feasibility of an RDD attack against the U.S., the ability to identify the perpetrators of such an attack is critical. One fundamental piece of forensic evidence which may aid investigators in identifying the perpetrators is knowing the source that produced the material. The objective of this research is to develop a methodology to attribute the radiological material of a detonated RDD to a source reactor. That is, to identify the specific facility that produced the

---

This thesis follows the style of *Nuclear Technology*.

material used in the RDD. Initially a number of likely source reactors will be identified that have the ability to produce radiological material that matches the material used in the RDD. Analysts can then compare the possible source reactor list with databases that contain information on power plants worldwide, the spent fuel they produce, and known cases of nuclear material smuggling. One such database, called SENTRY, exists at Los Alamos National Laboratory (LANL). SENTRY is a Lotus Domino database and contains a large amount of information. It is divided into classified and unclassified areas and is maintained by the LANL group N-3.

Radiological sources that can be used in an RDD vary widely in strength and size. The following are all possible sources of radiological material:

a. Large radioactive sources for industrial radiography (^{226}Ra, ^{192}Ir, ^{60}Co)

b. Large neutron sources (PuBe, AmBe, AmLi, ^{252}Cf)

c. Sources for food irradiation (^{137}Cs, ^{60}Co)

d. Medical isotopes (^{137}Cs, ^{60}Co)

e. Radioisotope thermoelectric generators (^{90}Sr, ^{238}Pu)

f. Well-logging sources (AmBe, PuBe, ^{137}Cs)

g. Spent nuclear fuel

h. Reprocessing facility waste products

Generally, these materials can be divided into two categories: (1) *large radioactive sources* and (2) *spent fuel sources*. Examples a-h are all examples of *large radioactive sources*. These materials are fairly well characterized and generally consist of only one

major radioisotope, and potentially contain several impurities or trace isotopes. These sources vary from low to high levels of radioactivity. Examples g and h are spent fuel sources. Spent fuel sources are poorly characterized in their current state but are well characterized prior to irradiation in a nuclear reactor. These sources are highly radioactive and even a single fuel assembly could kill anyone exposed directly to its radiation.

An RDD's effectiveness can vary greatly depending upon the location of detonation and the radiological source used. A study was conducted to assess the damage of detonating an RDD containing 100 pounds of high explosives at the Washington Monument using various sources [2]. If an RDD contained a typical industrial 5 kCi ^{60}Co radioactive source, the maximum dose would be 12 rem. To put this in perspective, a certified radiological technician, or RAD worker, is allowed to receive 5 rem per year. The increased exposure would be so small that the increased cancer rate would be difficult to measure. If the RDD contained 50 kg of one year old spent fuel rods at 125 kCi the maximum dosage would be 3,064 rem, or six times the lethal dose [2].

Large radioactive sources vary between 10 Ci and 10,000 kCi, with the larger sources used by the food and sterilization industry [3]. A 1,000 kCi industrial ^{60}Co source will decrease to a safer lever of 5 kCi after 40 years. There are less than 300 industrial facilities worldwide that have sources in the range of 10 kCi to 11,000 kCi [4]. The security at these facilities is generally high and transportation of sources above a few kCi

requires significant shielding, making it very difficult to transport covertly. A second possible source is terrestrial radioisotope thermoelectric generators (RTG), which are usually powered by ^{90}Sr at levels as high as 500 kCi [4]. RTGs are used mainly by Russia and the U.S. to provide power in remote locations. They are secured only by their remote locations and by being encased in heavy shielding. A few have been reported stolen [5].

Large radioactive sources are a real threat, but RDDs containing spent fuel have much higher consequences. In the U.S. alone there are over 40,000 tons of spent fuel spread out across the country [6]. The damage a terrorist could cause with spent fuel is very high. The forward model code ORIGEN [7], calculates a pressurized water reactor (PWR) assembly at the time of discharge to be over 230,000 kCi. After 5 years the activity drops to 550 kCi. A spent fuel assembly will not decrease to the safer level of 5 kCi until 300 years after discharge. A comparison of radioactivity versus time for spent fuel is given in Fig. 1. If a terrorist detonated an RDD in a large city with a recently discharged spent fuel assembly as the radioactive source, not only would it kill a large number of people, but it would heavily contaminate the area. This would shut down access to the area for years. Depending on the location of the detonation, the economic impact could be devastating to the U.S. economy. If such an event occurs, this study will help identify the actors involved in perpetrating the attack.

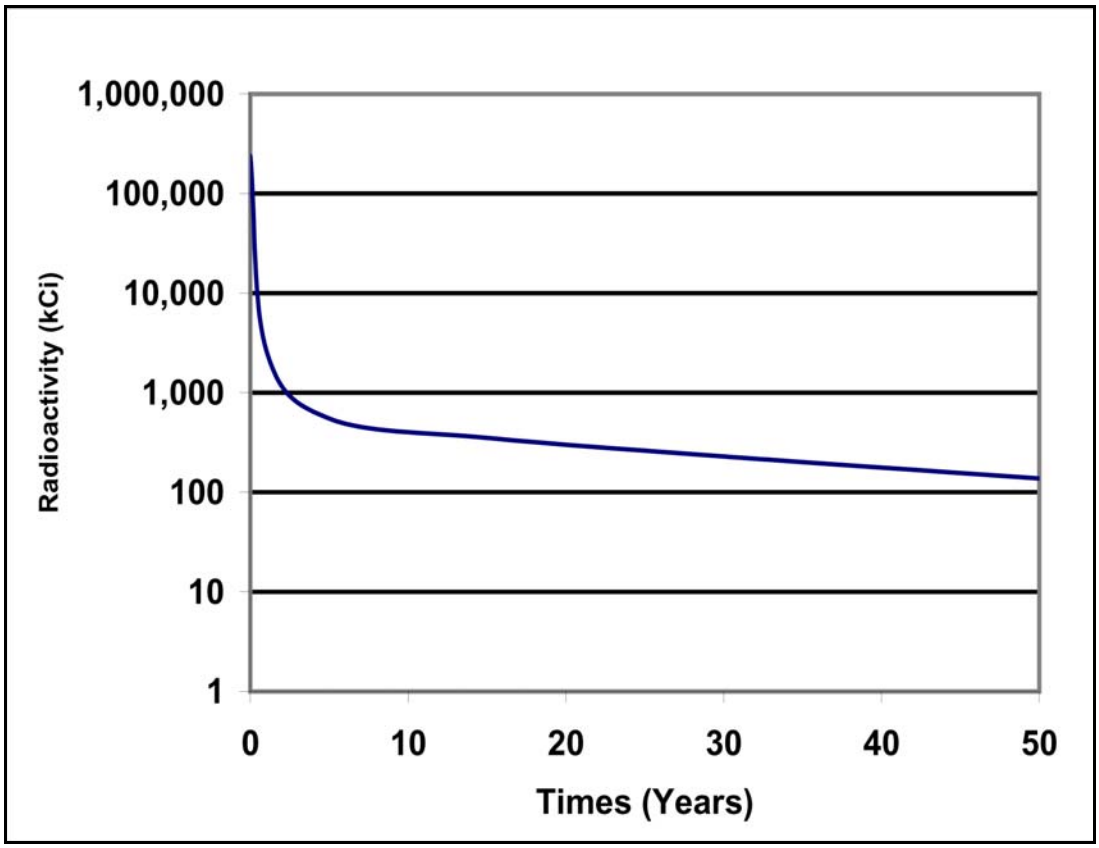

Fig. 1. Radioactivity versus decay time for PWR spent fuel and a large ^{60}Co source.

If a terrorist is able to obtain spent fuel for an RDD, the spent fuel will most likely be an intact fuel assembly. Detonation of an intact assembly as a weapon would not result in a completely uniform dispersion of material. Large pieces of structure, cladding and fuel would be present post-detonation. There are several methods and forensic techniques one could use to identify the origin of the spent fuel in this case. By measuring the composition and size of the structure and cladding, one could narrow down the possible sources considerably. If a more complete dispersion of the material occurs other methods could be used to identify the material. The source could be identified by the isotopic composition of the fuel in the debris. This method would be independent of the

nature of the explosion and structural material present. Even if terrorists ground up the fuel into a powder, to produce even dispersion, isotopics can be used to identify the source.

B. PROJECT OVERVIEW

The work described in this thesis is only part of a larger effort to secure the US against RDDs. Figure 2 shows the complete RDD material attribution project. Once the event occurs, the effort to discover the responsible party will begin immediately. Before forensics can be applied, measurements from the RDD debris must be obtained to determine the isotopic composition of the spent fuel or radiological source. Determining the isotopic composition of spent fuel is complicated and will require significant resources to accurately determine. The preferred method for performing these measurements is with mass spectroscopy. This method is the most accurate for measuring ratios of isotopes from the samples.

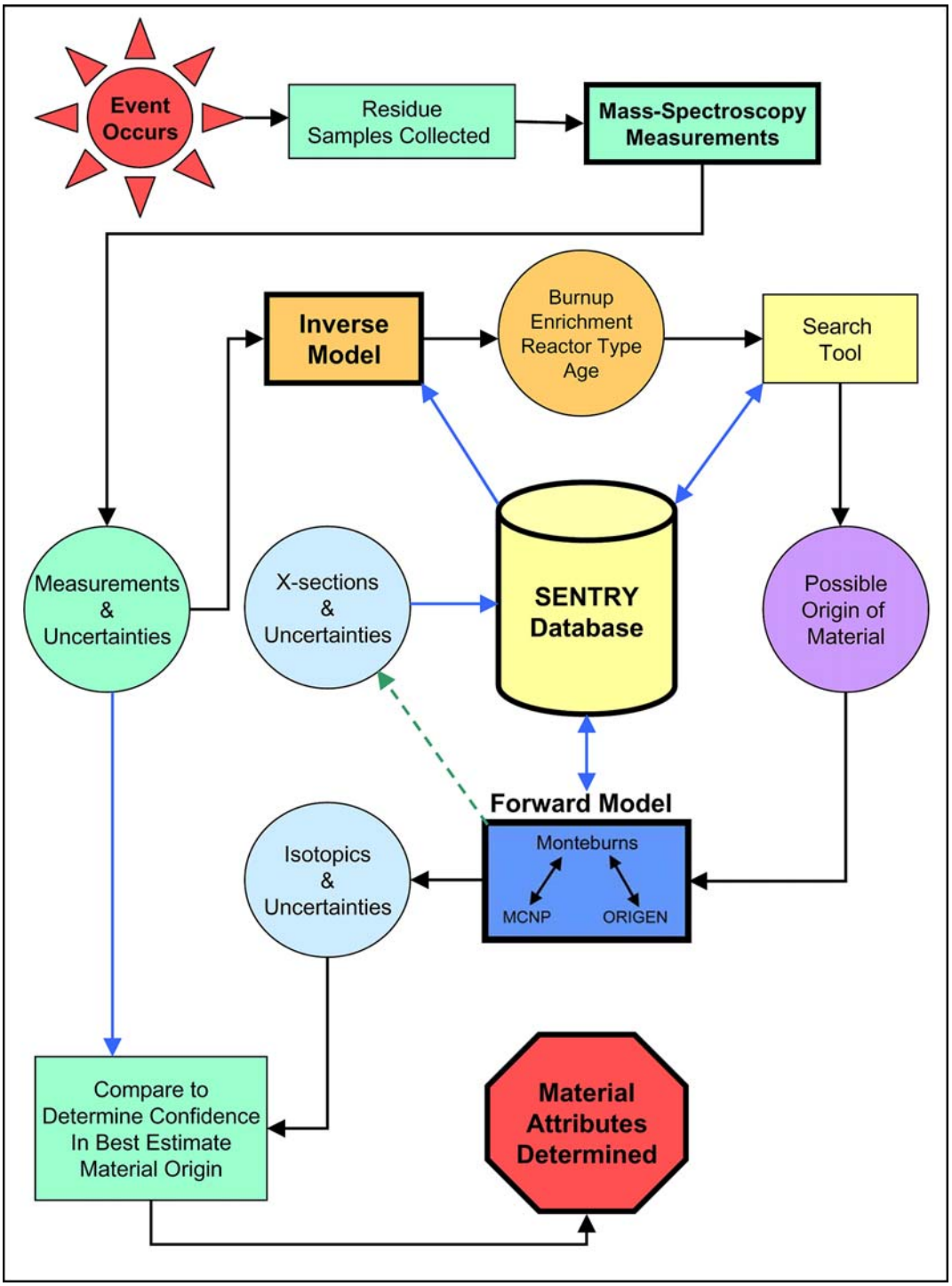

Fig. 2. Overview of RDD material attribution project.

The ability to measure trace actinides in spent fuel with an accuracy better than 1% is necessary. In a 1 gram sample of spent fuel, some of the measured actinides will be present in the microgram range. Mass spectrometry has advanced to the point that isotopic ratios can be measured in the nanogram range for fission products and picogram range for actinides [8,9].

Isotopic composition of spent fuel is not determined on a regular basis because the material is highly radioactive and mass spectrometry is cost prohibitive. The reactor type, fuel enrichment and burnup are generally known prior to the irradiation of the fuel. This allows an analyst to predict the spent fuel isotopics with a forward model code. The composition is usually determined by use of a forward model reactor physics code (e.g. ORIGEN, CASMO, HELIOS, etc). The results from the forward model codes are then benchmarked with a set of actinides that can be determined by either a non-destructive method such as gamma spectrometry or mass spectrometry. Thus a database of present day spent fuel isotopics does not exist with which to compare (and would likely be very cumbersome if it did). Therefore, a forensics methodology is needed which will allow for a comparison of likely reactor properties with known facilities.

Before the inverse process to find source reactors can begin, models of each reactor type must be developed to allow for calculations of present day isotopics from given reactor designs and operational data. These reactor models can be simulated by coupling MCNP (Monte Carlo N-Particle Transport) [10] and ORIGEN. MCNP is a code that is capable

of modeling essentially any reactor type and producing one-group cross sections and scalar fluxes. MCNP uses Monte Carlo techniques to solve the transport of neutrons. In MCNP each interaction is characterized by a probability distribution function. Each particle is tracked, and a history is kept, as it interacts with the material atoms. It can obtain the flux at any area by keeping a history of the particles passing through the area. With the flux, MCNP can calculate reaction rates and collapse to one-group cross sections. MCNP lacks the ability to calculate the burnup and decay of material.

Oak Ridge National Laboratory (ORNL) developed ORIGEN in the 1960s to calculate buildup, burnup and decay of materials. To get an accurate result MCNP and ORIGEN can be run together in steps. First MCNP will run and create one-group cross sections and scalar fluxes for ORIGEN, then ORIGEN will calculate the burnup and decay of the material. The information from the output of ORIGEN will then be used to modify the original MCNP model and then run again. Generally the more steps you take, the more accurate your results will be.

The process of exchanging information back and forth between the MCNP and ORIGEN models is a slow and tedious process. The code MONTEBURNS [11] was developed to simplify this process. MONTEBURNS is a Perl script file that manages the data extraction from the output files and modifying the inputs of both MCNP and ORIGEN. MONTEBURNS gives the user the capability to change the specific power of the reactor core at each step.

The process of creating one-group cross sections and scalar fluxes for the inverse process is shown by the green dotted arrow on Fig. 2. Currently the MCNP model development for cross section generation is being completed by others [12]. This information is stored by SENTRY and used by the inverse model when needed.

Once the cross-sections for each reactor type have been stored in SENTRY, we need to process the measured data to find a source reactor. As shown in Fig. 2. the inverse model, or forensics developed in this paper produces information on the burnup, initial enrichment, reactor type and the age. The characteristics found can then be used by a search tool. This search tool will look for facilities that match the reactor type and uses fuel that matches the initial enrichment. It will then see if the facilities produced spent fuel with the same age and burnup. If it finds any matches it will then check the suspect facilities with the database of known missing material. Only one or two matches will likely be made.

To validate the results, an MCNP model will be created of the suspect reactor. The power history of the suspect fuel will then be retrieved, and a MONTEBURNS deck created. MONTEBURNS will then run a forward model of that specific fuel assembly. The results will be compared to the measured data. This will generally allow for a more accurate match to the measured data.

C. THE THEORY BEHIND THE INVERSE PROBLEM

Fresh fuel is well characterized and contains only trace quantities of other nuclides. As the fuel is burned fission products are created throughout the fuel pin. Each nuclide has a fission yield, which relates how likely it will be produced from fission. Figure 3 shows the normalized yield for each nuclide of the same mass chain in a U.S. PWR reactor.

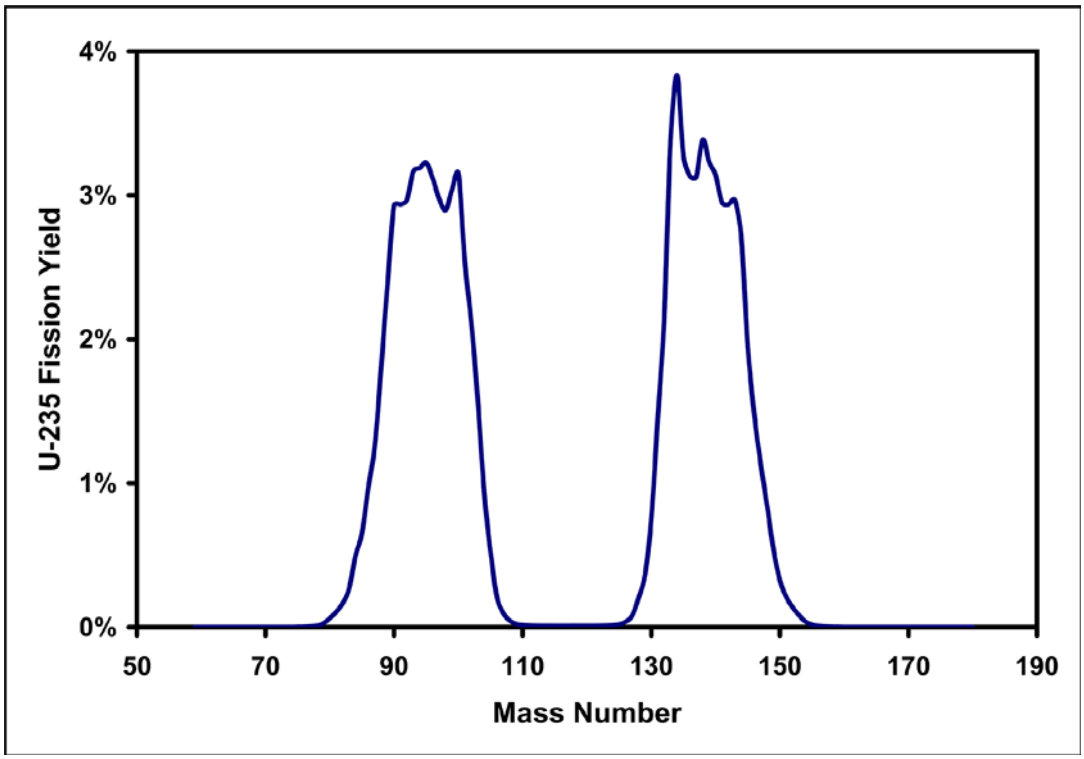

Fig 3. ^{235}U fission yield curve versus isotope mass for a U.S. PWR.

During reactor operation each isotope will be produced at a constant rate in relation to its yield. If no other factors are involved, the production of isotope *B* is given by:

$$\frac{dN_B}{dt} = N^{U235}(t)\int_0^\infty \sigma_f^{U235}(E)Y_B^{U235}(E)\phi(E,t)dE\,, \tag{1.1}$$

where Y_Y^X is the fission yield for the production of isotope *Y* from the fission of isotope *X*, N^X is the atom density of isotope *X*, σ_f^X is the fission cross section of isotope *X*, ϕ is the neutron flux. *E* is the incident neutron energy, and *t* is time. As isotope *B* is produced it will absorb neutrons by radiative capture at a rate related to its capture cross section. By adding this loss term to Eq. 1.1 we get:

$$\frac{dN^B}{dt} = N^{U235}(t)\int_0^\infty \sigma_f^{U235}(E)Y_B^{U235}(E)\phi(E,t)dE - N^B(t)\int_0^\infty \sigma_\gamma^B(E)\phi(E,t)dE \tag{1.2}$$

where σ_γ^X is the radiative neutron capture of isotope *X*. We also have to consider other fission products decaying into isotope *B* and the decay of isotope *B* itself. By accounting for decay, Eq. 1.2 becomes:

$$\frac{dN^B}{dt} = N^{U235}(t)\int_0^\infty \sigma_f^{U235}(E)Y_B^{U235}(E)\phi(E,t)dE - N^B(t)\int_0^\infty \sigma_\gamma^B(E)\phi(E,t)dE + \sum_i \lambda_{i\to B}N_i(t) - \lambda_B N_B(t) \tag{1.3}$$

where $\lambda_{i\to B}$ is the decay constant for isotope *i* that leads to the production of isotope *B*. Typically there are about 100 different fission products produced by a reactor. Each fission product will have an equation such as Eq. 1.3 to represent it. This would result in

over 100 coupled differential equations. Attempting to solve 100 coupled differential equations with both energy and time dependence, without knowledge of the material's initial composition is intractable.

To avoid solving the coupled differential equations with unknown initial conditions, we look for special cases were several assumptions can be made. This allows us to decouple the fission products and simplify Eq. 1.3 to a more manageable equation. By finding these special cases, we can determine enough information about the scenario to run forward models in ORIGEN. We still need more information about the initial conditions, so we run several carefully crafted models until we get results matching the measured data. This is the basis for the inverse model developed in this thesis (Chapter II).

The inverse model must enable an analyst to determined the initial fuel enrichment, burnup, age, and reactor type. These characteristics will be used to uniquely identify the facility from which the fuel was discharged. The initial fuel enrichment refers to the initial enrichment (or the initial ratio of ^{235}U to total uranium) of the fuel prior to irradiation in the reactor. Burnup is the amount of energy per kilogram of fuel and is usually measured in megawatt days per metric ton (MWd/MT). Age refers to the time from the discharge date to the time of measurement. Reactor type is a broad classification given to reactors that have the same general concept design. Some common reactor types are: pressurized water reactors (PWR), boiling water reactors

(BWR), Canadian deuterium uranium reactors (CANDU) and liquid metal fast breeder reactors (LMFBR). Once these characteristics are known, there will only be a few possible source reactors. Analysts can then use this information to compare it with known cases of nuclear theft or diversion.

CHAPTER II

MONITOR DEVELOPMENT

Each fuel characteristic of interest to the forensics problem uses an isotopic monitor. The fuel characteristics of interest here are fuel burnup, fuel enrichment, reactor type, and fuel age. Each characteristic has unique requirements that the monitor must meet. A specific set of monitors must be used for determination of the enrichment. Burnup, reactor type and age determination can be performed using any combination of isotopes that meet the requirements. The sections below detail the mathematical development of each of these monitors.

A. BURNUP

The nuclear industry has been interested in fuel burnup for decades and has developed accurate methods to measure burnup [13]. Burnup is the amount of energy produced per kg of fuel. At nuclear power plants, initially the fissions from ^{235}U and ^{238}U provide the source of neutrons and power in the reactor. As the reactor burns the fuel, ^{236}U, ^{239}Pu, ^{240}Pu and several other minor fissionable actinides are created and fissioned which then contribute to the neutron flux. For economic reasons, nuclear power plants want to increase the fuel burnup as high as possible. After the fuel is discharged from the reactor, the burnup can be measured indirectly by measuring a burnup monitor. Coupled with mass spectrometry, it has been shown that burnup can be measured within a one percent accuracy [14].

A burnup monitor can be any fission product that is created directly proportional to the burnup. For the forensics problem a more stringent restriction exists because the reactor type is unknown. Therefore, we need burnup monitors that are produced at the same rate regardless of the reactor type. The neutron energy spectrum and fissioning isotopes are primarily what changes from one reactor type to another. So for our purpose a burnup monitor is any nuclide whose fission yield is constant as a function of the neutron flux energy and fissioning isotopes. By identifying a burnup monitor that has a constant yield across reactor types, the burnup may be determined without knowing additional information about the fuel. To avoid the effects of decay, burnup monitors that have half-lives longer than 200 years should be chosen. Another concern is transmutation of the burnup monitor, a burnup monitor needs to have a small radiative capture cross section to avoid the destruction of the burnup monitor as it is produced. Having a large radiative capture cross section introduces reactor-dependent terms in the burnup calculation. To find the burnup without knowing other reactor parameters, it is necessary to avoid these terms.

The production of an isotope in a reactor is usually dependent upon more than its direct fission yield. Isotopes are also produced by the decay of other fission products. An isotope's decay chain can have a large impact on its production if its cumulative yield is significantly higher than its direct yield. This requires all of the fission products in a burnup monitor's decay chain to exhibit the same properties as the burnup monitor. The exception to this would be in the case where the fission products individual yield is so

small that it doesn't significantly effect the production of the burnup monitor. All of the isotopes in the decay chain must have a half-life shorter than a few days to avoid delayed production of the burnup monitor after the spent fuel is discharged. There are a few isotopes that are shielded by a stable isotope and thus has no decay chain. If a shielded isotope meets the requirements of a burnup monitor, then the complications of a decay chain will not apply.

After a burnup monitor has been identified, the fuel burnup can be calculated by starting with the reaction rate formula:

$$RR_f = N^{U235}(t)\int_0^\infty \left[\sigma_f^{U235}(E)\phi(E,t)dE + N^{Pu239}(t)\int_0^\infty \sigma_f^{Pu239}(E)\phi(E,t)dE + \ldots \right] \tag{2.1}$$

where RR_f is the fission reaction rate. The terms not shown are the integrals of all the remaining fissioning isotopes: ^{238}U, ^{240}Pu and ^{241}Pu. We will define one-group average fluxes and cross-sections as:

$$\bar{\phi}(t) = \int_0^\infty \phi(E,t)dE \tag{2.2}$$

and

$$\bar{\sigma}_f^{U235} = \frac{\int_0^\infty \sigma_f^{U235}(E)\phi(E,t)dE}{\int_0^\infty \phi(E,t)dE}. \tag{2.3}$$

By multiplying Eqs. 2.2 and 2.3, we get:

$$\bar{\phi}(t)\bar{\sigma}_f^{U235} = \int_0^\infty \sigma_f^{U235}(E)\phi(E,t)dE. \tag{2.4}$$

A similar term can be produced for each of the fissioning isotopes. These can be substituted into Eq. 2.1 to get:

$$RR_f = N^{U235}(t)\bar{\sigma}_f^{U235}\bar{\phi}(t) + N^{Pu239}(t)\bar{\sigma}_f^{Pu239}\bar{\phi}(t) + \ldots. \tag{2.5}$$

The power density in the reactor is given by:

$$P'''(t) = E_R RR_f, \tag{2.6}$$

where E_R is the average recoverable energy per fission. We will assume that the average energy per fission is the same for all fissions. The power density then becomes:

$$P'''(t) = E_R\left(N^{U235}(t)\bar{\sigma}_f^{U235}\bar{\phi}(t) + N^{Pu239}(t)\bar{\sigma}_f^{Pu239}\bar{\phi}(t) + \ldots\right). \tag{2.7}$$

The specific power (or watts per gram of the fuel) is:

$$P_S(t) = P'''(t) \Big/ \rho_o^U, \tag{2.8}$$

where ρ_o^U is the initial density of uranium in the fuel. Substituting Eq. 2.7 into Eq 2.8

yields:

$$P_S(t) = \frac{E_R}{\rho_o^U}\left(N^{U235}(t)\bar{\sigma}_f^{U235}\bar{\phi}(t) + N^{Pu239}(t)\bar{\sigma}_f^{Pu239}\bar{\phi}(t) + \ldots \right). \tag{2.9}$$

The burnup is simply the integral of the specific power over time:

$$BU(T) = \int_0^T P_S(t)dt \tag{2.10}$$

or

$$BU(T) = \frac{E_R}{\rho_o^U}\int_0^T \left(N^{U235}(t)\bar{\sigma}_f^{U235}\bar{\phi}(t) + N^{Pu239}(t)\bar{\sigma}_f^{Pu239}\bar{\phi}(t) + \ldots \right) dt\,. \tag{2.11}$$

Since the one-group cross sections are assumed to be time independent,

$$\bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + \ldots = \frac{\rho_o^U}{E_R}BU(T)\,. \tag{2.12}$$

We will leave this equation for a moment and use the balance equation defining the change rate of the burnup monitor *B* as:

$$\begin{aligned}\frac{dN^B}{dt} &= N^{U235}(t)\int_0^\infty \sigma_f^{U235}(E)Y_B^{U235}(E)\phi(E,t)dE \\ &\quad + N^{Pu239}(t)\int_0^\infty \sigma_f^{Pu239}(E)Y_B^{Pu239}(E)\phi(E,t)dE + \ldots.\end{aligned} \tag{2.13}$$

As stated earlier, the burnup monitor is chosen such that the yield has no energy dependence. Thus, Eq. 2.13 can be rearranged to:

$$\frac{dN^B}{dt} = N^{U235}(t)Y_B^{U235}(t)\int_0^\infty \sigma_f^{U235}(E)\phi(E,t)dE + N^{Pu239}(t)Y_B^{Pu239}(t)\int_0^\infty \sigma_f^{Pu239}(E)\phi(E,t)dE + \ldots. \quad (2.14)$$

At this point, we will also assume the cumulative yield for the burnup monitor is the same for all fission sources. Thus, Eq. 2.14 becomes:

$$\frac{dN^B}{dt} = Y_B\left[N^{U235}(t)\bar{\sigma}_f^{U235}\bar{\phi}(t) + N^{Pu239}(t)\bar{\sigma}_f^{Pu239}\bar{\phi}(t) + \ldots\right]. \quad (2.15)$$

This differential equation can be converted to an integral equation as follows:

$$\int_0^{N^B(T)} dN^B = Y_B\left[\bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + \ldots\right], \quad (2.16)$$

where $N^X(T)$ is the atom density of isotope *X* at the time of measurement. Note this assumes that initially the concentration of the burnup monitor *B* in the fuel is zero. The left hand integral thus simply becomes:

$$N^B(T) = Y_B\left[\bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + \ldots\right]. \quad (2.17)$$

Substituting Eq. 2.12 into Eq. 2.17 yields:

$$N^B(T) = \frac{\rho_o^U}{E_R}Y_B BU(T). \quad (2.18)$$

By dividing equation 2.18 by the initial uranium atom density, we acquire:

$$\frac{N^B(T)}{N_o^U} = \frac{\rho_o^U}{E_R N_o^U} Y_B BU(T) . \tag{2.19}$$

The mass density can be expressed in terms of an atom density to acquire:

$$\frac{N^B(T)}{N_o^U} = \frac{M_U}{E_R N_a} Y_B BU(T) . \tag{2.20}$$

Solving for burnup we get:

$$BU(T) = \frac{N^B(T) N_a E_R}{N_o^U M_U Y_B} . \tag{2.21}$$

The uranium atom density and uranium mass changes with time as the fuel is burned. Mass spectrometry generally requires measurements to be made as isotopic ratios. So we will modify equation 2.21 to reflect the burnup monitor being measured with respect to the ^{238}U content at the time of measurement,

$$BU(T) = \left[\frac{N^B(T)}{N^{U\,238}(T)} \right] \left[\frac{N^{U\,238}(T)}{N_o^U} \right] \frac{N_a E_R}{Y_B} . \tag{2.22}$$

The quantity $\frac{N^B(T)}{N^{U\,238}(T)}$ can be measured directly and N_a, E_R and Y_B are known constants. The term $\frac{N^{U\,238}(T)}{N_o^U}$ is not known and can not be measured directly; however, it is similar to the quantity $\frac{N^{U\,238}(0)}{N_o^U}$. This term must be accurately determined from an iteration with burnup.

The initial atom density of uranium is equal to the sum of the measured uranium

densities and all of the uranium reactions that occurred during irradiation. The reactions include fission and neutron capture which leads to:

$$N_o^U = N^{U235}(T) + N^{U238}(T) + \bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_\gamma^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{U238}\int_0^T N^{U238}(t)\bar{\phi}(t)dt + \bar{\sigma}_\gamma^{U238}\int_0^T N^{U238}(t)\bar{\phi}(t)dt \quad . \tag{2.23}$$

The radiative capture terms are essentially equal to the products in the capture chain and the fissions of those products. For example:

$$\bar{\sigma}_\gamma^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t) = N^{U236}(T) + \bar{\sigma}_f^{U236}\int_0^T N^{U236}(t)\bar{\phi}(t)dt \, , \tag{2.24}$$

and

$$\bar{\sigma}_\gamma^{U238}\int_0^T N^{U238}(t)\bar{\phi}(t) = N^{Pu239}(T) + \bar{\sigma}_f^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + N^{Pu240}(T) + \bar{\sigma}_f^{Pu240}\int_0^T N^{Pu240}(t)\bar{\phi}(t)dt + \ldots \quad . \tag{2.25}$$

Note that the delayed production of ^{239}Pu from the decay of ^{239}U and ^{239}Np is ignored. Equations 2.24 and 2.25 are substituted into Eq. 2.23 to get:

$$N_o^U = N^{U235}(T) + N^{U238}(T) + \left[N^{U236}(T) + N^{Pu239}(T) + N^{Pu240}(T) + \ldots \right] + \left[\bar{\sigma}_f^{U235} \int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{U238} \int_0^T N^{U238}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{U236} \int_0^T N^{U236}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239} \int_0^T N^{Pu239}(t)\bar{\phi}(t)dt \ldots \right] \quad (2.26)$$

The last term is the same term in Eq. 2.26 is the same term we found in Eq. 2.12. By substitution, we can avoid the integrals and acquire:

$$N_o^U = N^{U235}(T) + N^{U238}(T) + \left[N^{U236}(T) + N^{Pu239}(T) + N^{Pu240}(T) + \ldots \right] + \frac{\rho_o^U}{E_R} BU(T) \quad (2.27)$$

The uranium density can be changed to be in terms atom density again and the uranium terms combined to find:

$$N_o^U = N^U(T) + \left[N^{Pu239}(T) + N^{Pu240}(T) + \ldots \right] + \frac{N_o^U M_o^U}{N_a E_R} BU(T) . \quad (2.28)$$

We will divide Eq. 2.28 by N^{U238} to acquire:

$$\frac{N_o^U}{N^{U238}} = \frac{N^U(T)}{N^{U238}} + \left[\frac{N^{Pu239}(T)}{N^{U238}} + \frac{N^{Pu240}(T)}{N^{U238}} + \ldots \right] + \frac{N_o^U}{N^{U238}} \frac{M_o^U}{N_a E_R} BU(T) \quad (2.29)$$

We can solve Eq. 2.28 for the ratio of initial uranium atom density to the measured ^{238}U

atom density to get:

$$\left[\frac{N_o^U}{N^{U238}(T)}\right]=\frac{\left[\frac{N^U(T)}{N^{U238}(T)}\right]+\left[\frac{N^{Pu239}(T)}{N^{U238}(T)}\right]+\left[\frac{N^{Pu240}(T)}{N^{U238}(T)}\right]+\cdots}{1-\frac{M_o^U}{N_a E_R}BU(T)}. \qquad (2.30)$$

This is still in terms of burnup which is unknown. We will initially guess that *BU*(T) = 0 and then solve Eq. 2.30 to get $\frac{N_o^U}{N^{U238}(T)}$. We will then solve Eq. 2.22 for *BU*(T). We will then iterate between Eq. 2.30 and 2.22 until they converge. The burnup equation is not very sensitive to changes in the initial uranium atom density, allowing it to converge after only a few iterations. On average the error in calculating N_o^U was around 0.1%, (see Table I) which is sufficient accuracy for this work. The term $\frac{N_o^U}{N^{U238}(T)}$ will be used again by our enrichment calculation, and the enrichment is more sensitive to this term. Thus, while an error of 0.1% is insignificant for the purposes of estimating burnup, it is more important in later calculations. Thus, iterations will always be performed to reduce the error as small as possible.

TABLE I

Initial Uranium Atom Density per ^{238}U Atom Density Calculation After Using an Iteration Scheme with Burnup

Reactor Type	Burnup (MWd/MT)	Exact $\frac{N_o^U}{N^{U\,238}(T)}$	Calculated $\frac{N_o^U}{N^{U\,238}(T)}$	Percent Error
PWR	40,000	.9375	.9386	0.109
BWR	40,000	.9378	.9387	0.097
CANDU	15,000	.9769	.9770	0.007
LMFBR	70,000	.8291	.8284	0.076

Methods, such as show above, and others have been developed previously to determine fuel burnup from the measurement of fission product isotopes and is used at reactors and reprocessing facilities [15]. Two common burnup monitors that are used in industry are ^{148}Nd and ^{137}Cs. ^{137}Cs has a 30 year half-life and can only be used on fresh fuels or when the discharge date is known. By measuring ^{137}Cs, it has been shown that the burnup can be determine within a 5% accuracy [16]. Mass spectrometry of ^{148}Nd has been used to determine burnup with a higher accuracy [17], but this rarely occurs because of the difficulties and costs associated with destructive assay of spent fuel.

B. ENRICHMENT

Determining the initial enrichment after the spent fuel has been discharge is a new problem. Fuel is very well characterized before being used in a nuclear power plant and

there has been no need in the past to determine the initial enrichment from spent fuel. If no other fissionable isotopes are present while the fuel is in the reactor, then the enrichment would change linearly with burnup. In reality, several other fissionable isotopes are produced and burned, thus complicating the determination of the initial enrichment.

Figure 4 shows how the other fissionable isotopes will affect the ^{235}U enrichment as the fuel is burned. The solid line is an accurate calculation of the ^{235}U enrichment in a PWR reactor burned to 40,000 MWd/MT (calculated by ORIGEN). If this represented fuel that was used in an RDD, the present day enrichment would be measured at 0.6%. By considering ^{235}U as the only fissionable isotope the initial enrichment would be back calculated to be just below 4.8%, which has an error of 49%. The error will become larger as the fuel burnup increases.

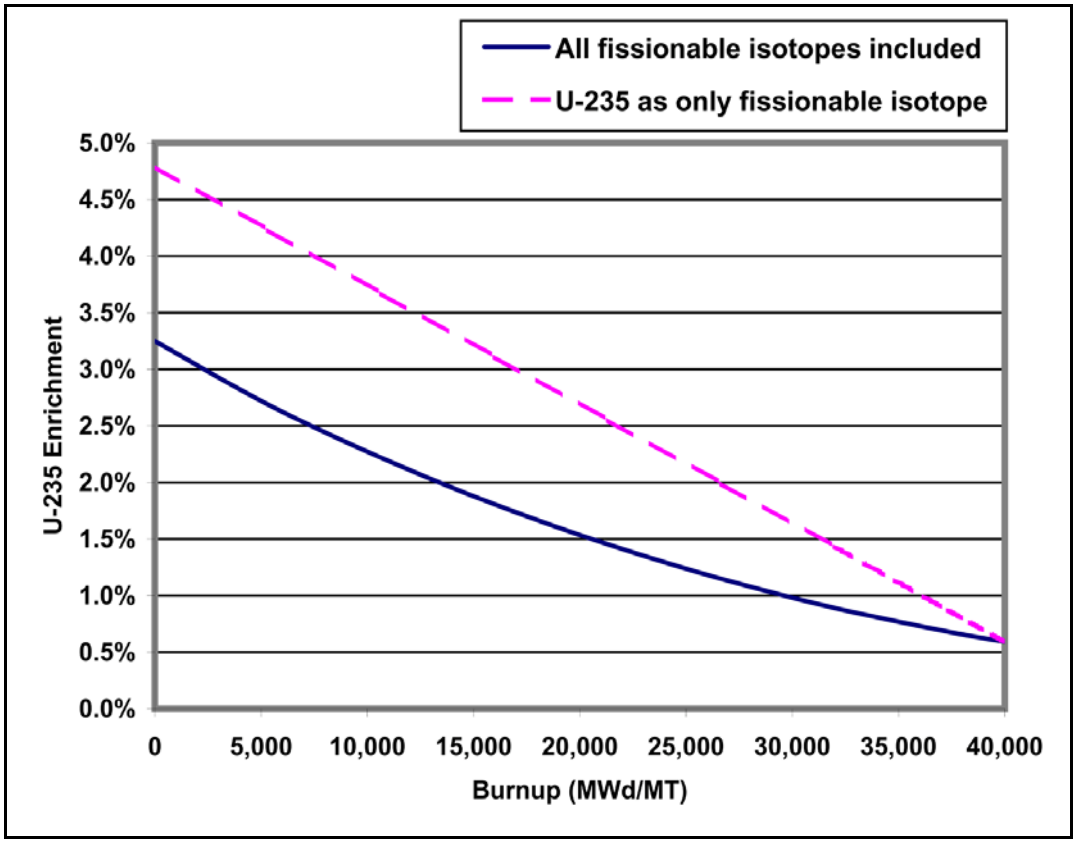

Fig. 4. Comparison of enrichment versus burnup, between a theoretical fuel rod that only fissions ^{235}U and a true fuel rod.

There are several reasonable assumptions made when calculating the enrichment. We assume that:

1. The only isotopes that fission are ^{235}U, ^{238}U, ^{239}Pu, ^{240}Pu and ^{241}Pu, as shown in Fig. 5.
2. There is no production of ^{235}U and ^{238}U. All one group cross sections are constant with time.
3. ^{239}Np and ^{239}U decay instantaneously to ^{239}Pu.
4. The decay of all fissionable isotopes are neglected.

The only isotope that has a short enough half-life to significantly effect our calculation is ^{241}Pu. ^{241}Pu has a 14.1 year half-life, which will cause our calculation for initial enrichment to be slightly high. The higher the burnup and the longer the fuel has been discharged, the larger the effect will be on our results.

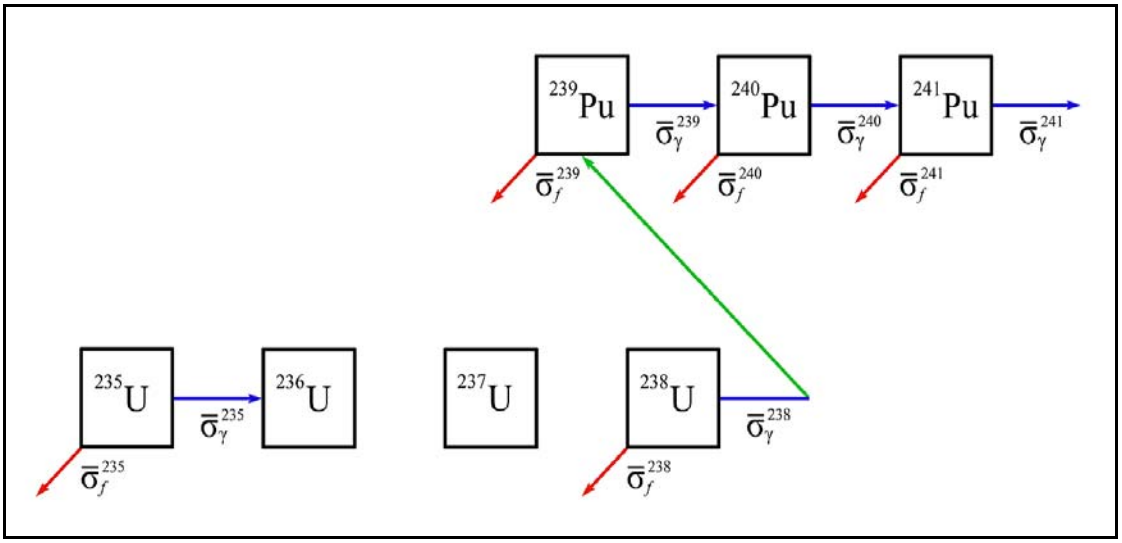

Fig. 5. The reactions that are considered in the enrichment calculation.

To find the enrichment, we will start with the initial density of ^{235}U atoms as:

$$N_o^{U235} = N^{U235}(T) + \bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt + \bar{\sigma}_\gamma^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt\,. \qquad (2.31)$$

Again, we use Eq. 2.24 for the radiative capture of ^{235}U and ignore the fission of ^{236}U to get:

$$N_o^{U235} = N^{U235}(T) + N^{U236}(T) + \bar{\sigma}_f^{U235}\int_0^T N^{U235}(t)\bar{\phi}(t)dt\,. \qquad (2.32)$$

We will substitute in Eq. 2.12 for the ^{235}U fission integral as follows:

$$N_o^{U235} = N^{U235}(T) + N^{U236}(T) + \frac{\rho_o^U}{E_R} BU(T) - \left[\bar{\sigma}_f^{U238} \int_0^T N^{U238}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239} \int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + \ldots \right] \tag{2.33}$$

Now we divide Eq. 2.33 by the initial uranium atom density:

$$\frac{N_o^{U235}}{N_o^U} = e_o = \frac{N^{U235}(T)}{N_o^U} + \frac{N^{U236}(T)}{N_o^U} + \frac{\rho_o^U}{E_R N_o^U} BU(T) - \frac{1}{N_o^U}\left[\bar{\sigma}_f^{U238} \int_0^T N^{U238}(t)\bar{\phi}(t)dt + \bar{\sigma}_f^{Pu239} \int_0^T N^{Pu239}(t)\bar{\phi}(t)dt + \ldots \right] \tag{2.34}$$

where e_o is the initial enrichment. By using the balance equation for the rate of change, a solution for each integral in Eq. 2.34 can be found. The balance equation for ^{238}U is:

$$\frac{dN^{U238}}{dt} = -N^{U238}(t)\bar{\sigma}_a^{U238}\bar{\phi}(t)\,. \tag{2.35}$$

Converting Eq. 2.35 to an integral form, yields:

$$N^{U238}(T) = N_o^{U238} - \bar{\sigma}_a^{U238} \int_0^T N^{U238}(t)\bar{\phi}(t)dt\,. \tag{2.36}$$

We can then solve for the integral:

$$\int_0^T N^{U238}(t)\bar{\phi}(t)dt = \frac{1}{\bar{\sigma}_a^{U238}}\left[N_o^{U238} - N^{U238}(T) \right]. \tag{2.37}$$

The balance equation for ^{239}Pu is:

$$\frac{dN^{Pu239}}{dt} = N^{U238}(t)\bar{\sigma}_{\gamma}^{U238}\bar{\phi}(t) - N^{Pu239}(t)\bar{\sigma}_{a}^{Pu239}\bar{\phi}(t). \tag{2.38}$$

Converting Eq. 2.38 to an integral form, yields:

$$\begin{aligned} N^{Pu239}(T) = N_o^{Pu239} &- \bar{\sigma}_{\gamma}^{U238}\int_0^T N^{U238}(t)\bar{\phi}(t)dt \\ &-\bar{\sigma}_{a}^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt \end{aligned} \tag{2.39}$$

Solving for the ^{239}Pu integral:

$$\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt = \frac{1}{\bar{\sigma}_{a}^{Pu239}}\left[N_o^{Pu239} - N^{Pu239}(T) + F^{238} \right], \tag{2.40}$$

where,

$$F^{238} \equiv \frac{\bar{\sigma}_{\gamma}^{U238}}{\bar{\sigma}_{a}^{U238}}\left[N_o^{U238} - N^{U238}(T) \right]. \tag{2.41}$$

The balance equation for ^{240}Pu is:

$$\frac{dN^{Pu240}}{dt} = N^{U239}(t)\bar{\sigma}_{\gamma}^{U239}\bar{\phi}(t) - N^{Pu240}(t)\bar{\sigma}_{a}^{Pu240}\bar{\phi}(t). \tag{2.42}$$

Converting Eq. 2.42 to an integral form, yields:

$$\begin{aligned} N^{Pu240}(T) = N_o^{Pu240} &- \bar{\sigma}_{\gamma}^{Pu239}\int_0^T N^{Pu239}(t)\bar{\phi}(t)dt \\ &-\bar{\sigma}_{a}^{Pu240}\int_0^T N^{Pu240}(t)\bar{\phi}(t)dt \end{aligned} \tag{2.43}$$

Solving for the ^{240}Pu integral:

$$\int_0^T N^{Pu240}(t)\bar{\phi}(t)dt = \frac{1}{\bar{\sigma}_a^{Pu240}}\left[N_o^{Pu240} - N^{Pu240}(T) + F^{239} \right], \tag{2.44}$$

where,

$$F^{239} \equiv \frac{\bar{\sigma}_\gamma^{Pu239}}{\bar{\sigma}_a^{Pu239}}\left[N_o^{Pu239} - N^{Pu239}(T) + F^{238} \right]. \tag{2.45}$$

The balance equation for ^{241}Pu is:

$$\frac{dN^{Pu241}}{dt} = N^{U240}(t)\bar{\sigma}_\gamma^{U240}\bar{\phi}(t) - N^{Pu241}(t)\bar{\sigma}_a^{Pu241}\bar{\phi}(t). \tag{2.46}$$

Converting Eq. 2.46 to an integral form, yields:

$$\begin{aligned} N^{Pu241}(T) = N_o^{Pu241} &- \bar{\sigma}_\gamma^{Pu240}\int_0^T N^{Pu240}(t)\bar{\phi}(t)dt \\ &- \bar{\sigma}_a^{Pu241}\int_0^T N^{Pu241}(t)\bar{\phi}(t)dt \end{aligned} \tag{2.47}$$

Solving for the ^{241}Pu integral:

$$\int_0^T N^{Pu241}(t)\bar{\phi}(t)dt = \frac{1}{\bar{\sigma}_a^{Pu241}}\left[N_o^{Pu241} - N^{Pu241}(T) + F^{240} \right], \tag{2.48}$$

where

$$F^{240} \equiv \frac{\bar{\sigma}_\gamma^{Pu240}}{\bar{\sigma}_a^{Pu240}}\left[N_o^{Pu240} - N^{Pu240}(T) + F^{239} \right]. \tag{2.49}$$

Substituting Eq. 2.37, 2.40, 2.44 and 2.48 into Eq. 2.34, yields:

$$e_o = \frac{N^{U235}(T)}{N_o^U} + \frac{N^{U236}(T)}{N_o^U} + \frac{\rho_o^U}{E_R N_o^U} BU(T)$$
$$- \frac{1}{N_o^U}\left[\frac{\bar{\sigma}_f^{U238}}{\bar{\sigma}_a^{U238}} \left[N_o^{U238} - N^{U238}(T) \right] \right.$$
$$+ \frac{\bar{\sigma}_f^{Pu239}}{\bar{\sigma}_a^{Pu239}} \left[N_o^{Pu239} - N^{Pu239}(T) + F^{238} \right] \qquad (2.50)$$
$$+ \frac{\bar{\sigma}_f^{Pu240}}{\bar{\sigma}_a^{Pu240}} \left[N_o^{Pu240} - N^{Pu240}(T) + F^{239} \right]$$
$$\left. + \frac{\bar{\sigma}_f^{Pu241}}{\bar{\sigma}_a^{Pu241}} \left[N_o^{Pu241} - N^{Pu241}(T) + F^{240} \right] \right]$$

It is assumed that initially there is no plutonium in the fuel or:

$$N_o^{Pu239} = N_o^{Pu240} = N_o^{Pu241} = 0 .$$

We will also pass the $\frac{1}{N_o^U}$ through the brackets and make the following substitution:

$$\frac{N_o^{U238}}{N_o^U} = 1 - e_o ,$$

to get:

$$e_o = \frac{N^{U238}(T)}{N_o^U} \left[\frac{N^{U235}(T)}{N^{U238}(T)} + \frac{N^{U236}(T)}{N^{U238}(T)} \right]$$
$$+ \frac{M_o^U}{N_a E_R} BU(T) - G^{238} - G^{239} - G^{240} - G^{241} \qquad (2.51)$$

where

$$G^{238} \equiv \frac{\bar{\sigma}_f^{U238}}{\bar{\sigma}_a^{U238}}\left[1-e_o-\frac{N^{U238}(T)}{N_o^U}\right], \tag{2.52}$$

$$G^{239} \equiv \frac{\bar{\sigma}_f^{Pu239}}{\bar{\sigma}_a^{Pu239}}\left[-\frac{N^{Pu239}(T)}{N^{U238}(T)}\frac{N^{U238}(T)}{N_o^U}+G^{238}\right], \tag{2.53}$$

$$G^{240} \equiv \frac{\bar{\sigma}_f^{Pu240}}{\bar{\sigma}_a^{Pu240}}\left[-\frac{N^{Pu240}(T)}{N^{U238}(T)}\frac{N^{U238}(T)}{N_o^U}+G^{239}\right], \tag{2.54}$$

$$G^{241} \equiv \frac{\bar{\sigma}_f^{Pu241}}{\bar{\sigma}_a^{Pu241}}\left[-\frac{N^{Pu241}(T)}{N^{U238}(T)}\frac{N^{U238}(T)}{N_o^U}+G^{240}\right]. \tag{2.55}$$

Now we have an equation in known terms and e_o. We still have to solve explicitly for e_o, which is extremely tedious but can be done by simply algebra. Due to the length of the solution, it can be found in Appendix A.

The solution for e_o from Eq. 2.51 is sensitive to slight changes in N_o^U and has found to be somewhat unstable. The error can be as high as 15%. The solution from Eq. 2.51 is thus used as an initial calculation for the enrichment. The final enrichment result is produced through an iteration method using ORIGEN (see Fig. 6). After finding the burnup and enrichment from Eq. 2.22 and 2.51, the results are then used to build a reactor core model for each reactor type being considered. Each model is then simulated by ORIGEN. After each run the ^{235}U content is compared with the measured value. The initial enrichment is then adjusted by using

$$e_{new} = e_o \left(\frac{{}^{U235}U_{measured} - {}^{U235}U_{ORIGEN}}{2\,{}^{U235}U_{measured}} + 1 \right) \tag{2.56}$$

until the output of ORIGEN matches the measured value of ^{235}U. Then the same process is repeated for the burnup monitors. The whole cycle is then repeated twice. This method improved the enrichment calculations to an accuracy of 2% or better. Table II shows improvement made by performing the iteration process. This calculation used data which will be described in more detail in Chapter IV.

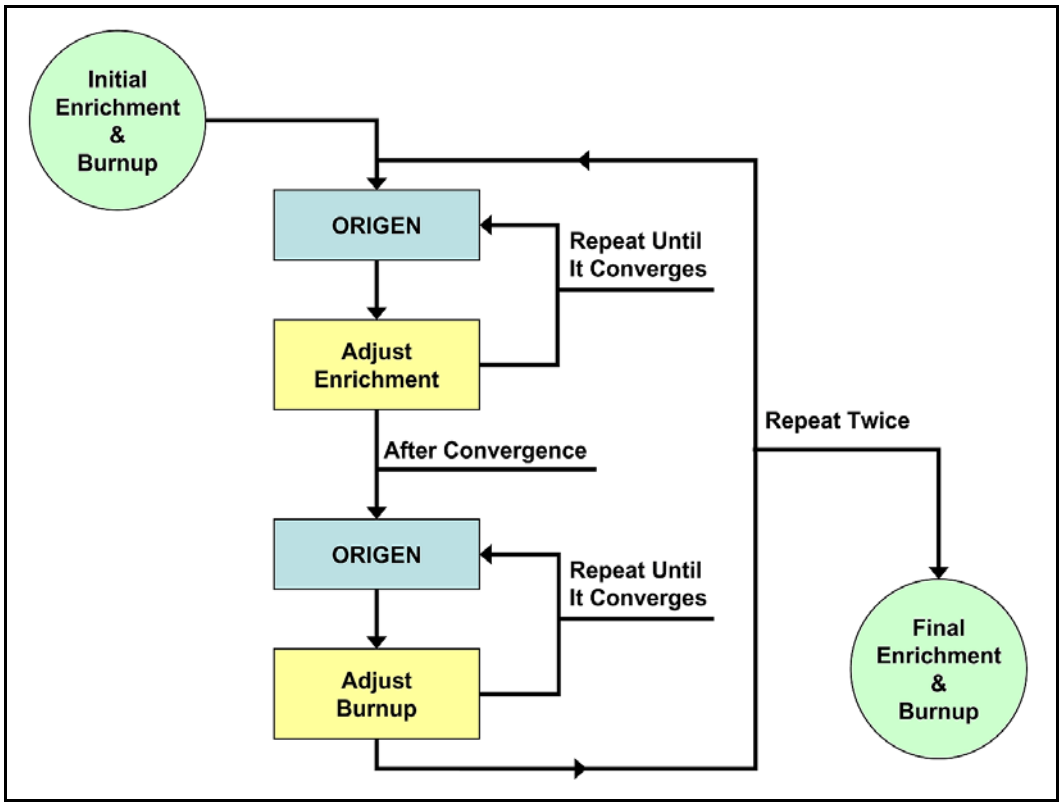

Fig. 6. Flow of forward model iteration scheme for improving the enrichment calculation.

TABLE II

Enrichment Calculations Before and After Using an Iteration Scheme with ORIGEN

	Reported	Initial Calculation		After Iteration	
Mihama-3	Enrichment (a/o)	Enrichment (a/o)	Percent Error	Enrichment (a/o)	Percent Error
Sample 1	3.25	2.93	**9.94**	3.22	**1.08**
Sample 2	3.25	2.98	**8.31**	3.27	**0.62**
Sample 3	3.24	2.81	**13.27**	3.20	**1.33**
Sample 4	3.24	2.91	**10.19**	3.27	**0.93**
Sample 5	3.24	2.77	**14.51**	3.20	**1.23**
Sample 6	3.25	3.16	**2.77**	3.21	**1.23**
Sample 7	3.25	3.32	**2.15**	3.29	**1.23**
Sample 8	3.25	3.36	**3.38**	3.22	**0.92**
Sample 9	3.25	3.38	**4.00**	3.21	**1.23**

C. REACTOR TYPE

Work has been done previously to verify the reactor type of spent fuel for reprocessing facilities [15]. This was done by taking samples of the gases that escaped from the reprocessing process. A burnup monitor that met similar requirements as our burnup monitor was chosen. Due to being restricted to gases, they were very limited on possible isotopes and the accuracy wasn't as good as the burnup monitors we chose. They found production of $^{132}Xe/^{134}Xe$ ratio varied by as much as 25% from the different reactor types. By measuring this ratio, they were able to determine from what type of reactor the fuel originated.

Determining the reactor type is critical to the success of this project. Our ability to predict the reactor type relies heavily on the accuracy of the one-group cross sections and yield values. As discussed in Chapter I, the cross sections are created from MCNP reactor core models. To differentiate the reactor types from each other, isotopes with cross sections and yields that change significantly from reactor type to reactor type are needed. To avoid the complication of decay, stable or long-lived isotopes are needed. The method for determining reactor type depends upon the accuracy of ORIGEN, and as the decay chain becomes more complicated the accuracy is diminished. ORIGEN also has a more difficult time tallying isotopes in trace amounts with a high degree of accuracy. To avoid these issues, it is best if the chosen isotopes have a simple decay chain and are produced in quantities above 0.01 moles per fuel assembly.

The mass spectroscopy measurements are generally performed as isotopic ratios. ^{238}U is the most abundant isotope by many orders of magnitude in spent fuel. By measuring a reactor type monitor with respect to ^{238}U, slight changes of the quantity produced of the monitor will make little difference in the measured value. To avoid this and improve the accuracy, reactor type monitors are measured with respect to something that is produced in similar quantities. This will ensure that slight changes in reactor type monitor quantities will be easily evaluated. This is especial true when the burnup is low. Figure 7 shows the $^{132}Ba/^{238}U$ isotopic ratio versus burnup. At 10,000 MWd/MT our ability to distinguish between reactor types is low. By measuring ^{132}Ba with respect to ^{148}Nd it is much easier to distinguish between the reactor types at a lower burnup (see Fig. 8).

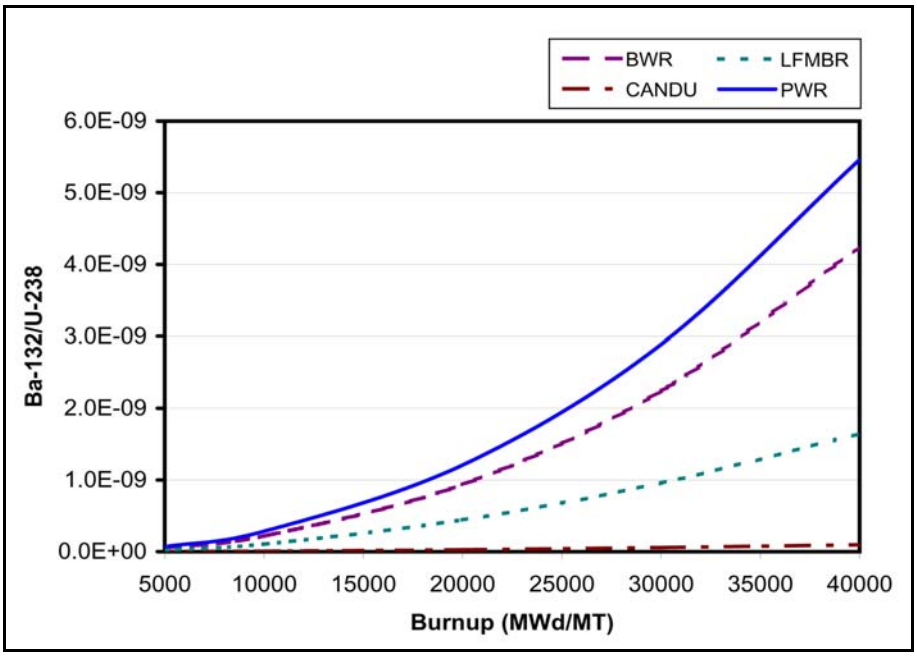

Fig. 7. The production of ^{132}Ba with respect to ^{238}U for different reactor types.

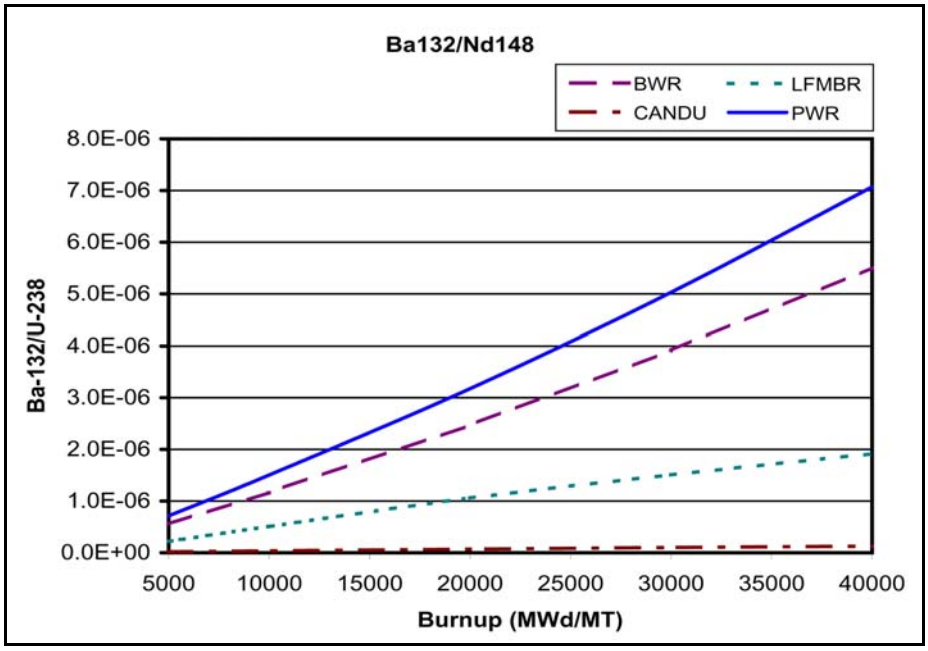

Fig. 8. The production of ^{132}Ba with respect to ^{148}Nd for different reactor types.

In the case where others attempted to validate the reactor type of spent fuel by the gases released from reprocessing, they had difficulty distinguishing between PWR and BWR reactors. We have found a series of isotopes that allows us to differentiate between PWR and BWR reactors much more accurately. They are: ^{132}Ba, ^{98}Tc, ^{115}In, ^{72}Ge and ^{115}Sn, and will be discussed in detail later.

The method for finding the correct reactor type is simple and direct. A model for each reactor type to be considered is created and run by ORIGEN. The isotopics are then compared with the measurements and the relative error is calculated. Each reactor type is then ranked according to the lowest average error. To minimize the effect of a single isotope being miscalculated by ORIGEN, as many isotope ratios as possible are used.

D. AGE

Once the burnup, initial enrichment and reactor type are known, finding the time since the discharge date is relatively straightforward. Once again the method requires ORIGEN and the accuracy of the solution depends upon the accuracy of ORIGEN. Age monitors should be chosen by their half-life and the ability of ORIGEN to accurately calculated them. To get the best accuracy, an isotope with a half-life close to the age of the spent fuel is used to make the final calculation. Age monitors that have a half-life between one and thirty years should be chosen, since the oldest spent fuel is less than 60 years old.

At this point, five reactor types that match the reactor type monitors the best are chosen. For these five reactor types an ORIGEN model is built for each. ORIGEN runs the models and only allows the spent fuel to decay for approximately 30 days. This allows the short-lived isotopes to completely decay into the more stable isotopes. The isotopic quantity is collected for each of the age monitors from the ORIGEN output. This value is then set as the initial quantity, N_o^C (shown in Fig. 9), which is the quantity of isotope *C* at the time of discharge.

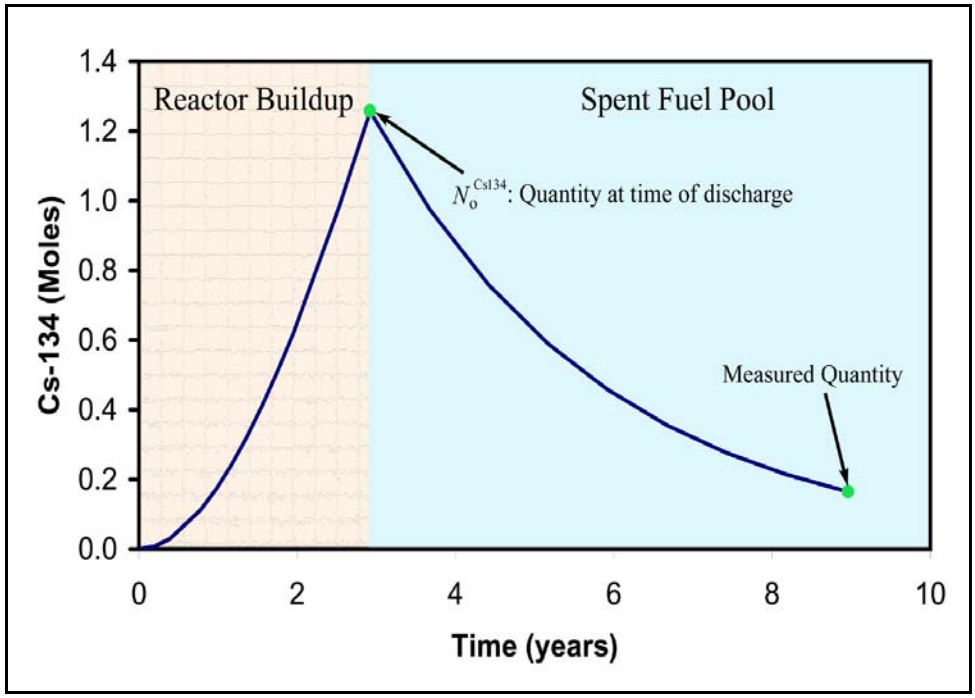

Fig. 9. Buildup of ^{134}Cs in a PWR reactor and its decay in the spent fuel pool.

We first start with the standard decay equation:

$$N^C(T) = N_o^C \, \mathrm{e}^{(-\lambda T)}, \tag{2.57}$$

where N^C is the concentration of the age monitor *C* and λ is the decay constant for isotope *C*, and is given by:

$$\lambda = \frac{\ln(2)}{T_{1/2}}. \tag{2.58}$$

By substituting equation 2.58 in 2.57, and solving for *T* we get the following:

$$T = -\frac{T_{1/2}}{\ln(2)} \ln\left(\frac{N^C(T)}{N_o^C}\right), \tag{2.59}$$

where *T* is the spent fuel age. After rearrangement into measurable terms we acquire:

$$T = -\frac{T_{1/2}}{\ln(2)} \ln\left(\left[\frac{N^C(T)}{N^{U238}}\right] \Big/ \left[\frac{N_o^C}{N^{U238}}\right]\right). \tag{2.60}$$

The age is then calculated for each age monitor and averaged. Not all of the age calculations from each age monitor are going to be accurate. The closer the half-life of the age monitor is to the actual age the more accurate it will be. The average age of all the age monitors should be close to the actual age. Once an initial average is calculated the two age monitors that have the closest half-life to the initial calculated average age are selected. Then a new average is calculated with just these two age monitors. This is then reported as the age, or the time passed since the spent fuel was discharged from the reactor. The age determination relies upon the correct calculation of burnup, initial

enrichment, and reactor type. Therefore, if there are large errors in any of these calculations, it will affect the age prediction.

If ^{241}Pu is used as an age monitor, problems may arise with mixed oxide (MOX) fuels. This may cause the initial amount of ^{241}Pu, or N_o^{Pu241} to be more than the present-day measured amount. The only solution to Eq. 2.60 in this case is for *T* to be less than zero. In this event the ^{241}Pu should be removed as an age monitor and not used. Another factor that will affect the age prediction is the shutdown time. Fuel is usually used in cycles and in between cycles the fuel is temporarily stored onsite. The storage time between cycles is referred to as shutdown time. During the shutdown the age monitors will decay and cause the ORIGEN forward model to incorrectly calculate the atom density of the age monitor at the time of discharge.

E. SHUTDOWN TIME

The shutdown time is when a fuel assembly is not being irradiated during its fuel cycle. Knowing the approximate shutdown time is another important piece of information that would aid an investigator. Once the shutdown time is known it can be used to verify the suspect fuel assembly against its power history.

The shutdown monitor is produce mostly from the decay of other fission products. Its individual fission product yield is either zero or small compared to its accumulated yield.

The shutdown monitor must be stable or long-lived to be effective. The production of the shutdown monitor ^{147}Sm can be seen in Fig. 10. The key that gives us the ability to find the shutdown time is the shutdown monitor's immediate precursor in the decay chain. As shown in Fig. 10., the shutdown monitor's precursor is ^{147}Pm. The precursor must have several characteristics. Its half-life needs to range from a few months to approximately five years, and its one group radiative capture cross section needs to be large.

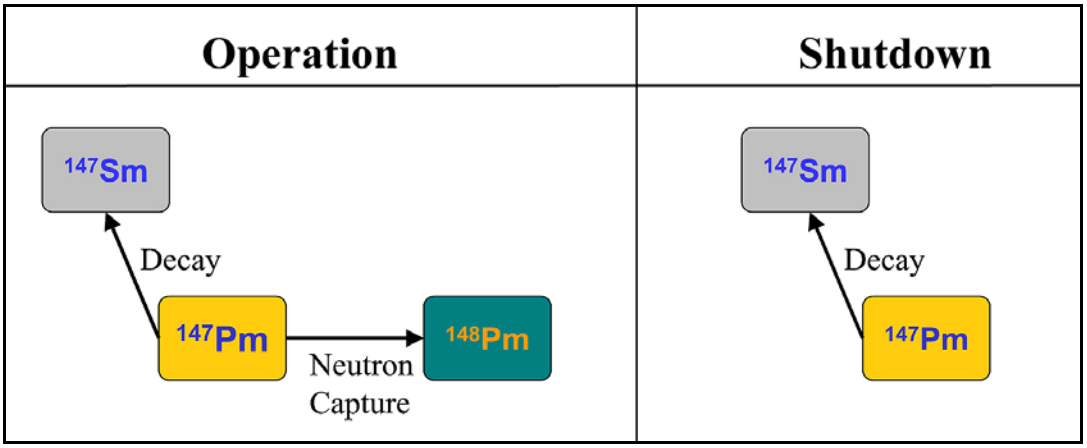

Fig. 10. Comparison of the production of shutdown monitor ^{147}Sm during and after being irradiated.

The driving concept behind the shutdown monitor is transmutation of the precursor. As the precursor ^{147}Pm is produced in the reactor, part of it is transmuted into ^{148}Pm and part of it decays to ^{147}Sm. During shutdown, the fuel is no longer being irradiated, so all of the ^{147}Pm begins to decay into ^{147}Sm. This will boost the total production of ^{147}Sm in the fuel relative to the shutdown time. In Fig. 11, the production of the shutdown monitor ^{147}Sm is shown in two different fuel cycles. Power history A is a straight burnup of the

fuel without any shutdown time. In power history B, the fuel experiences two separate shutdowns and is temporarily stored for 200 days. The increase production of the ^{147}Sm during the shutdown time can easily be seen in Fig. 11.

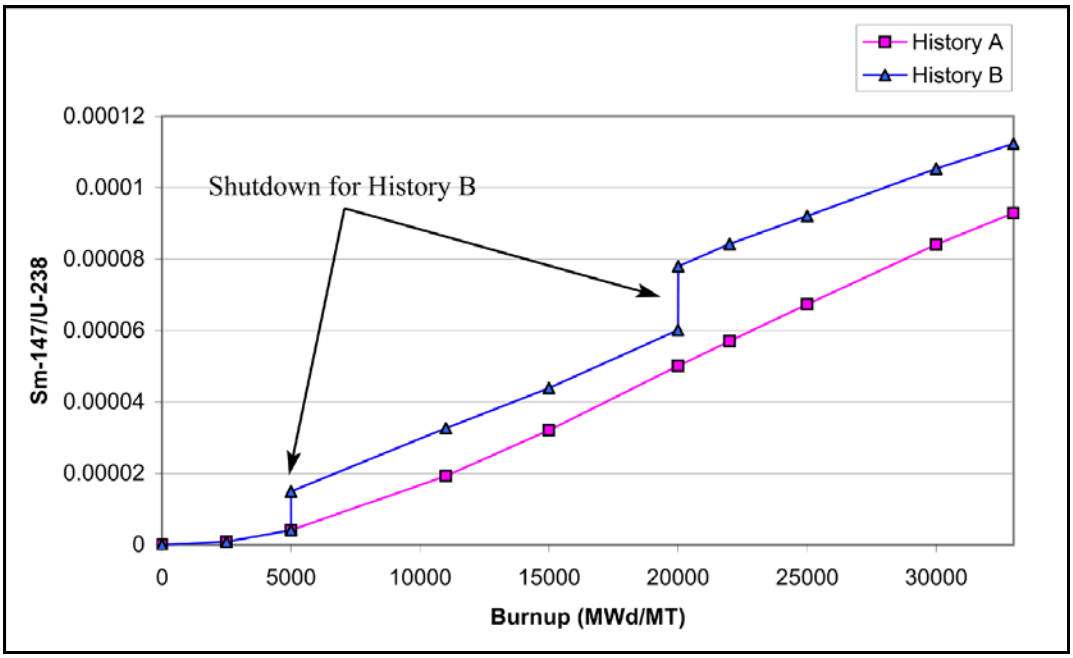

Fig. 11. Production of ^{147}Sm in a PWR with two different power histories.

The shutdown time can be found by the predicting the change in atom density of the shutdown monitor from the base density. The base density is the shutdown monitor's atom density when the fuel experiences no shutdown time. The base density in Fig. 11 is shown by power history A. As you increase the shutdown time, the atom density of the shutdown monitor will increase. To show the relationship between the shutdown time and atom density, the graph shown in Fig. 12 was created. Figure 12 shows that there is a roughly linear relationship between the shutdown time and the change in atom density.

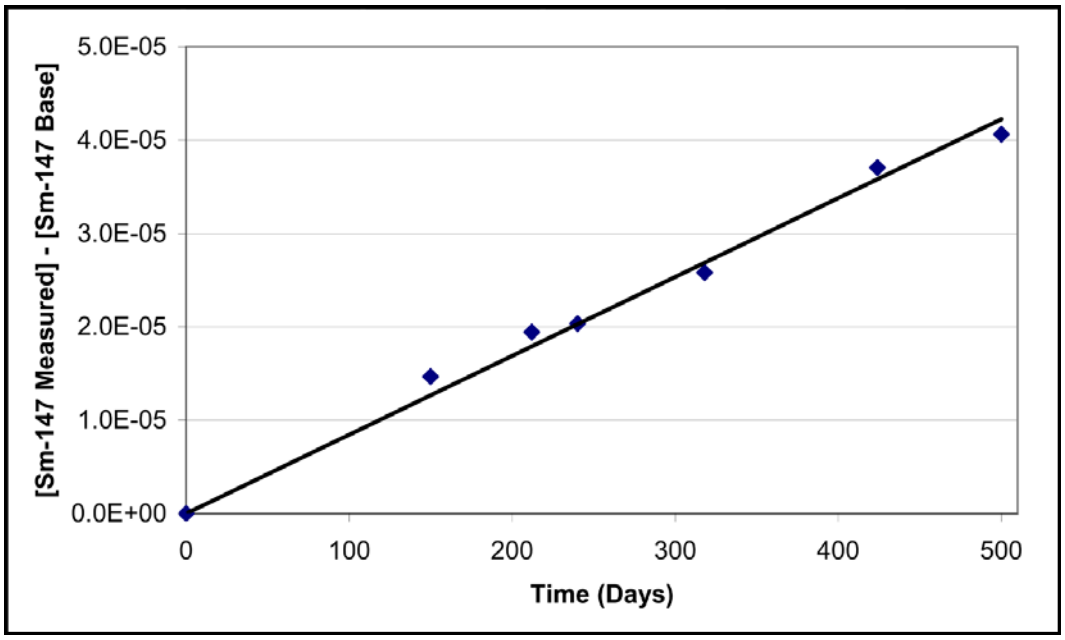

Fig. 12. The change in atom density of ^{147}Sm versus shutdown time.

By knowing the linear relationship show in Fig. 12, one could estimate the shutdown time by measuring the shutdown monitor's atom density and subtracting the base atom density predicted by a forward model (e.g. ORIGEN). This concept has been not been benchmarked by measured data due to the lack of data on known shutdown monitors. It has also not been implemented.

F. SELECTED ISOTOPES

The characteristics needed for each category are different. There are over 1500 known isotopes which require an extensive effort to find isotopes that matched the needed parameters. ORIGEN was used to run forward models of different reactor types. We studied the output and found isotopes that met our requirements. The list given here

includes the suggested nuclides to use for the forensics problem; however, it is acknowledged that this list may not be all inclusive. As additional monitor nuclides are identified, they should be studied in more detail.

1. BURNUP MONITORS

The following are the suggested burnup monitors: ^{140}Ce, ^{100}Mo, ^{98}Mo, ^{97}Mo, ^{138}Ba, ^{142}Ce and ^{148}Nd. ^{148}Nd (which is stable) is generally one of the more accurate monitors and is regularly used in the nuclear industry. Figures 13 and 14 show the ratio of two burnup monitors as a function of burnup for different reactor types. Graphs of each burnup monitor can be found in Appendix B. ^{137}Cs is not included in this list due to its 30 year half-life. Since the age will not be known prior to the burnup determination, this half-life could prove problematic. Therefore, it should only be used if no other burnup monitors are available. In the case where it is necessary to use ^{137}Cs, an iterative process between burnup and age could be performed to provide a solution.

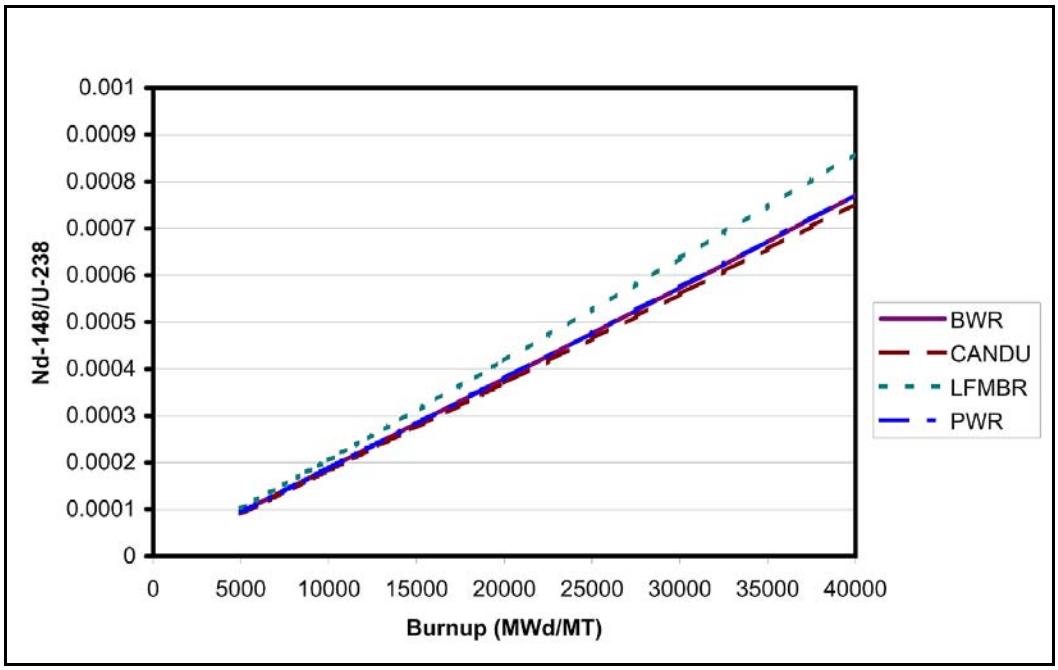

Fig. 13. Atom ratio of ^{148}Nd to ^{238}U for several reactor types.

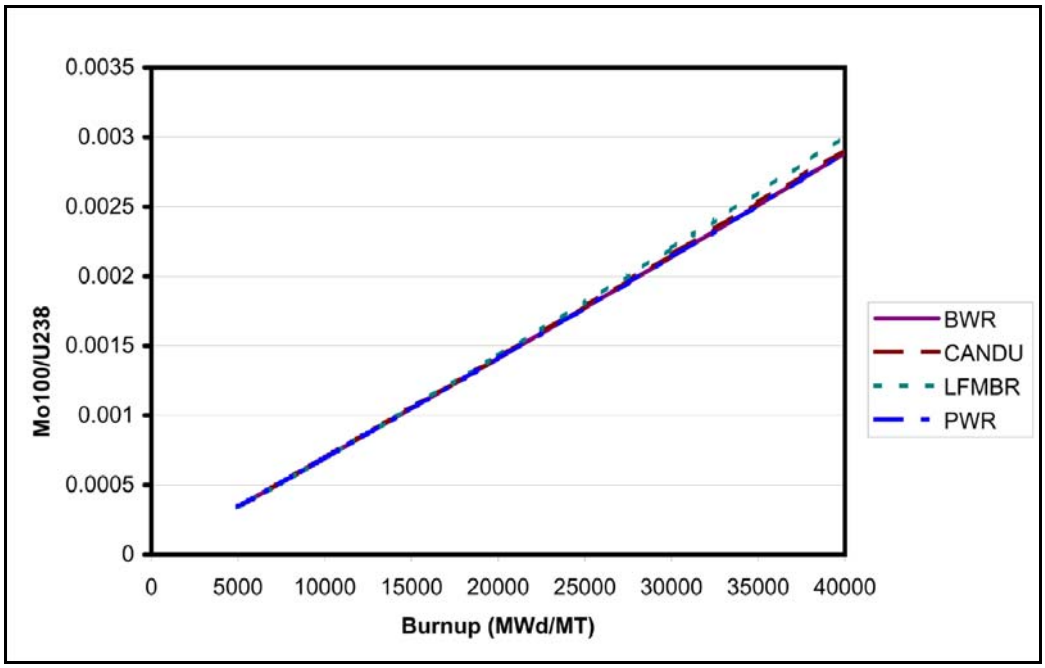

Fig. 14. Atom ratio of ^{100}Mo to ^{238}U for several reactor types.

2. ENRICHMENT MONITORS

Enrichment is unique among the monitors because there is a particular set of isotopes needed to calculate it. This set is: ^{234}U, ^{235}U, ^{236}U, ^{238}U, ^{239}Pu, ^{240}Pu and ^{241}Pu. These are all of the fissioning isotopes that are considered in the enrichment calculation. For the initial calculation, it is assumed that only ^{235}U and ^{238}U are present in the fresh fuel. This is a good approximation but not strictly correct in all cases. Fresh fuel has traces of ^{234}U and U.S. origin fresh fuel contains a small fraction of ^{236}U from down blending of military materials through commercial facilities [18].

In the case of MOX fuels, plutonium is present in the fresh fuel and will cause the error in the initial enrichment calculation to be larger. This error will eliminated via the ORIGEN iteration step. However, the plutonium isotopic composition in MOX fuels varies and therefore even the ORIGEN iteration will likely have larger errors (greater than 2%).

3. REACTOR TYPE MONITORS

The following nuclides are suggested as possible reactor type monitors: ^{109}Ag, ^{153}Eu, ^{156}Gd, ^{143}Nd, ^{240}Pu, ^{108}Cd, ^{113}Cd, ^{149}Sm, ^{166}Er, ^{132}Ba, ^{98}Tc, ^{115}In, ^{72}Ge and ^{115}Sn. The discrimination capability of these monitors varies. For example, the ability to distinguish a fast reactor from a thermal reactor is easily done with almost any of the isotopes listed. The ability to distinguish between a PWR and BWR is difficult because the neutron spectrum in both reactors is very similar. Only a few of the reactor type monitors listed

can distinguish PWR from BWR. ^{132}Ba, ^{98}Tc, ^{115}In, ^{72}Ge, ^{108}Cd and ^{115}Sn are the only monitors listed that can distinguish a PWR from a BWR, with ^{132}Ba being the best (see Fig. 7). ^{113}Cd, ^{149}Sm, ^{115}In, ^{72}Ge and ^{115}Sn are particularly good for distinguishing a fast from a thermal reactor. A graph of each reactor type monitor versus burnup can be found in Appendix C.

MOX fuels can also affect our ability to use ^{240}Pu as a reactor type monitor. The assumption was made that no plutonium was present in the fresh fuel. If plutonium is present this will greatly affect calculations of the ^{240}Pu content, and will cause errors in attempts to distinguish the different reactors with ^{240}Pu.

4. AGE MONITORS

Two characteristics should be considered when choosing an age monitor: They are the monitors half-life and ORIGEN's ability to accurately predict the monitor. Table III lists all of the isotopes that have a half-life between 1 and 30 years and is produced in sufficient quantities in spent fuel that it can be measured.

Unlike reactor type monitors, the accuracy of any individual age monitor is more important, since it is not averaged over a large number of nuclides. The reported age is calculated from just two of the age monitors. If one of them has a significant error, it will directly impact the results. The accuracy of the age monitors can vary significantly, each one should be benchmarked with measured data.

TABLE III

List of All Isotopes That Are Produced in a Reactor with a Half-life Between 1 and 30 Years

Isotope	**Half-Life (years)**
^{90}Sr	28.8
^{93}Nb	16.1
^{106}Ru	1.0
^{101}Rh	3.3
^{102}Rh	3.7
^{125}Sb	2.8
^{134}Cs	2.1
^{137}Cs	30.1
^{146}Pm	5.5
^{147}Pm	2.6
^{153}Eu	13.5
^{154}Eu	8.6
^{155}Eu	4.8
^{228}Th	1.9
^{236}Pu	2.9
^{241}Pu	14.3
^{243}Cm	29.1
^{244}Cm	18.1

5. SUMMARY OF MONITOR NUCLIDES

A summary table listing all of the suggested monitor nuclides for the nuclear forensics problem is given in Table IV.

TABLE IV

Suggested Monitor Nuclides

Burnup Monitors	^{140}Ce, ^{100}Mo, ^{98}Mo, ^{97}Mo, ^{138}Ba, ^{142}Ce, ^{148}Nd
Enrichment Monitors	^{234}U, ^{235}U, ^{236}U, ^{238}U, ^{239}Pu, ^{240}Pu, ^{241}Pu
Reactor Type Monitors	^{109}Ag, ^{153}Eu, ^{156}Gd, ^{143}Nd, ^{240}Pu, ^{108}Cd, ^{113}Cd, ^{149}Sm, ^{166}Er, ^{132}Ba, ^{98}Tc, ^{115}In, ^{72}Ge, ^{115}Sn
Age Monitors	^{90}Sr, ^{93}Nb, ^{106}Ru, ^{101}Rh, ^{102}Rh, ^{125}Sb, ^{134}Cs, ^{137}Cs, ^{146}Pm, ^{147}Pm
Shutdown Monitors	^{90}Zr, ^{113}In, ^{125}Te, ^{134}Ba, ^{137}Ba, ^{147}Sm, ^{154}Gd

CHAPTER III

NEMASYS

A Microsoft Visual Studio .NET 2003, Visual Basic code was written to implement the methodology described in Chapter II. This code was named the Nuclear Event Material Attribution SYStem, or NEMASYS. NEMASYS is GUI (graphical user interface) based and user-friendly. The code was written on a Microsoft Windows XP platform and requires Microsoft's .NET framework for proper execution.

A. CODE REQUIREMENTS

The code needed to meet several goals and requirements. The first goal was to provide a way for a user who only has a basic understanding of nuclear engineering principles the ability to obtain useful information in identifying the radiological source in an RDD. The code needed to be easy to operate. The code also needed the ability to provide all of the necessary information a technical person would want. This would give someone who is familiar with the process and theory behind the code the ability to check the conclusions and make adjustments as necessary.

NEMASYS is only needed in times of emergencies and, therefore, will not be used on a regular basis. Thus if an event occurs NEMASYS must be useable regardless of whether or not trained personnel are available. Therefore, it needs to run in an intuitive manner that is user friendly. There will not be time for the user to read a large manual or spend

days understanding how the process works. If an event occurs, the user will have a few days at best to process the data and produce a detailed report on their findings.

B. UNDERSTANDING NEMASYS

In Chapter II, the calculation of each characteristic was discussed. Here, the connection of these steps will be shown and how they provide information for each other. Each step provides the data needed for the next step until a solution is found. Figure 15 shows a graphical overview of the flow of data in NEMASYS. Before NEMASYS can perform any calculation, it must first load all the necessary information about each reactor and the list of isotopes to be used as monitors. The list of monitors and reactor types are pulled from a Microsoft Access database (see Appendix D). This was done to give the user the ability to add new monitors and reactor types without having to do any re-coding. A text input file is also used to store the location of the files needed for each reactor type. This includes the cross-section and yield data files and a skeleton ORIGEN model. It is stored in a text file to allow SENTRY the ability to modify and change the location of the data and the data itself. For example, if a better cross-section set is found, a user could replace the old cross-section set through a SENTRY interface (Lotus Notes).

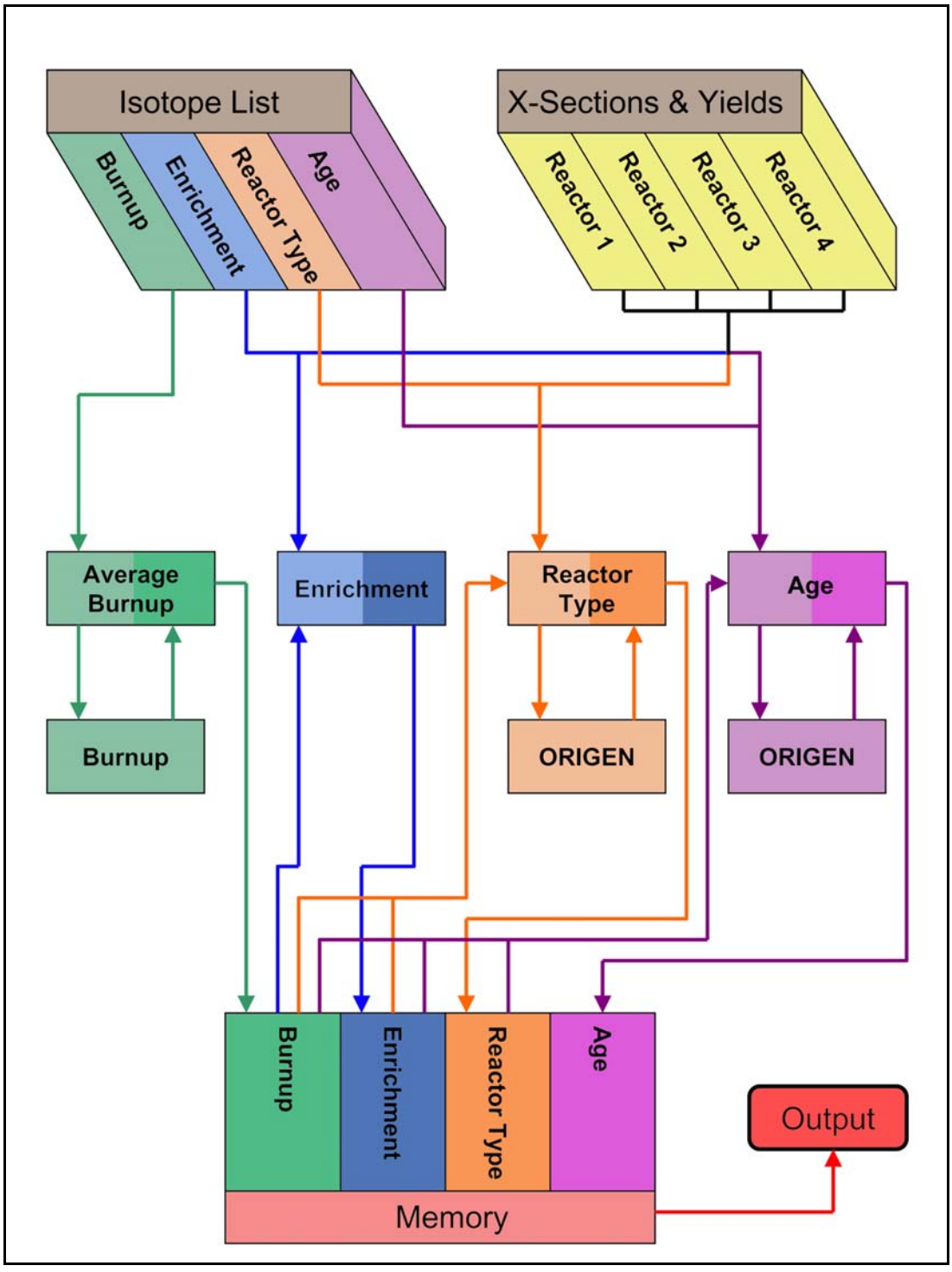

Fig. 15. Overview of the data flow within NEMASYS.

1. BURNUP DETERMINATION

Burnup requires no previous calculations nor information about the different reactor types. It is a straightforward calculation that only requires the list of burnup monitors and the yield for each monitor. When choosing a burnup monitor we made the assumption that the yield was the same for all reactor types was made. Therefore, NEMASYS can pull the necessary yield data from any of the reactor types. By default, NEMASYS gets the yield information from the first reactor type listed in the text file just mentioned in Section III.B. First, it calculates the burnup for each monitor; then, it averages the burnup data using:

$$BU(T) = \frac{1}{n}\sum_{i}^{n} BU_i(T). \tag{3.1}$$

The average burnup is then stored in memory.

2. ENRICHMENT DETERMINATION

NEMASYS then uses the average burnup value to calculate the enrichment. The monitors needed for the enrichment are set and can not be changed. They are the only isotopes that are hard coded into NEMASYS. The enrichment calculation requires cross sections for each of the reactors (see Fig. 15). The cross-sections will vary for each reactor type; thus, the enrichment is calculated for each reactor type being considered and stored in memory. This calculation can be done very quickly with no significant time penalty.

3. REACTOR TYPE DETERMINATION

Identifying the reactor type uses a straightforward concept, but the process itself can be very complicated. As explained in Chapter II, reactor type monitors are chosen because their production varies for each reactor type. By knowing the burnup and enrichment, ORIGEN can be used as a forward model to calculate the quantity of each reactor type monitor that each reactor type should produce at the time the fuel is discharged from the reactor. The calculated value is then compared with the value entered by the user for each reactor type monitor. The errors are averaged by:

$$E_{average} = \frac{1}{n}\sum_{i}^{n} E_i \,, \tag{3.2}$$

and the reactor types are ranked in order by the lowest error. The average error is then used to find the confidence:

$$Confidence = \exp(-E_{average}) \,. \tag{3.3}$$

The five reactor types with the lowest average error are stored in memory.

To perform the ORIGEN calculation, a skeleton ORIGEN model is created for each reactor type (see Appendix E). Any time NEMASYS needs to run ORIGEN, it will only need to modify the skeleton model and then call to execute ORIGEN. This is used to predict the quantity of a particular monitor for any reactor type. ORIGEN automatically calculates the production of all major isotopes. Therefore, no modification is required to use additional monitors. To add a reactor type, the user only needs to create a simple

skeleton ORIGEN model. NEMASYS then modifies the skeleton model and runs ORIGEN to make all of the production calculations. First, NEMASYS pulls the burnup and enrichment data from memory. Then, NEMASYS will modify the correct skeleton ORIGEN model with the burnup and enrichment values. NEMASYS then pulls the correct cross section and yield data for ORIGEN, and ORIGEN is run and data collected for each reactor type. The values are compared with the user inputted values as explained before, and the reactors are ranked. This information is then stored in memory. While the use of forward model calculations increases computational time, the overall increase is small. Each time ORIGEN is run, it requires approximately one second on a single 3.2 GHz Intel processor.

4. AGE DETERMINATION

To find the age, NEMASYS needs the burnup, enrichment, and cross-section and yield data for ORIGEN. Once again, NEMASYS modifies the skeleton models of each reactor type and sets the decay to 30 days. This is done to make sure any short-lived fission products that decay into the age monitor are accounted for. ORIGEN is run and the quantity of each age monitor is collected. This is set as the quantity of the age monitor at the time of discharge. The age is calculated by using the decay equation for each age monitor. Then, the average age is found as follows:

$$Age_{average} = \frac{1}{n}\sum_{i}^{n} Age_i \ , \tag{3.4}$$

and the age is re-calculated using the two age monitors that have the closest half-life to

the average. Thirty days are then added to the average age to account for the 30 days added to the ORIGEN models. This process is repeated for each reactor type and stored in memory.

C. FORWARD MODEL ITERATION

As mentioned in Chapter II, the results for the enrichment calculation have errors as high as 15%. Both the reactor type and age were determined using the calculated enrichment values. This naturally leads to propagated errors in the reactor type and age prediction. To improve this result, a forward model iteration was implemented (see Fig. 6). Once the iteration process is complete, the reactor type and age are re-determined using the new enrichment and burnup values. This process runs ORIGEN an extra 10 to 30 times for each reactor type, causing the process to take several minutes to run on a 3.2 GHz Intel processor.

The iteration process is not run automatically. NEMASYS makes its first calculations without the iteration process and reports the results. This will take approximately two seconds for every reactor type being considered. NEMASYS then gives the user the option to run the iteration process for only one reactor type or for all of the reactor types. If the iteration process is run for all the reactor types, they are re-ranked at the end.

D. USING NEMASYS

Nemasys can be used after a few simple instructions. After starting NEMASYS the first task is to input the measurements. The area at the top left hand side (see Fig. 16) is where the measured values can be entered. First click on the isotope in the list on the left. Then enter the value and the measured error. If the error associated with the measurement is unknown, leave it at zero. After clicking on "Enter Ratio" the isotope will appear on the right with its measurement.

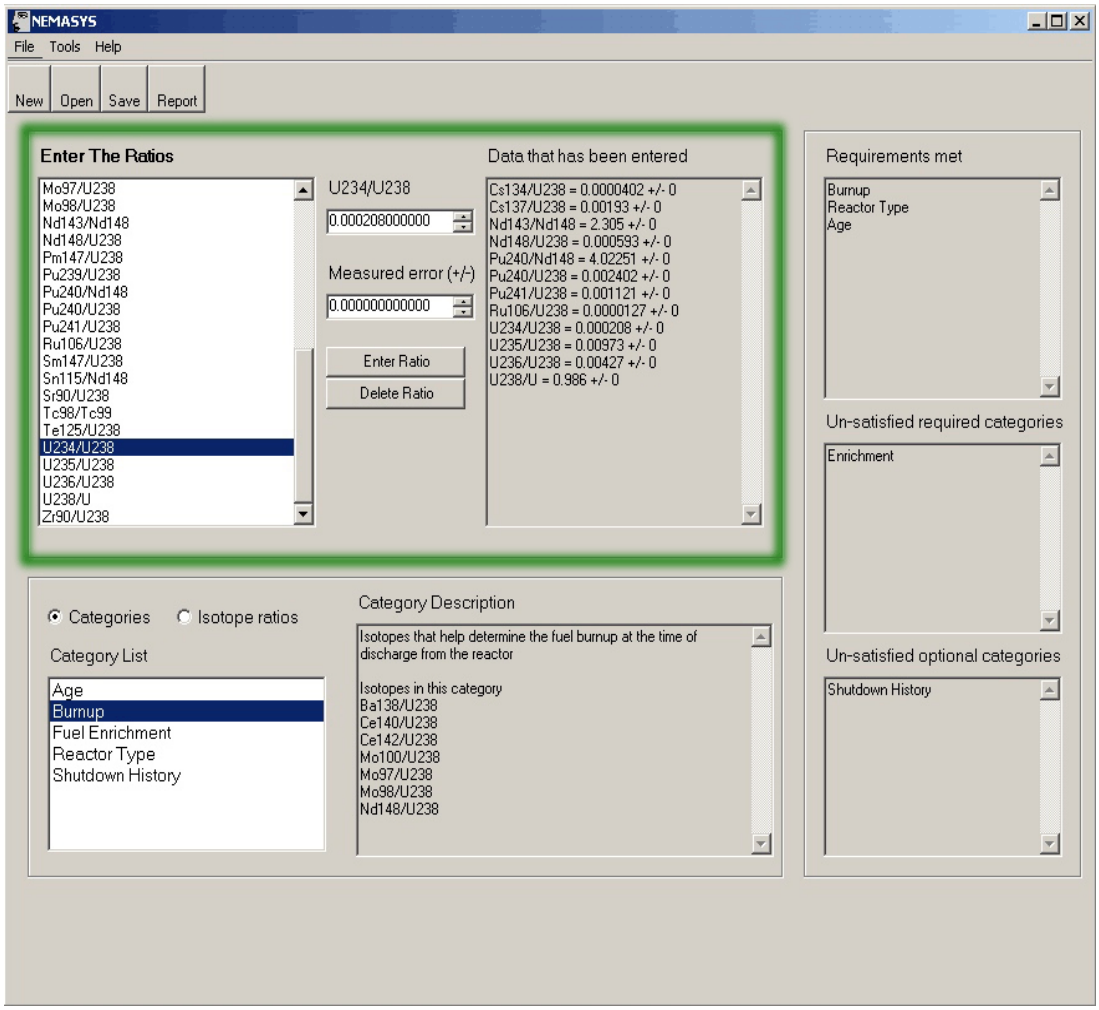

Fig. 16. Entering measured values into NEMASYS.

Each category requires certain isotopes. As the requirements are met for each category by entering in the measured values, the category description (e.g. Burnup, Enrichment, Reactor Type and Age) will appear at the top in the "Requirements met" box (see Fig. 17). Only after all the requirements are met will NEMASYS allow the user to process the data.

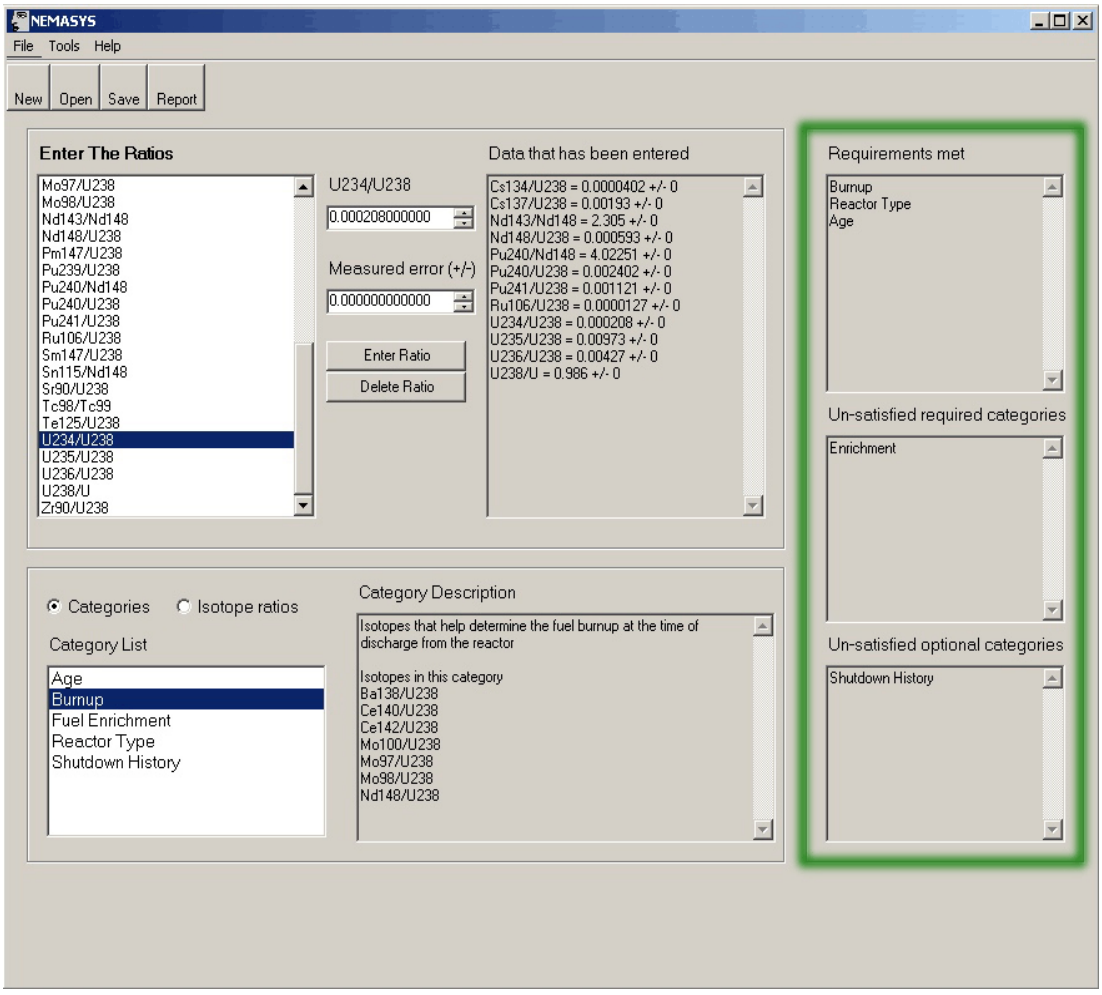

Fig. 17. Meeting the requirements for NEMASYS to process the data.

If the user is uncertain about what isotopes belong in what categories, that information can be obtained in the bottom left hand box (see Fig. 18). By clicking on the category in the list on the left, it will display all of the isotopes in that category on the right. This will help identify what measurements need to be taken. If the user wants to look up what category a particular isotope is in, select the "Isotope ratios" button and the list on the left will be populated with all of the isotopes. By selecting an isotope it will display a description and the category on the right.

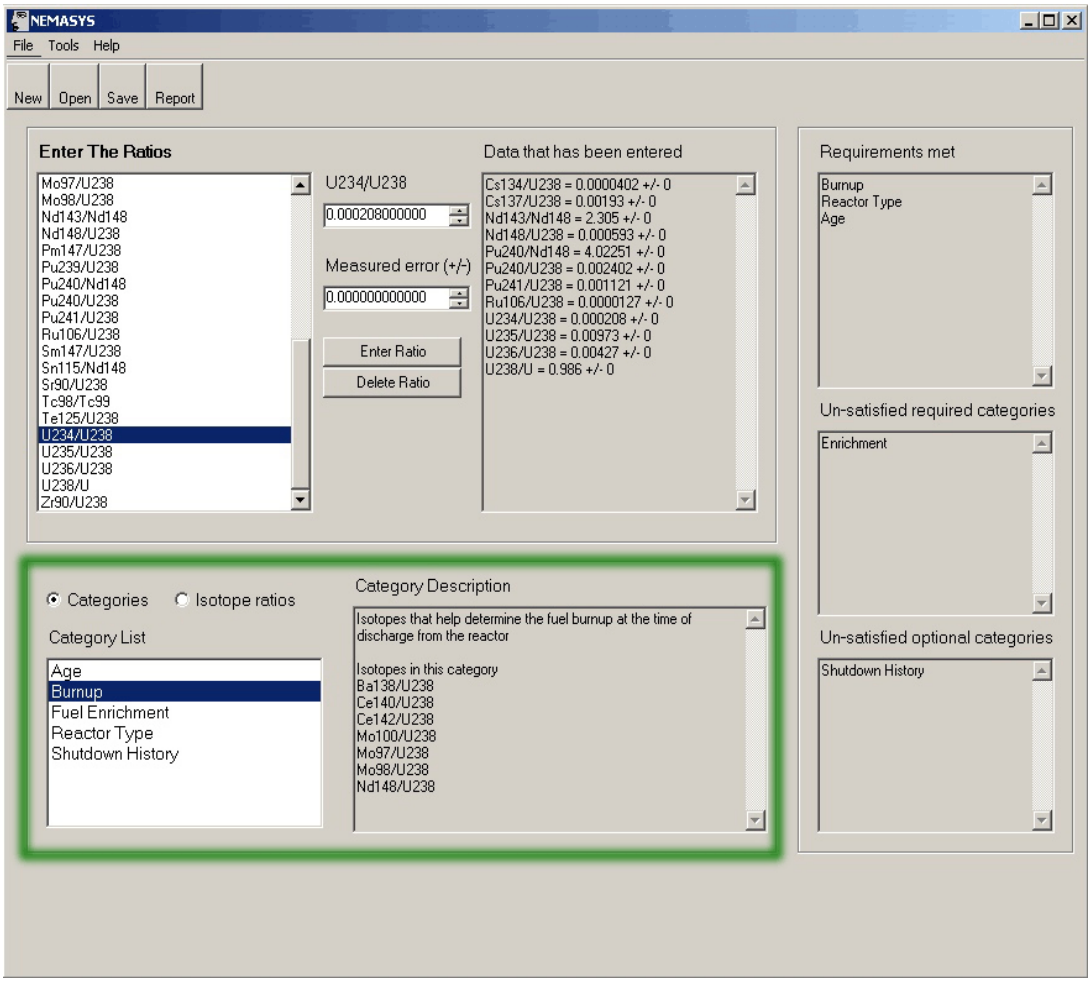

Fig. 18. Viewing the descriptions of the categories and individual nuclides.

After all of the requirements have been met, click on the “Report” button. This process will take approximately 30 seconds to run. It will the display a report similar to Fig. 19. There are five reactor types listed by their confidence rating. The confidence level does not consider the burnup, enrichment or age predictions; it is based solely on the reactor type prediction. The reactor that most closely matches the reactor type monitors are listed first. The burnup calculation is the same for each reactor and is listed at the top. To perform the ORIGEN iteration process you can select a single reactor type and click on “Iterate Selected.” Or the iteration process can be ran for all of the reactors by clicking on “Iterate All.” If all of the reactor types are iterated, it will re-sort the reactors by the new confidence rating.

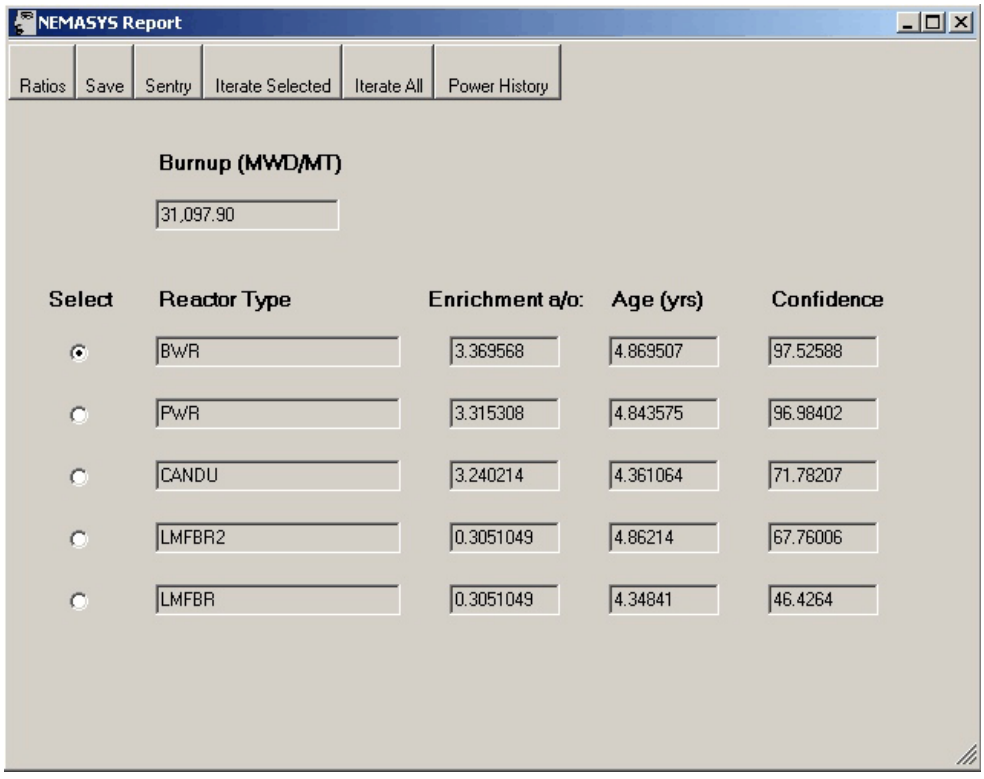

Fig. 19. The NEMASYS reactor report.

If the user would like to see the detailed results for a particular reactor type, select the reactor type and then click on the “Ratios” button. This will bring up a report similar to that shown in Fig. 20. At the top it lists each burnup monitor and the burnup calculation associated with it. The reported burnup is the burnup shown on the report above (Fig. 19). Two age monitors are listed because the age is always calculated by only two monitors, regardless of how many age monitors are entered. The reactor type monitors are listed at the bottom. The measured value is the value entered in by the user. The calculated value is the value calculated by ORIGEN. The errors for each monitor is shown to allow the user to verify the validity of the prediction and measurements. If one particular monitor has a large error, the user can remove that monitor and rerun the report to make sure it wasn’t adversely affecting the results. The large error can be due to ORIGEN’s inability to predict it, a measurement error, or an error while entering the measurement.

The description above is not a full manual but it does allow someone who is unfamiliar with NEMASYS to use it and understand its basic functionality. A full manual will also explain how to add monitors and reactor types. Enhancements will continue to be made in the future as needed.

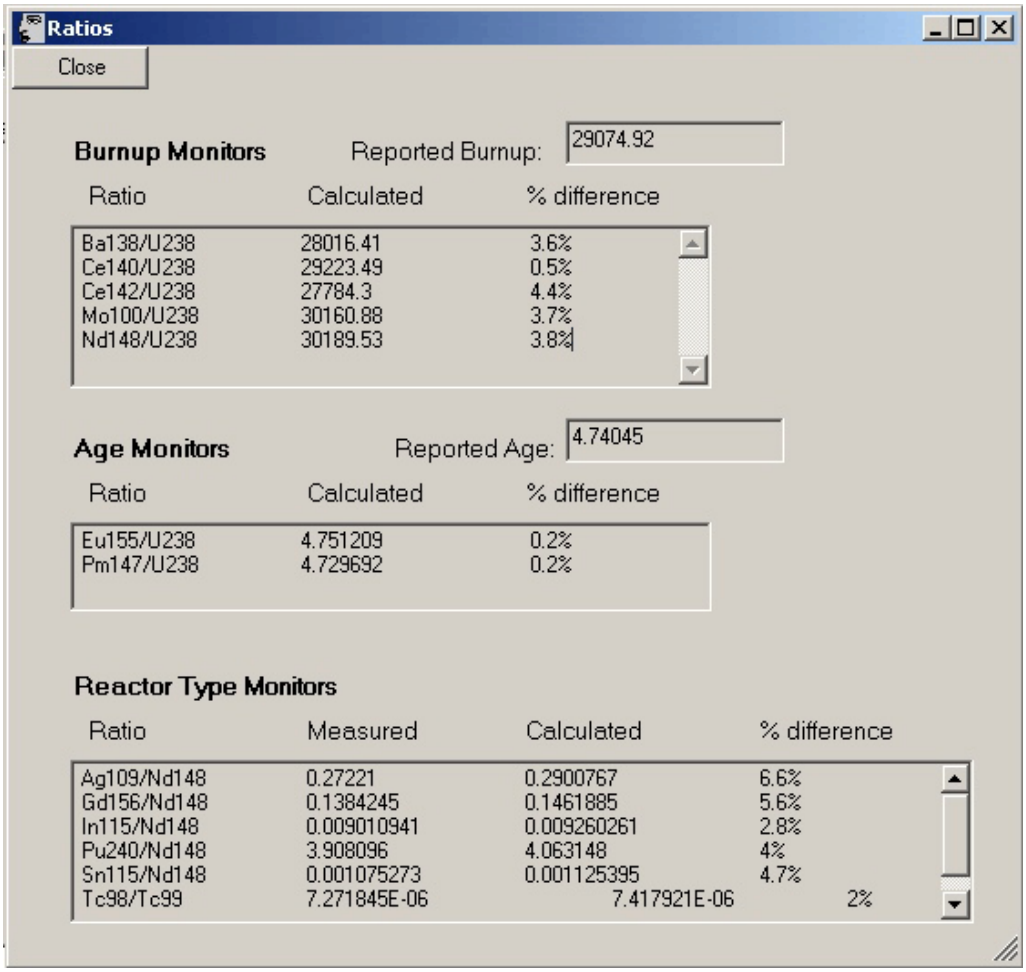

Fig. 20. NEMASYS individual ratio report for each reactor type.

CHAPTER IV

BENCHMARKING NEMASYS

Benchmarking the results is critical to validating the methodology and process used. Unfortunately, there is limited data available on the isotopic composition of spent fuel. Obtaining spent fuel composition measurements requires enormous resources. The data that is available is limited mainly to actinide content and a few fission products. A generally complete listing of possible isotopic spent fuel measurements can be found in the NEA data bank [19]. A complete set of data that met the minimal requirements for analysis by NEMASYS was found for a reactor in Japan. The measurements were from the Mihama Unit 3 reactor.

A. MIHAMA-3 REACTOR

The Mihama nuclear reactor is located in the Fukui prefecture in Japan. Nine samples were taken from three different spent fuel assemblies from the third unit (Mihama-3). The Mihama-3 reactor is a typical U.S. PWR design with a 15×15 rod array. It is operated by Kansai Electric Power Co. and started commercial operation in 1978. The power history of the fuel assemblies measured can be seen in Table V. Three samples were taken from the first assembly (JPNNM3SFA1), two samples from the second assembly (JPNNM3SFA2), and four samples from the third assembly (JPNNM3SFA3).

TABLE V

Power History of Three Assemblies from the Mihama-3 Unit

Assembly JPNNM3SFA1		**Assembly JPNNM3SFA2**		**Assembly JPNNM3SFA3**	
Days	**Condition**	**Days**	**Condition**	**Days**	**Condition**
229	Operation	229	Operation	229	Operation
		375	Shutdown	375	Shutdown
		345	Operation	345	Operation
				163	Shutdown
				389	Operation

B. MONITORS USED FOR MIHAMA-3

Table VI shows the isotope ratios used for monitors in each category. $^{234}U/^{238}U$ was not given in the data, so it was assumed to be 2.76×10^{-4}, which is the amount found in a typical PWR reactor. The $^{234}U/^{238}U$ value was shown to have a negligible effect on the results, so this assumption should be reasonable. It was found to have no visible effect. For the reactor type, there was no fission product data available to differentiate a PWR from a BWR. The accuracy of the reactor type prediction is thus limited due to this lack of available data.

TABLE VI

Monitors Used for Attribution of the Mihama-3 Unit

$^{148}Nd/^{238}U$	Burnup
$^{235}U/^{238}U$	Enrichment
$^{236}U/^{238}U$	Enrichment
$^{238}U/U$	Enrichment
$^{239}Pu/^{238}U$	Enrichment
$^{240}Pu/^{238}U$	Enrichment
$^{143}Nd/^{148}Nd$	Reactor Type
$^{240}Pu/^{148}Nd$	Reactor Type
$^{241}Pu/^{238}U$	Age
$^{134}Cs/^{238}U$	Age
$^{137}Cs/^{238}U$	Age
$^{106}Ru/^{238}U$	Age

C. NEMASYS RESULTS FOR MIHAMA-3

Table VII shows the NEMASYS results for the burnup prediction for each sample. Table VII shows a consistent lower predicted burnup than the reported burnup. The overall uncertainty in the results however is quite good. A 4% error is sufficient for the forensics problem as long as it does not affect the other calculations significantly.

TABLE VII

NEMASYS Burnup Results for Mihama-3

Assembly	Sample No.	Reported Burnup (MWd/MT)	Predicted Burnup (MWd/MT)	Error
JPNNM3SFA1	1	8,300	7,952	-4.19%
JPNNM3SFA1	2	6,900	6,678	-3.22%
JPNNM3SFA1	3	15,300	14,664	-4.16%
JPNNM3SFA2	4	21,200	20,399	-3.78%
JPNNM3SFA2	5	14,600	14,043	-3.82%
JPNNM3SFA3	6	29,400	28,394	-3.42%
JPNNM3SFA3	7	32,300	30,931	-4.24%
JPNNM3SFA3	8	33,700	32,371	-3.37%
JPNNM3SFA3	9	34,100	32,920	-3.46%
Average Error	-	-	-	-3.74%

In Section II.B., Table II shows the results for the enrichment calculation. Before using the forward model iteration method, the errors were as high as 14.5%. By using the forward model iteration method, the error was reduced to less than 2%. This is sufficient for the forensics problem and gives us greater confidence in our reactor type and age predictions.

For the reactor type predictions, NEMASYS easily predicted that the Mihama reactor was not an LMFBR or CANDU reactor but had difficulty distinguishing between a PWR

and BWR. NEMASYS predicted PWR correctly only twice out of the nine samples. It reported the difference between the two reactors was less than 1%. Thus, to accurately distinguish a PWR and BWR the monitors discussed in Section II.F.3 must be used.

TABLE VIII

Age Prediction Results for Mihama-3.

Assembly	Sample No.	Reported Age	Calculated Age	Error
JPNNM3SFA1	1	5	4.85	3.0%
JPNNM3SFA1	2	5	4.81	3.8%
JPNNM3SFA1	3	5	4.70	6.1%
JPNNM3SFA2	4	5	4.78	4.4%
JPNNM3SFA2	5	5	4.52	9.6%
JPNNM3SFA3	6	5	4.97	0.6%
JPNNM3SFA3	7	5	4.85	3.0%
JPNNM3SFA3	8	5	5.07	1.4%
JPNNM3SFA3	9	5	5.01	0.2%
Average Error	-	-	-	3.6%

Table VIII shows the results for the age calculation. The age predictions all have errors less than 9.6%. No other attributes depend upon the age prediction and so the error in this result will not propagate. The age could have been used to correct for the decay of ^{241}Pu used in the enrichment prediction, but with the implementation of the ORIGEN iteration scheme this is unnecessary. This result is accurate enough that it will allow for

proper identification of the suspected source reactor.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Terrorism has demanded that the U.S. be prepared for many situations that it never considered a significant threat before. One of these threats is the possibility of an organization detonating an RDD on U.S. soil. The ability to quickly identify the source of the radiological material used in an RDD would aid investigators in identifying the perpetrators. If spent fuel is used as the radiological material for the weapon, it would prove to be deadly and cause extensive disruption. In this work, we developed and implemented a forensics methodology to attribute spent fuel to a source reactor. The specific attributes determined are the spent fuel burnup, age from discharge, reactor type, and initial fuel enrichment.

It was shown that post-event material attribution is possible with enough accuracy to be useful for investigators. The burnup can be found to within a 5% accuracy, enrichment to within a 2% accuracy, and age to within a 10% accuracy. The methodology can discriminate between a CANDU, LMFBR, BWR and PWR. It was implemented into a code call NEMASYS. NEMASYS was written so that its capabilities could be expanded and improved without the need to perform any re-coding. New reactor types and monitors can be added and defined by the user. This gives the user the ability to fine tune the process as more data becomes available. NEMASYS is easy to use, and it takes a minimum amount of time to learn its basic functions. It will process data within a few

minutes and provide detailed information about the results and conclusions.

As stated earlier, this work is only a small part in a larger effort to attribute spent fuel (Fig. 2.). Additional work on NEMASYS and on the complete attributing project still needs to be completed. Additional benchmark data would allow for improvements in the results and building additional confidence in NEMASYS. There are still opportunities to improve the accuracy of the methodology as well. This can be done by automating the process to find outlier monitors, and performing an error analysis of the processes used to find the burnup, enrichment, reactor type and age. Once a more stringent analysis of the methodology is complete, the calculation of error propagation needs to be implemented in NEMASYS. With sufficient benchmarking results, each monitor can be studied in greater detail and verified to work as predicted. This can only be done after more spent fuel measurements are performed. The implementation of the shutdown monitor has yet to be incorporated into NEMASYS. Not only will knowing the shutdown time help with matching the suspected fuel to a power history, but it might aid in the calculation of the age.

Only a few reactor types were considered in our study. This needs to be expanded to include all commercial and research reactors. Large commercial sources are not considered by NEMASYS, and a complete study needs to be performed to allow the implementation of a forensics methodology for large commercial sources.

There are scenarios when NEMASYS will be unable to identify spent fuel sources. These scenarios and any other points of failure need to be documented. The search tool shown in Fig. 2 has not been completed. There is a limited SENTRY search tool available but no attempt has been made to automate the flow of information between NEMASYS and this search tool. This is a critical component of the attribution project and needs to completed soon.

In conclusion, we have shown that with our methodology, attributing spent nuclear fuel is within our capabilities. This can be done quickly and easily with NEMASYS. NEMASYS has enough accuracy to provide useful results but would still benefit from evolutionary improvements by more data for benchmarking and modifications to the methodology.

REFERENCES

1. B. Anet, "Assessing the Risk of Radiological Terrorism: How Real is The Threat," *The Chemical and Biological Medical Treatments Symposia,* April 28 - May 3, Speiz, Switzerland (2002).
2. J.L. Ford, "Radiological Dispersal Devices: Assessing the Transnational Threat," *Institute for National Strategic Studies, National Defense University, Strategic Forum,* No. 136 (1998).
3. C. Watson, "Nuclear Terrorism," *British Pugwash Group: Britain and Unconventional Terrorism,* London, UK, December 10 (2002).
4. A.J. Gonzalez, "Security of Radioactive Sources: The Evolving New International Dimensions," IAEA Bulletin, 43 (4), 39 (2001).
5. G.J. Van Tuyle, T.L. Strub, H. O'Brien, C. Mason, and S. Gitomer "Reducing RDD Concerns Related to Large Radiological Source Applications," LA-UR-03-6664, Los Alamos National Laboratory Report (2003).
6. R. Alvarez, "Don't Short-Change Nuclear Safety: Tightening Security Around Nuclear Storage Facilities Should Be an Urgent National Priority," *Foreign Policy In Focus* (November 2001).
7. S. Ludwig, "ORIGEN The ORNL Isotope Generation and Delpetion Code," CCC-0371/17, Oak Ridge National Laboratory (2002) .
8. F. Pointurier, N. Baglan, and P. Hemet "Ultra Low-level Measurements of Actinides by Sector Field ICP-MS," *Applied Radiation and Isotopes,* **60**, 561 (2004).

9. M.V. Zoriy, C. Pickhardt, P. Ostapczuk, R. Hille, and J. S. Backer, "Determination of Pu in Urine at Ultratrace Level by Sector Field Inductively Coupled Plasma Mass Spectrometry," *International Journal of Mass Spectrometry*, **232**, 217 (2004).

10. J.F. Briesmeister, "MCNP – A General Monte Carlo N-Particle Transport Code Version 4B," LA-12625-M, Los Alamos National Laboratory (1997).

11. D.L. Poston and H.R. Trellue, "User's Manual, Version 2.0 for MONTEBURNS Version 1.0," LA-UR-99-4999, Los Alamos National Laboratory (1999).

12 D.E. Burk, "Forward Model Calculations for Determining Isotopic Compositions of Material Used in a Radiological Dispersal Device," M.S. Thesis, Texas A&M University (2005).

13. A.V. Bushuev, A.F. Kozhin, G. Li, V.N. Zubarev, A.A. Portnov, V. P. Alferov, and M.V. Shchurovskaya, "Determination of the Fuel-assembly Burnup in a Research Reactor by Repeated Short-time Irradiation Followed by G Spectrometric Measurements," *Atomic Energy*, **97**, 2 (2004).

14. J.J. Giglio, D.G. Cummings, M.M. Michlik, P.S. Goodall, and S.G. Johnson, "Determination of Burnup in Spent Nuclear Fuel by Application of Fiber Optic High-resolution Inductively Coupled Plasma Atomic Emission Spectroscopy," *Nuclear Instruments and Methods in Physics Research A* **396**, 251 (1997).

15. W.S. Charlton, B.L. Fearey, C.W. Nakhleh, T.A. Parish, R.T. Perry, J. Poths, J.R. Quagliano, W.D. Stanbro, and W.B. Wilson, "Operator Declaration Verification Technique for Spent Fuel at Reprocessing Facilities," *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, **168**(1), 98 (2000).

16. K. Abbas, G. Nicolaou, D. Pellotiero, P. Schwalback, L. Koch, "Gamma Spectrometry of Spent Nuclear Fuel Using a Miniature Cdte Detector," *Nuclear Instruments and Methods in Physics Research Section A,* **376**(2), 248 (1996).

17. C. Devida, E. Gautier, D. Gil, and A. Stankevicius, "The LFR Facility (CNEA) for Burnup Determination in Uranium Silicide Fuels 20% ^{235}U," *2002 International Meeting on Reduced Enrichment for Research and Test Reactors*, November 3-8, Bariloche, Argentina (2002).

18. G.W. Fox, and J. Matheson, "Dealing with the Legacy of US and Russian Nuclear Defence Programmes - AREVA's Contributions," *World Nuclear Association, Annual Symposium*, September 3-5, London (2003).

19. "List of Reactors in SFCOMPO on WWW Ver.2," http://www.nea.fr/html/science/wpncs/sfcompo/Ver.2/Eng, Nuclear Energy Agency (2005).

APPENDIX A

FINAL ENRICHMENT CALCULATION

The final solution for the initial enrichment calculation is long. To avoid having the equations stretch out across multiple pages, several sub-variables are defined. The sub-variables themselves have no relation to any physical concept and are simply pieces of the solution. The sub-variables H1 - H18 are defined as:

$$H1 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U238}}{N_o^U} \right] \left(\left[\frac{N^{U235}}{N^{U238}} \right] + \left[\frac{N^{U236}}{N^{U238}} \right] \right) \quad \text{(A.1)}$$

$$H2 = -M_o^U BU(T) \bar{\sigma}_a^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \quad \text{(A.2)}$$

$$H3 = N_a E_R \bar{\sigma}_f^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left(1 - \left[\frac{N^{U238}}{N_o^U} \right] \right) \quad \text{(A.3)}$$

$$H4 = -N_a E_R \bar{\sigma}_f^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U234}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.4)}$$

$$H5 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_f^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U239}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.5)}$$

$$H6 = -N_a E_R \bar{\sigma}_\gamma^{U238} \bar{\sigma}_f^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left(1 - \left[\frac{N^{U238}}{N_o^U} \right] \right) \quad \text{(A.6)}$$

$$H7 = -N_a E_R \bar{\sigma}_{\gamma}^{U238} \bar{\sigma}_f^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U234}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.7)}$$

$$H8 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_f^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U240}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.8)}$$

$$H9 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_{\gamma}^{Pu239} \bar{\sigma}_f^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U239}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.9)}$$

$$H10 = -N_a E_R \bar{\sigma}_{\gamma}^{U238} \bar{\sigma}_{\gamma}^{Pu239} \bar{\sigma}_f^{Pu240} \bar{\sigma}_a^{Pu241} \left[\frac{N^{U234}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.10)}$$

$$H11 = N_a E_R \bar{\sigma}_{\gamma}^{U238} \bar{\sigma}_{\gamma}^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_a^{Pu241} \left(1 - \left[\frac{N^{U238}}{N_o^U} \right] \right) \quad \text{(A.11)}$$

$$H12 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_a^{Pu240} \bar{\sigma}_f^{Pu241} \left[\frac{N^{U241}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.12)}$$

$$H13 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_a^{Pu239} \bar{\sigma}_{\gamma}^{Pu240} \bar{\sigma}_f^{Pu241} \left[\frac{N^{U240}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.13)}$$

$$H14 = -N_a E_R \bar{\sigma}_a^{U238} \bar{\sigma}_{\gamma}^{Pu239} \bar{\sigma}_{\gamma}^{Pu240} \bar{\sigma}_f^{Pu241} \left[\frac{N^{U239}}{N^{U238}} \right] \left[\frac{N^{U238}}{N_o^U} \right] \quad \text{(A.14)}$$

$$H15 = N_a E_R \bar{\sigma}_{\gamma}^{U\,238} \bar{\sigma}_{\gamma}^{Pu\,239} \bar{\sigma}_{\gamma}^{Pu\,240} \bar{\sigma}_{f}^{Pu\,241} \left(1 - \left[\frac{N^{U\,238}}{N_o^U} \right] \right) \tag{A.15}$$

$$H16 = \bar{\sigma}_{f}^{U\,238} \bar{\sigma}_{a}^{Pu\,239} \bar{\sigma}_{a}^{Pu\,240} \bar{\sigma}_{a}^{Pu\,241} - \bar{\sigma}_{a}^{U\,238} \bar{\sigma}_{a}^{Pu\,239} \bar{\sigma}_{a}^{Pu\,240} \bar{\sigma}_{a}^{Pu\,241} \tag{A.16}$$

$$H17 = \bar{\sigma}_{\gamma}^{U\,238} \bar{\sigma}_{f}^{Pu\,239} \bar{\sigma}_{a}^{Pu\,240} \bar{\sigma}_{a}^{Pu\,241} - \bar{\sigma}_{\gamma}^{U\,238} \bar{\sigma}_{\gamma}^{Pu\,239} \bar{\sigma}_{f}^{Pu\,240} \bar{\sigma}_{a}^{Pu\,241} \tag{A.17}$$

$$H18 = \bar{\sigma}_{\gamma}^{U\,238} \bar{\sigma}_{\gamma}^{Pu\,239} \bar{\sigma}_{\gamma}^{Pu\,240} \bar{\sigma}_{f}^{Pu\,241} \tag{A.18}$$

The final solution for the initial enrichment is then given by:

$$e_o = \frac{\left(H1 + H2 + ...H15 \right)}{H16 + H17 + H18} \tag{A.19}$$

APPENDIX B

GRAPHS OF BURNUP MONITORS

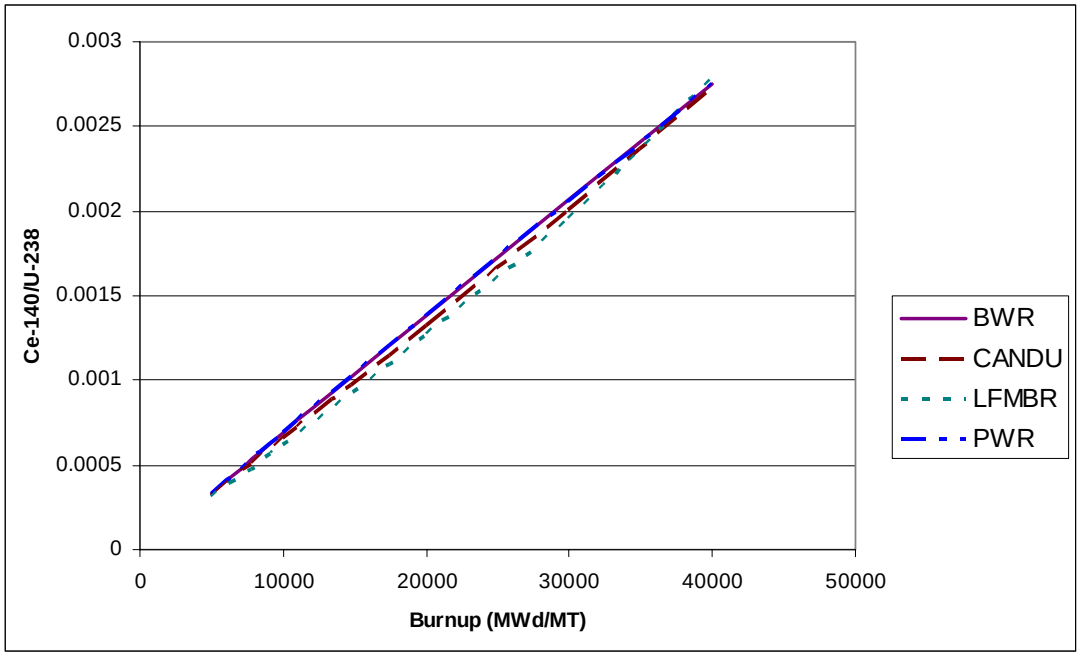

Fig. A1. Atom ratio of ^{140}Ce to ^{238}U for several reactor types.

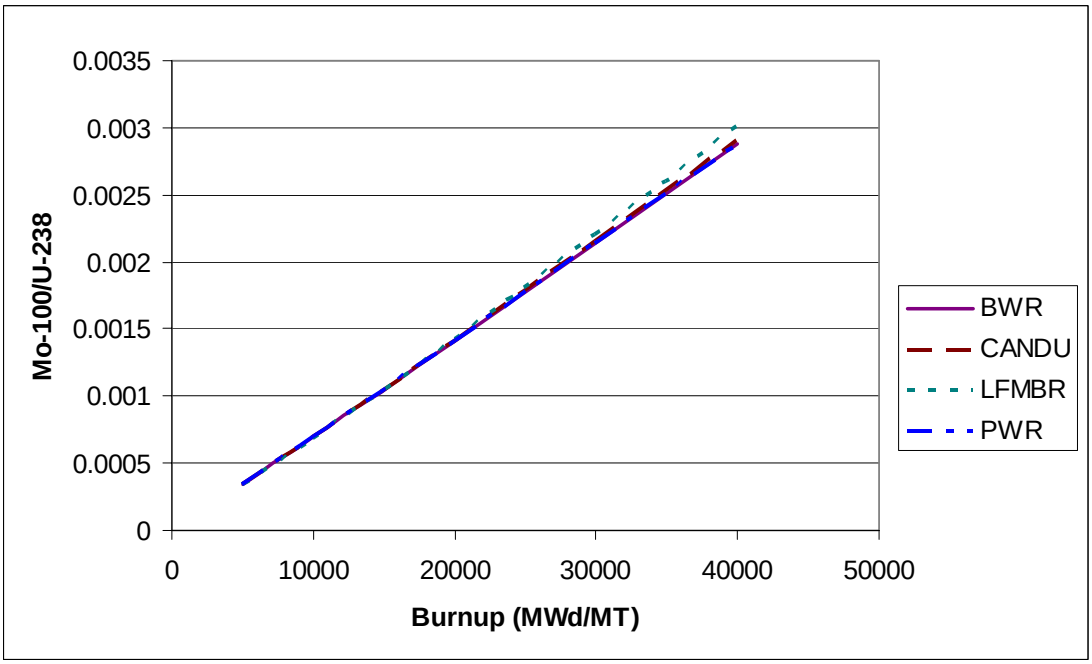

Fig. A2. Atom ratio of ^{100}Mo to ^{238}U for several reactor types.

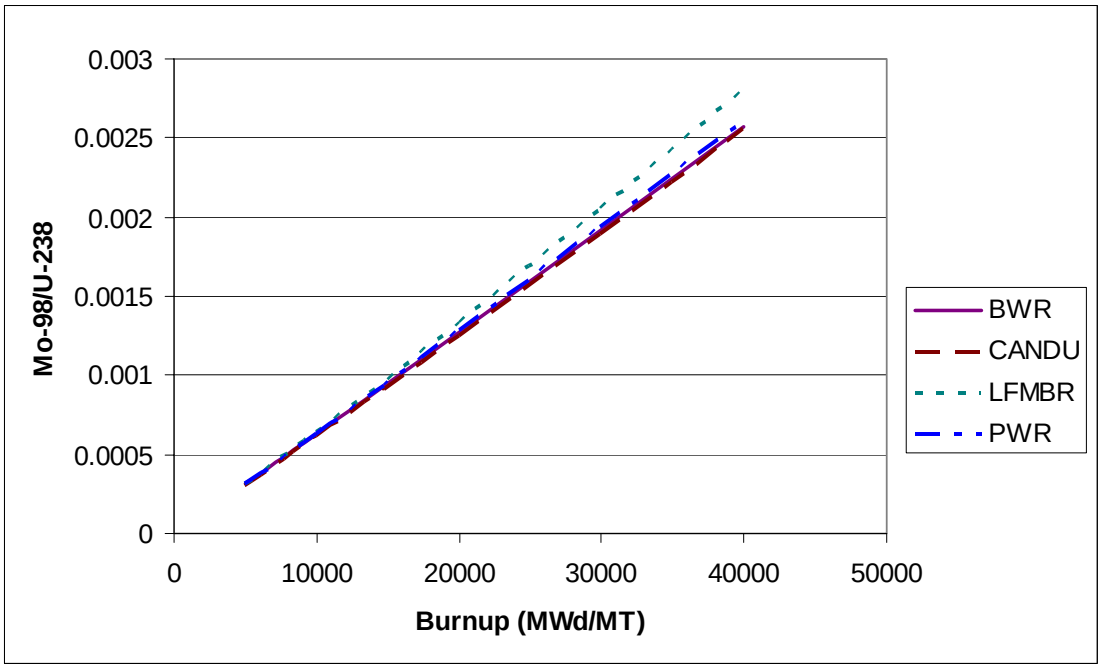

Fig. A3. Atom ratio of ^{98}Mo to ^{238}U for several reactor types.

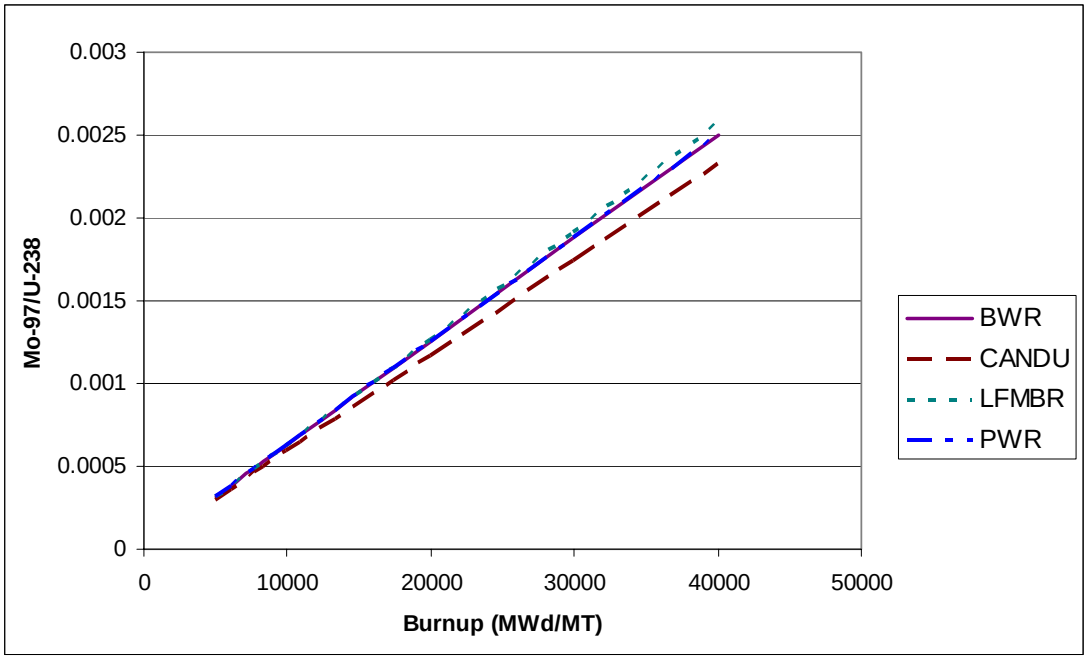

Fig. A4. Atom ratio of ^{97}Mo to ^{238}U for several reactor types.

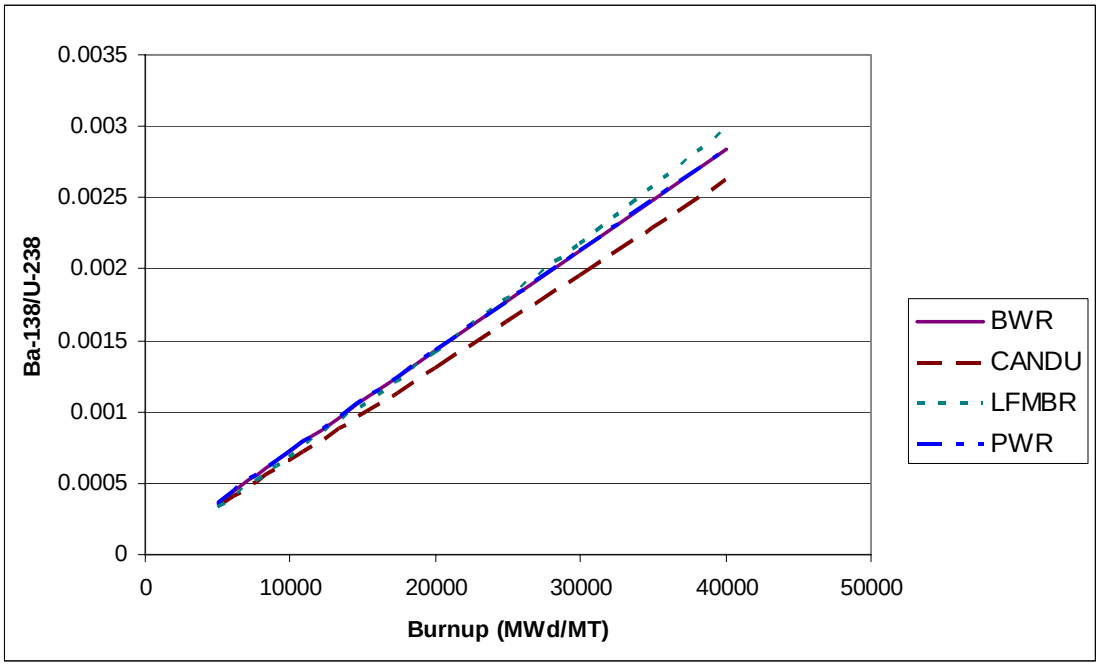

Fig. A5. Atom ratio of ^{138}Ba to ^{238}U for several reactor types.

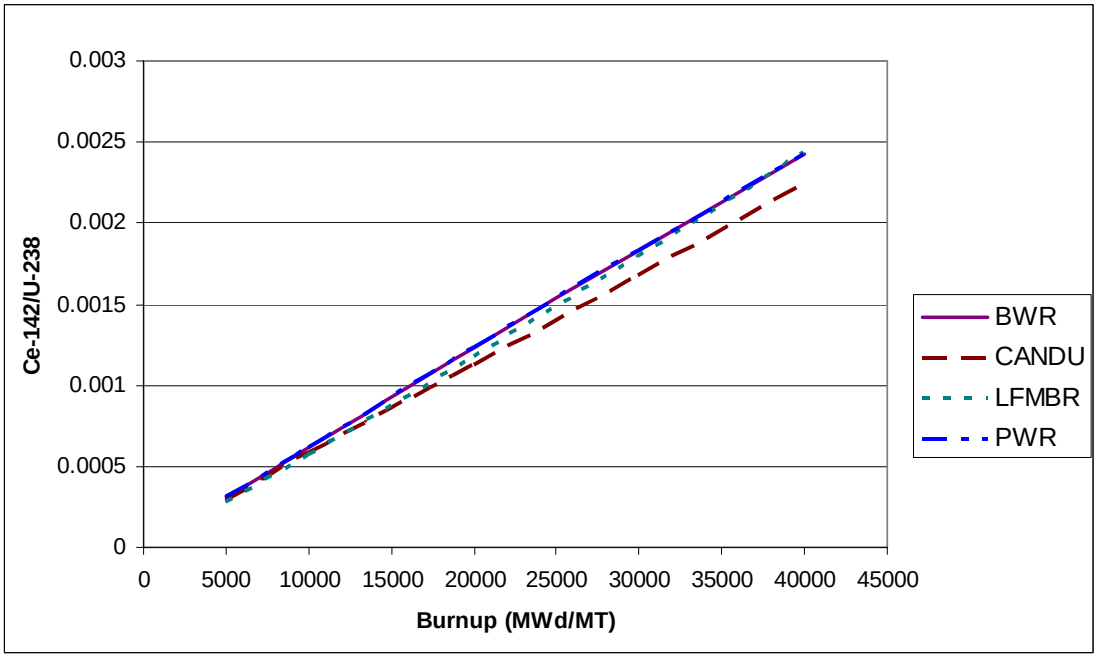

Fig. A6. Atom ratio of ^{142}Ce to ^{238}U for several reactor types.

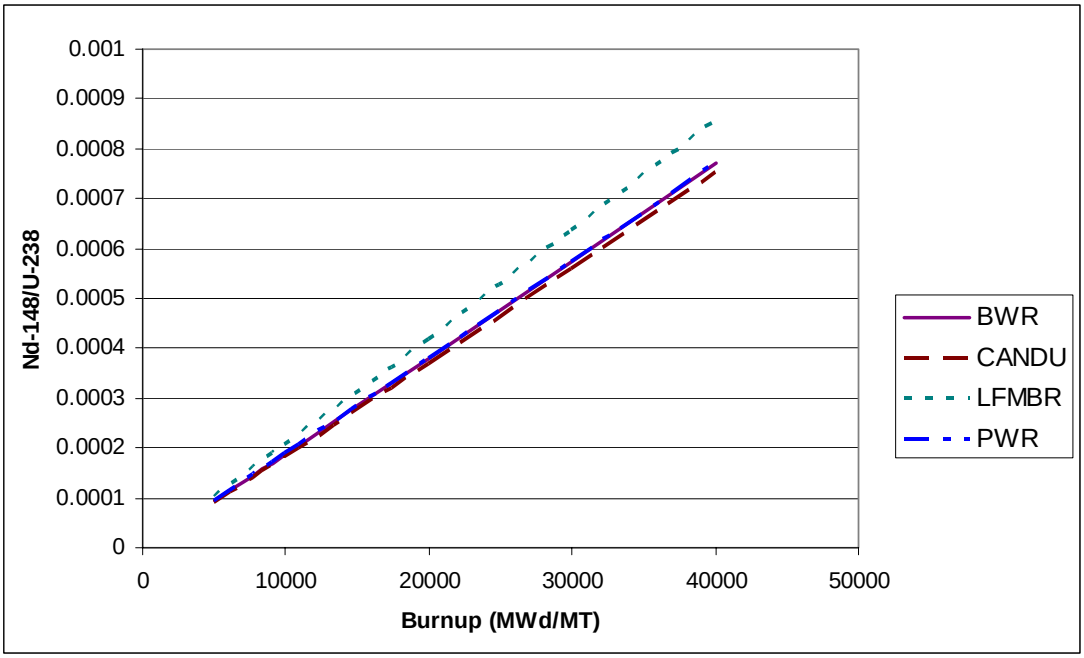

Fig. A7. Atom ratio of ^{148}Nd to ^{238}U for several reactor types.

APPENDIX C

GRAPHS OF REACTOR TYPE MONITORS

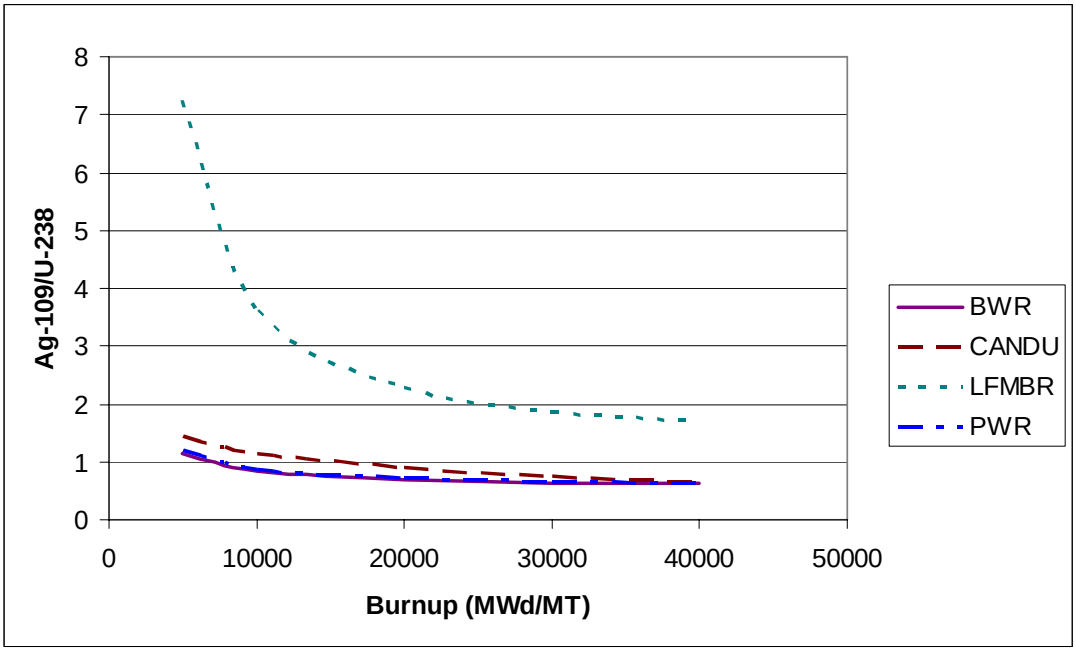

Fig. B1. Atom ratio of ^{109}Ag to ^{238}U for several reactor types.

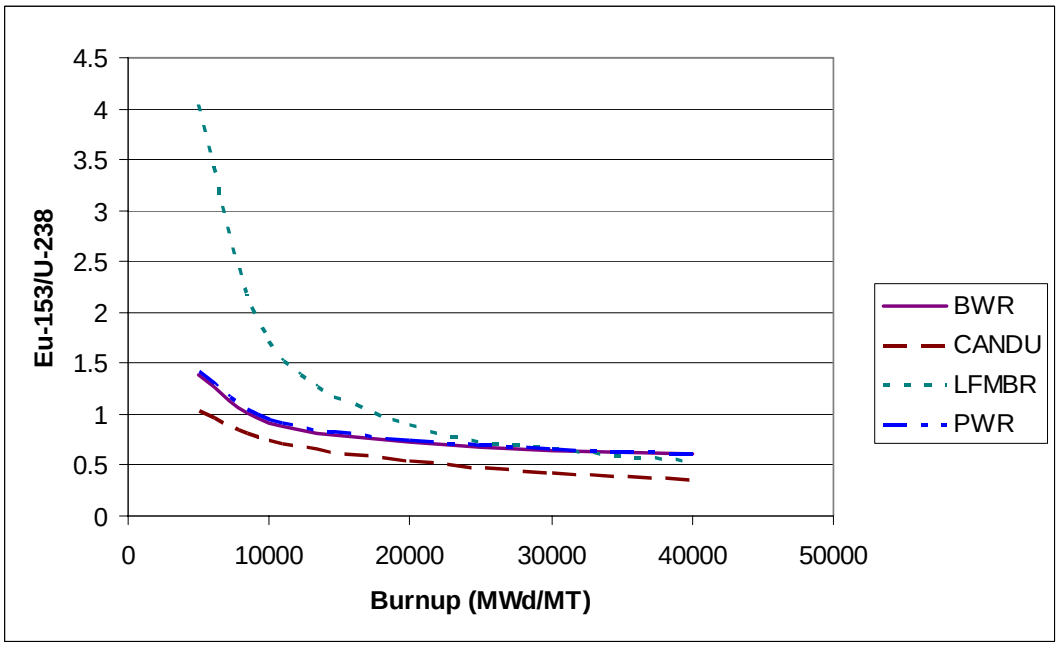

Fig. B2. Atom ratio of ^{153}Eu to ^{238}U for several reactor types.

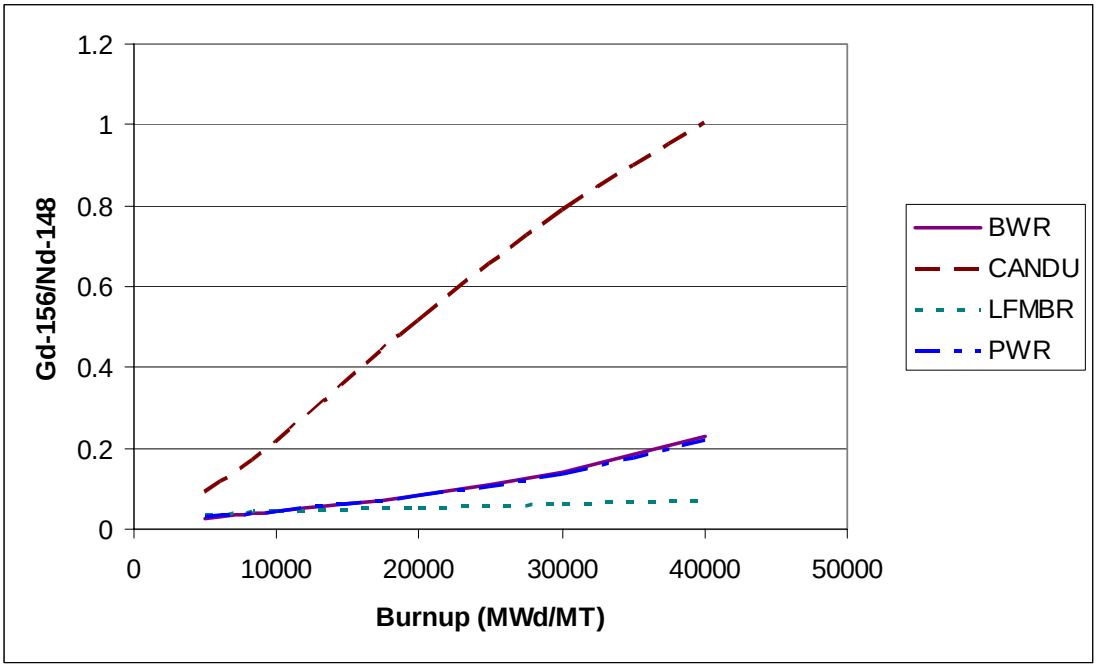

Fig. B3. Atom ratio of ^{156}Gd to ^{238}U for several reactor types.

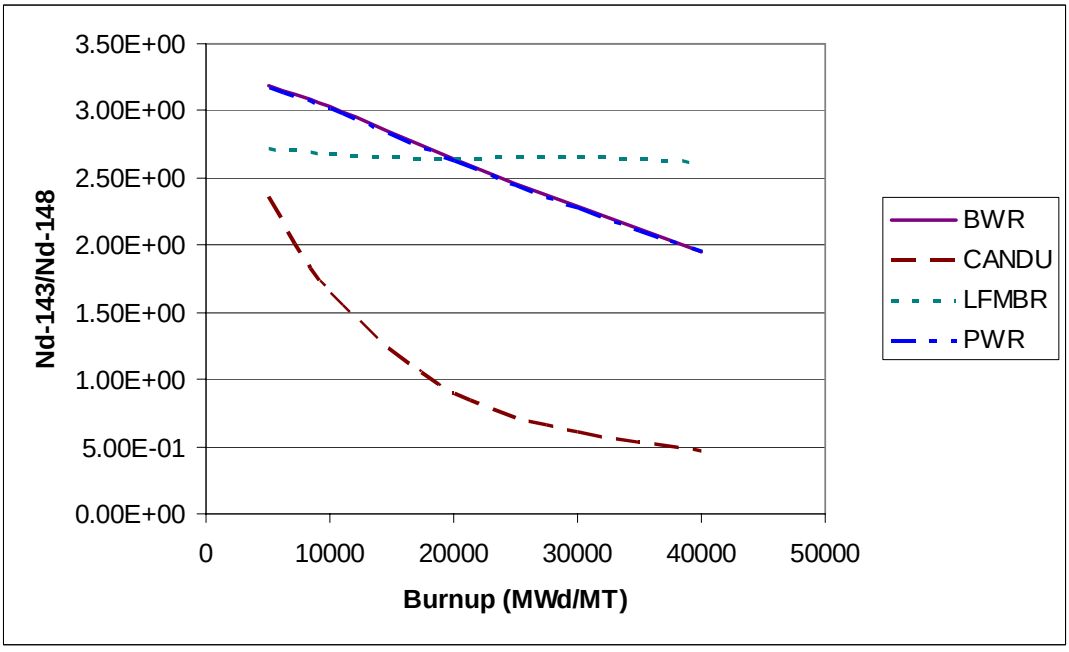

Fig. B4. Atom ratio of ^{143}Nd to ^{238}U for several reactor types.

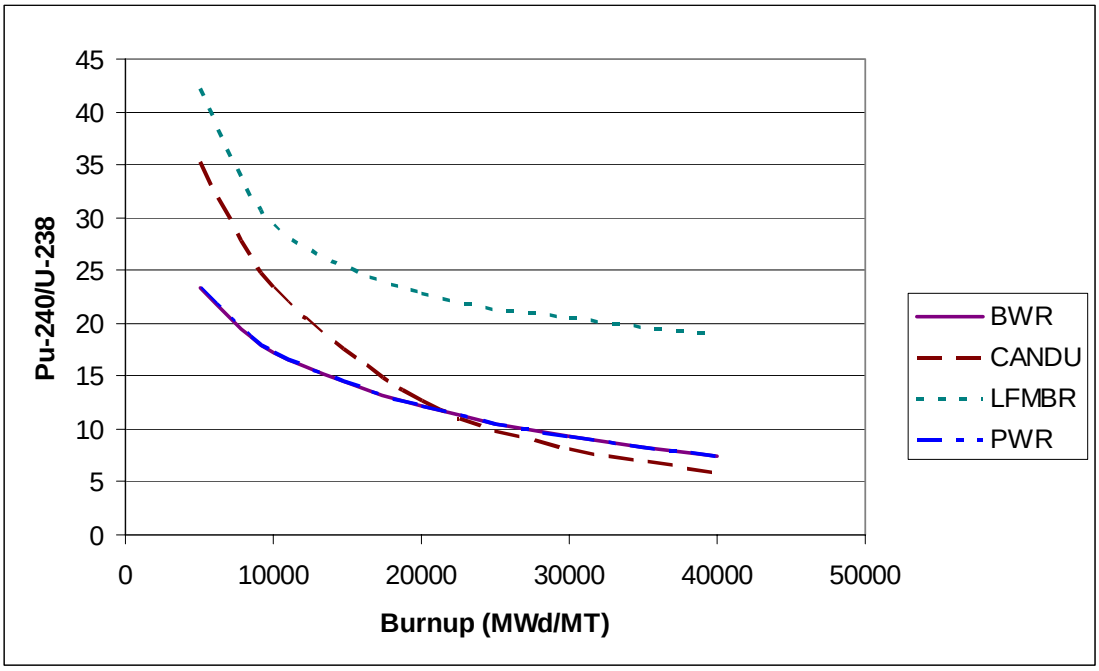

Fig. B5. Atom ratio of ^{240}Pu to ^{238}U for several reactor types.

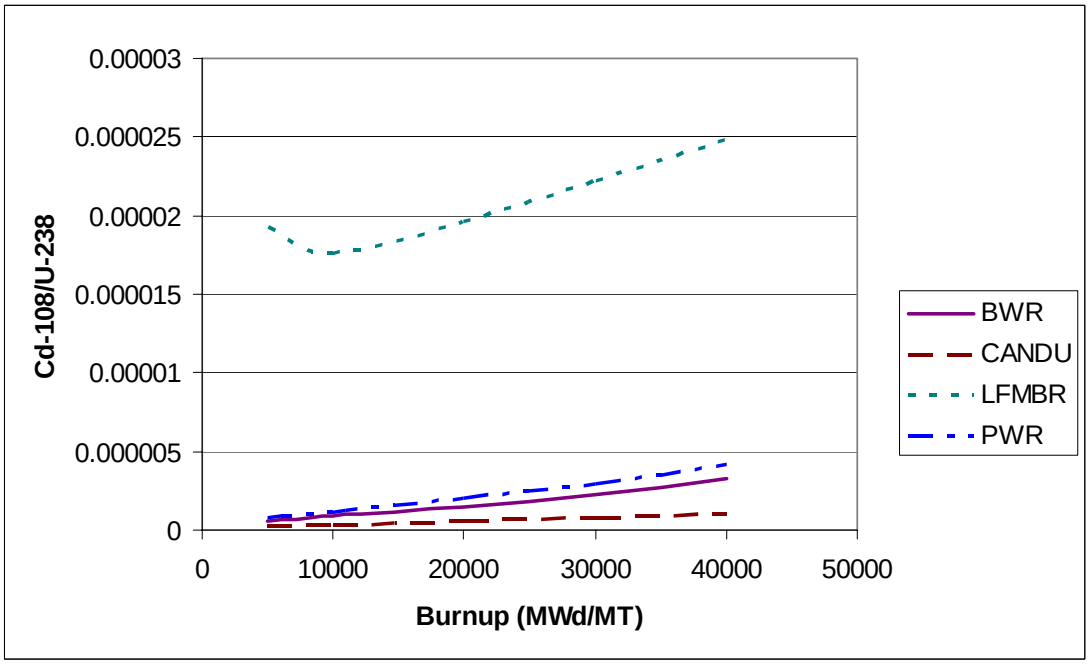

Fig. B6. Atom ratio of ^{108}Cd to ^{238}U for several reactor types.

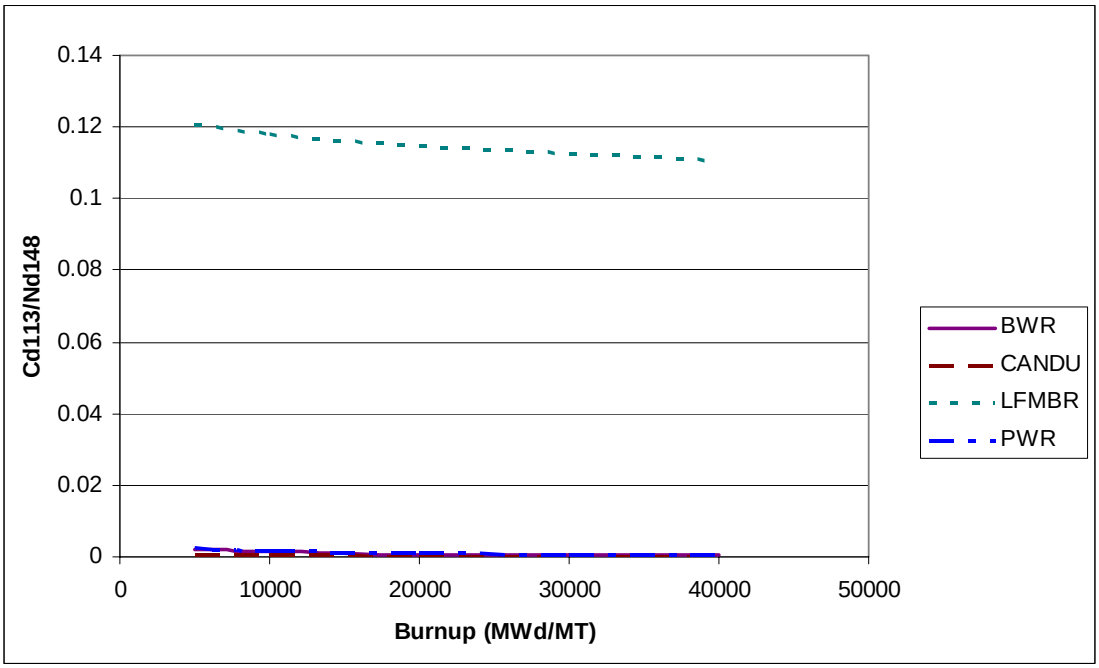

Fig. B7. Atom ratio of ^{113}Cd to ^{238}U for several reactor types.

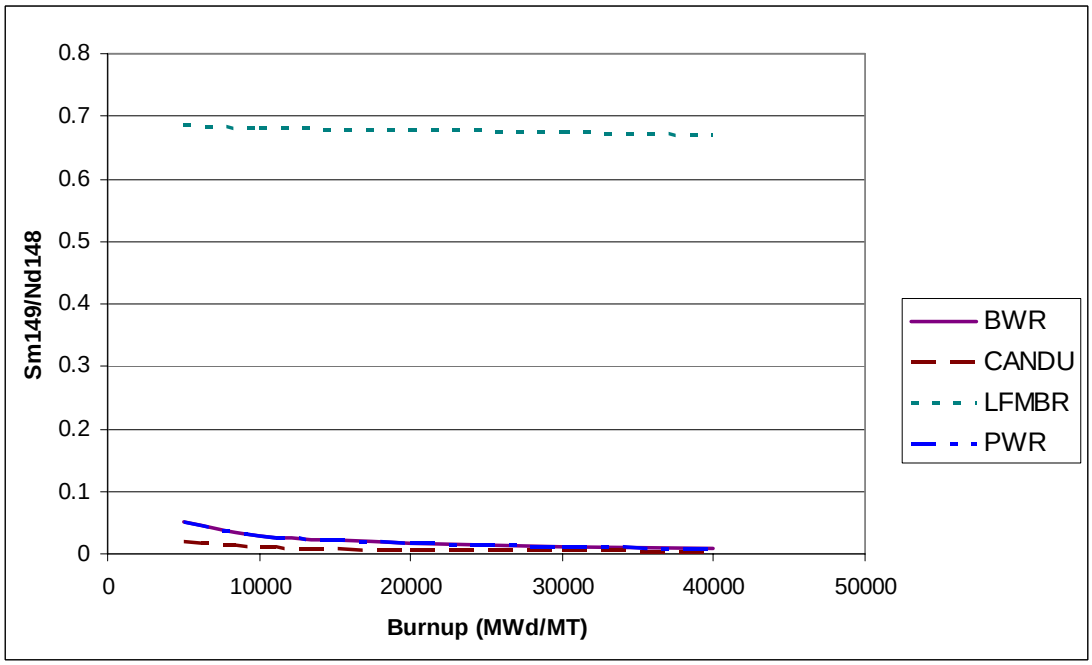

Fig. B8. Atom ratio of ^{149}Sm to ^{238}U for several reactor types.

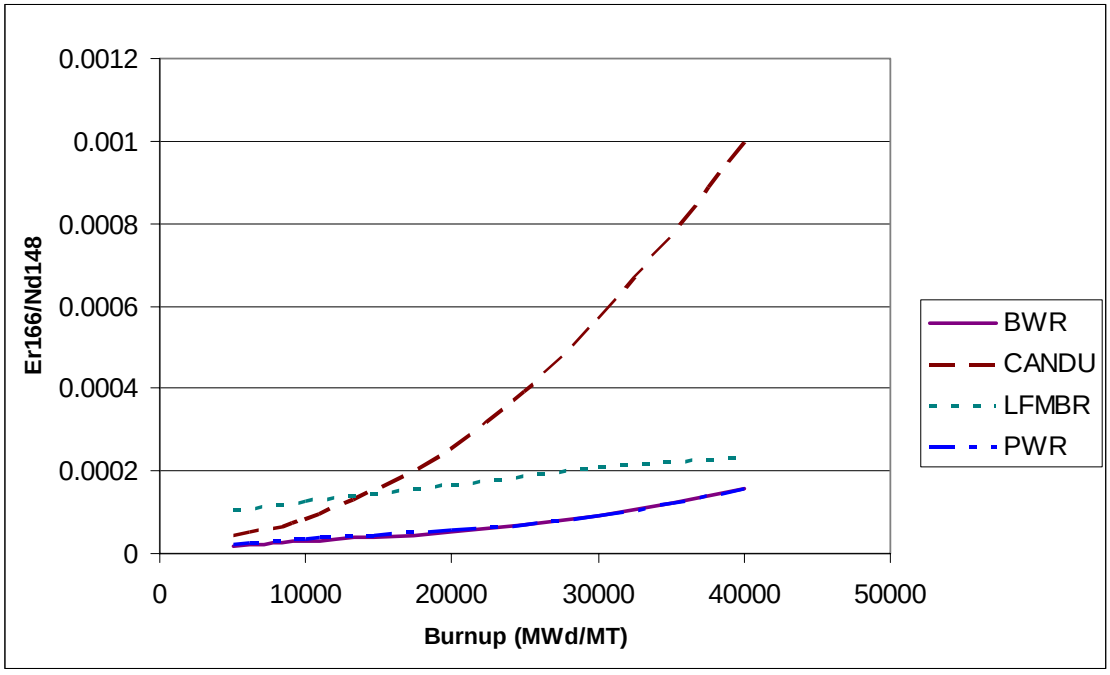

Fig. B9. Atom ratio of ^{166}Er to ^{238}U for several reactor types.

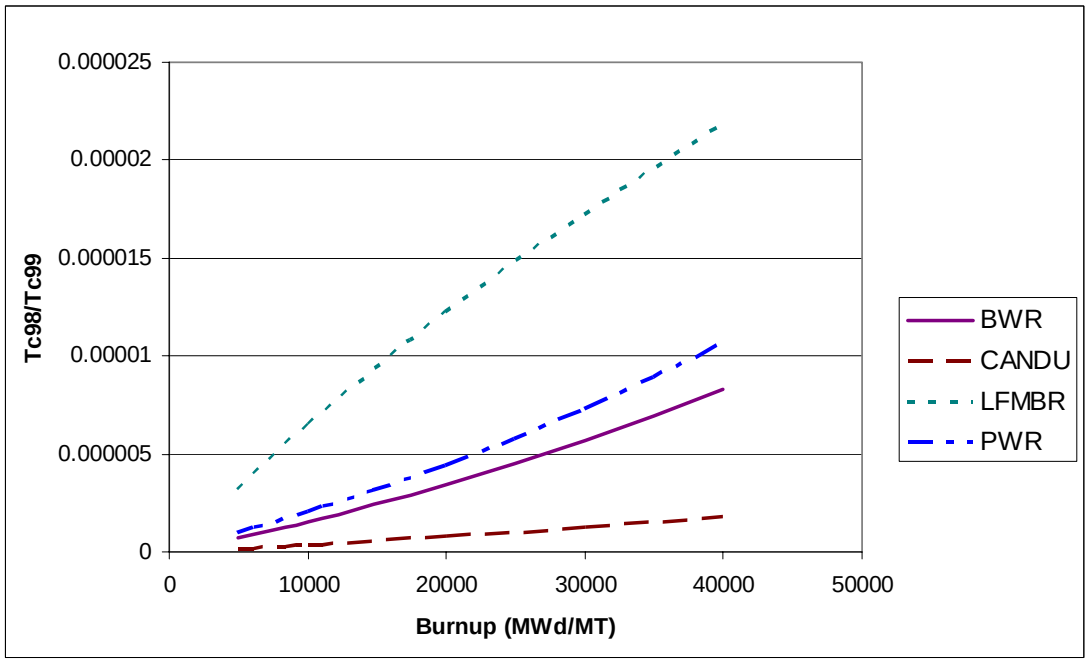

Fig. B10. Atom ratio of ^{99}Tc to ^{238}U for several reactor types.

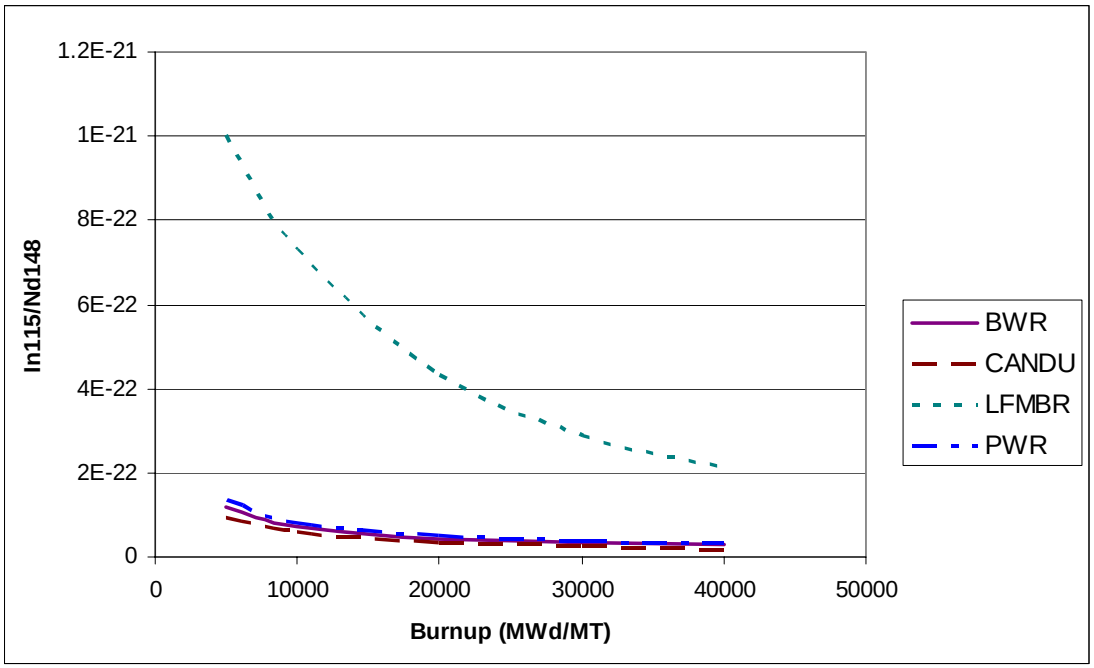

Fig. B11. Atom ratio of ^{115}In to ^{238}U for several reactor types.

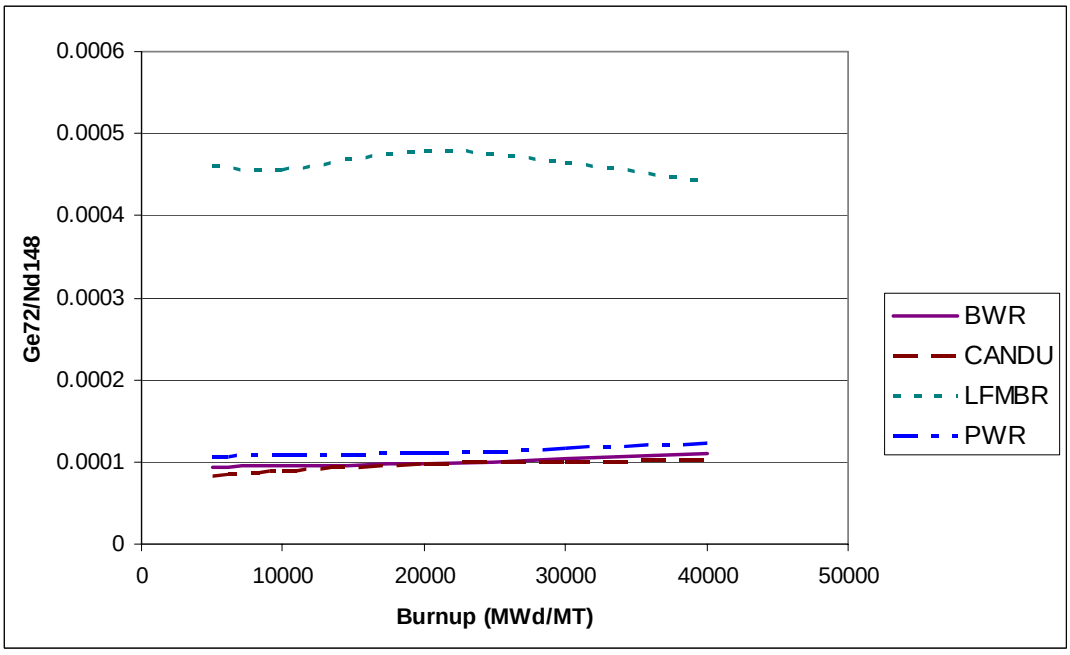

Fig. B12. Atom ratio of ^{72}Ge to ^{238}U for several reactor types.

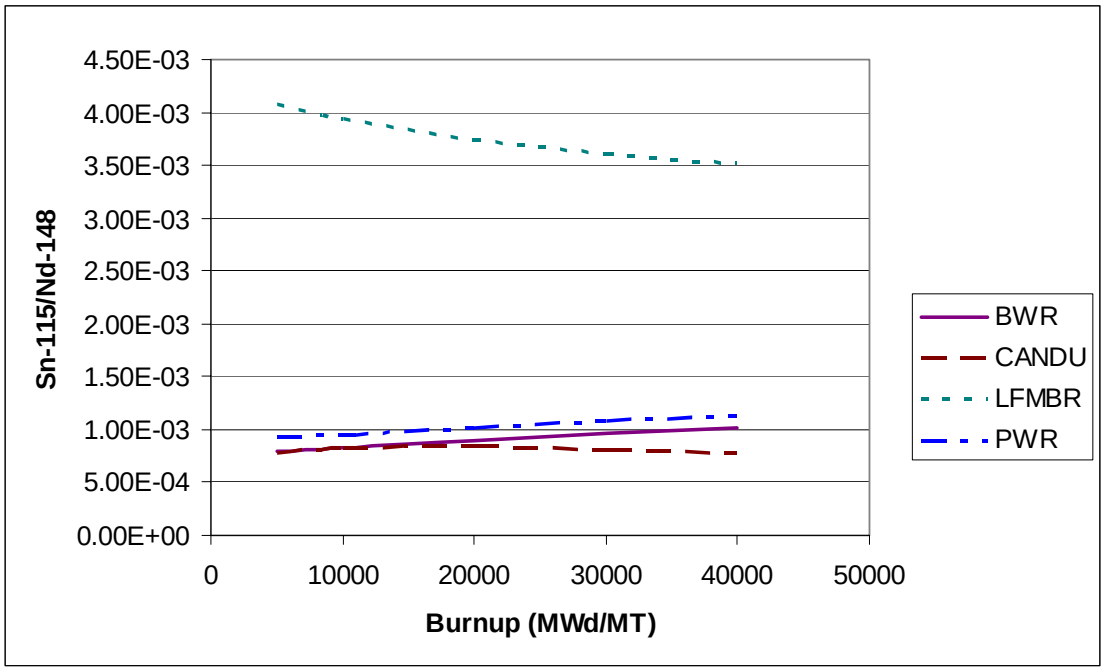

Fig. B13. Atom ratio of ^{115}Sn to ^{238}U for several reactor types.

APPENDIX D

THE NEMASYS ACCESS DATABASE

TABLE D1

Ratio Table Within the NEMASYS Access Database.

ID	Isotope Ratio	Numerator	Denominator	GroupID	Ratio	Error	Description
1	U235/U238	922350	922380	3	0	0	Enrichment
2	Pm147/U238	611470	922380	4	0	0	Age Monitor with
3	Ru106/U238	441060	922380	4	0	0	Age Monitor with
4	Sr90/U238	380900	922380	4	0	0	Age Monitor with
5	Cs137/U238	551370	922380	4	0	0	Age Monitor with
6	Ce140/U238	581400	922380	1	0	0	Burnup
7	Mo100/U238	421000	922380	1	0	0	Burnup
10	Nd148/U238	601480	922380	1	0	0	Burnup
11	Sm147/U238	621470	922380	5	0	0	History
12	Zr90/U238	400900	922380	5	0	0	History
13	In113/U238	491130	922380	5	0	0	History
14	Te125/U238	521250	922380	5	0	0	History
15	Ba134/U238	561340	922380	5	0	0	History
16	Ba137/U238	561370	922380	5	0	0	History
17	Gd157/U238	641570	922380	5	0	0	History
19	Pu240/Nd148	942400	601480	2	0	0	Reactor Type
20	Ag109/Nd148	471090	601480	2	0	0	Reactor Type
21	Eu153/Nd148	631530	601480	2	0	0	Reactor Type
22	Gd156/Nd148	641560	601480	2	0	0	Reactor Type
23	Pu239/U238	942390	922380	3	0	0	Enrichment
24	Pu240/U238	942400	922380	3	0	0	Enrichment
25	Pu241/U238	942410	922380	4	0	0	Enrichment and
26	U238/U	922380	922330, 922340,...	3	0	0	Enrichment
27	U236/U238	922360	922380	3	0	0	Enrichment
28	U234/U238	922340	922380	3	0	0	Enrichment

TABLE D2

Ratio Table Within the NEMASYS Access Database.

ID	Reactor Type	Description
1	LMFBR	
2	LMFBR2	
3	RBMK 1000	
4	PWR	
5	BWR	
6	CANDU	

APPENDIX E

AN EXAMPLE OF A NEMASYS ORIGEN SKELETON DECK

Below is a BWR ORIGEN skeleton deck.

```
-1
-1
-1
RDA       DECAY LIB    XSECT LIB                  VAR. XSECT
LIB    0    1 2 3     251 252 253    9   50   0   1       1
RDA     READ FUEL COMPOSITION INCLUDING IMPURITIES (1000 KG)
INP     1  1  -1  -1  1  1
BUP
IRP     ***1     37.5    1   1   4 2
IRP     ***2     37.5    1   1   4 0
IRP     ***3     37.5    1   1   4 0
IRP     ***4     37.5    1   1   4 0
IRP     ***5     37.5    1   1   4 0
IRP     ***6     37.5    1   1   4 0
IRP     ***7     37.5    1   1   4 0
IRP     ***8     37.5    1   1   4 0
IRP     ***9     37.5    1   1   4 0
IRP     ***10    37.5    1   1   4 0
IRP     ***11    37.5    1   1   4 0
IRP     ***12    37.5    1   1   4 0
IRP     ***13    37.5    1   1   4 0
IRP     ***14    37.5    1   1   4 0
IRP     ***15    37.5    1   1   4 0
DEC    ***16             1   1   4 0  DECAY FOR 6 MONTHS
BUP
OPTL      24*8
OPTA      2*8  5  21*8
OPTF      2*8  5  21*8
OUT       1    1     -1    0
END
2 922340  290.0  922350 **922350** 922380 **922380**      0   0.0
0
```

APPENDIX E

NEMASYS CODE

```
D:\Programs\Inverse_Model3\Form1.vb
Imports System.IO
Public Class InverseMod
Inherits System.Windows.Forms.Form
Windows Form Designer generated code
Public lbx_req_index, lbx_opt_index As Short
Public first_xy_index As Short
Public fU238C As Boolean = False
Function XYTableLoad()
'Uses the XYFile table and loads the information into the xsec_yield table
Dim i1 As Short
XYFileLoad()
For i1 = 0 To 99
If xyfiles(i1, 0) <> Nothing Then
YLoad(xyfiles(i1, 0), i1)
End If
Next
End Function
Function YLoad(ByVal filen As String, ByVal xy_index1 As Short)
'Load the yield from a Origen yield library
Dim i1, i2, i3, i4, i5, length1 As Integer
Dim TempS, Text1, fname, lname As String
Dim Temp2(300) As Single
Dim Abort1 As Boolean = False
Dim Exist As Boolean = False
Dim Start As Boolean = False
Dim found1 As Boolean = False
Dim add1, new1 As Boolean
Dim i6, i7 As Short
Dim xfile(1500, 35) As Single
Dim Temp1, index1 As Short
'Make sure the file exists
Exist = System.IO.File.Exists(filen)
If Exist = True Then
Try
FileOpen(1, filen, OpenMode.Input)
Catch
MsgBox("The file " & filen & " is locked or missing")
Abort1 = True
End Try
i1 = 0
i2 = 0
i4 = -1
i5 = 1
If Abort1 = False Then
Do Until (EOF(1))
TempS = LineInput(1)
If found1 = True Then
Dim OpenT() As String = Split(TempS, " ") 'Inputs each line as
space delimited
If OpenT(0) <> "-1" And OpenT(1) <> "-1" And OpenT(2) <> "-1"
Then
length1 = OpenT.GetLength(0)
i2 = 0
For i1 = 0 To length1 - 1
'Gets rid of the extra spaces between numbers
If OpenT(i1) <> "" Then
2 D:\Programs\Inverse_Model3\Form1.vb
'Checks to see if any number like 2.3E 01 exist and
gets rid of the space to 2.3E01
Temp1 = OpenT(i1).IndexOf("E")
If Temp1 <> -1 And Temp1 = OpenT(i1).Length - 1 Then
OpenT(i1) = OpenT(i1) & OpenT(i1 + 1)
```

```
OpenT(i1 + 1) = ""
End If
'Converts the chopped up string to an array of
floating point numbers
If OpenT(i1) <> "" Then
Try
Temp2(i2) = Convert.ToSingle(OpenT(i1))
Catch ex As Exception
MsgBox("Error reading the number " & OpenT
(i1))
End Try
i2 += 1
End If
End If
Next
'Tells it when to increment to the next isotope because the x
-sections can be over multiple lines
If Temp2(1) > 10000 Then
i4 += 1
i3 = 0
End If
For i5 = 0 To i2 - 1
xfile(i4, i3) = Temp2(i5)
i3 += 1
Next
Else
found1 = False
End If
Else
'Looks for the phrase Fission Product XSEC as the beginning for
importing numbers
Temp1 = TempS.IndexOf("ACTINIDE")
If Temp1 = -1 Then Temp1 = TempS.IndexOf("FISSION PRODUCT XSEC")
If Temp1 <> -1 Then found1 = True
End If
Loop
End If
FileClose(1)
'Because the info spans over two lines I will get rid of the library number
on the second line
For i1 = 0 To 999
For i2 = 9 To 16
xfile(i1, i2) = xfile(i1, i2 + 1)
Next
xfile(i1, 17) = 0
Next
'The File is now imported in the file xfile
'Now I want find the description of the isotope for the isotope number eg
(U238 for 922380)
Dim IsoDescript(i4 + 1), IsoNum As String
Dim TempNum As Integer
For i1 = 0 To i4
found1 = False
For i2 = 0 To 999
If found1 = False Then
TempNum = IsoInfo(i2, 2)
3 D:\Programs\Inverse_Model3\Form1.vb
If TempNum < 100 Then
IsoNum = Convert.ToString(IsoInfo(i2, 1)) & "0" & Convert.
ToString(IsoInfo(i2, 2)) _
& Convert.ToString(IsoInfo(i2, 3))
Else
IsoNum = Convert.ToString(IsoInfo(i2, 1)) & Convert.ToString
```

```
(IsoInfo(i2, 2)) _
& Convert.ToString(IsoInfo(i2, 3))
End If
If IsoNum = xfile(i1, 1) Then
IsoDescript(i1) = Convert.ToString(IsoInfo(i2, 0)) & Convert.
ToString(IsoInfo(i2, 2))
found1 = True
End If
End If
Next
Next
'Now I will transfer it to the xsec_yield table
For i1 = 0 To i4
xsec_yield(xy_index1, i1, 0) = IsoDescript(i1)
xsec_yield(xy_index1, i1, 1) = xfile(i1, 1)
xsec_yield(xy_index1, i1, 2) = xfile(i1, 5)
xsec_yield(xy_index1, i1, 3) = xfile(i1, 2)
xsec_yield(xy_index1, i1, 4) = xfile(i1, 9)
xsec_yield(xy_index1, i1, 5) = xfile(i1, 10)
xsec_yield(xy_index1, i1, 6) = xfile(i1, 11)
xsec_yield(xy_index1, i1, 7) = xfile(i1, 12)
xsec_yield(xy_index1, i1, 8) = xfile(i1, 13)
xsec_yield(xy_index1, i1, 9) = xfile(i1, 14)
xsec_yield(xy_index1, i1, 10) = xfile(i1, 15)
xsec_yield(xy_index1, i1, 11) = xfile(i1, 16)
Next
Else
MsgBox("The Origen library file at " & filen & " doesn't exist")
End If
End Function
Function XYFileLoad()
'Loads the information from the xyfiles.csv file which tell the program which
Origen libraries to use
Dim Exist As Boolean = False
Dim Abort1 As Boolean = False
Dim xyindex As Boolean = False
Dim TempS, path1, path2, path3 As String
'Checks to make sure the file exists
Exist = System.IO.File.Exists("data\xyfiles.csv")
If Exist = True Then
Try
FileOpen(1, "data\xyfiles.csv", OpenMode.Input)
Catch
'If it is currenlty opened by excel it will be locked
MsgBox("The file xyfiles.csv is locked or missing")
Abort1 = True
End Try
If Abort1 = False Then
Do Until (EOF(1))
TempS = LineInput(1)
Dim OpenT() As String = Split(TempS, ",")
If OpenT(0) = 1 Then
'Determines it the path is an absolute one or relative to the
executable
Try
If OpenT(1) = 0 Then
path1 = Application.StartupPath & "\" & OpenT(3)
4 D:\Programs\Inverse_Model3\Form1.vb
path2 = Application.StartupPath & "\" & OpenT(4)
path3 = Application.StartupPath & "\" & OpenT(5)
ElseIf OpenT(1) = 1 Then
path1 = OpenT(3)
path2 = OpenT(4)
```

```
path3 = OpenT(5)
End If
Catch ex As Exception
MsgBox("Missing information in the xyfiles.csv file")
End Try
'writes the path to the xyfiles table
xyfiles(OpenT(2), 0) = path1
xyfiles(OpenT(2), 1) = path2
xyfiles(OpenT(2), 2) = path3
If xyindex = False Then
first_xy_index = OpenT(2)
xyindex = True
End If
End If
Loop
End If
FileClose(1)
Else
MsgBox("The file xyfiles.csv is missing")
End If
End Function
Function enterratio()
Dim lbx_index, i1 As Short
'Record the data entered and pull up the data for the newly selected ratio
If num_rratio.Value <> 0 Or num_rerror.Value <> 0 Then
Req_ratios(lbx_req_index, 0) = lbx_rratio.Items.Item(lbx_req_index)
Req_ratios(lbx_req_index, 1) = num_rratio.Value
Req_ratios(lbx_req_index, 2) = num_rerror.Value
End If
'If num_rratio.Value <> 0 Or num_rerror.Value <> 0 Then
tbx_req.Clear()
For i1 = 0 To 49
If Req_ratios(i1, 1) <> Nothing Or "0" Then
tbx_req.Text += Req_ratios(i1, 0) & " = " & Req_ratios(i1, 1) & _
" +/- " & Req_ratios(i1, 2) & CR
End If
Next
'End If
Try
lbx_index = lbx_rratio.SelectedIndex
lbx_req_index = lbx_index
lbl_req.Text = lbx_rratio.SelectedItem
num_rratio.Value = Convert.ToSingle(Req_ratios(lbx_index, 1))
num_rerror.Value = Convert.ToSingle(Req_ratios(lbx_index, 2))
Catch ex As Exception
End Try
chkSatisfied()
End Function
Function chkSatisfied()
Dim e1, e2, e3, e4, e5, e6, e7, enrich1 As Boolean
Dim i1, i2, i3, length1 As Short
Dim b1, b2, a1, a2, r1 As Boolean
e1 = False
e2 = False
e3 = False
5 D:\Programs\Inverse_Model3\Form1.vb
e4 = False
e5 = False
e6 = False
e7 = False
enrich1 = False
a1 = False
a2 = False
```

```
b1 = False
b2 = False
r1 = False
'Check to see if all the necessary isotopes for enrichment are entered
For i1 = 0 To 199
If Req_ratios(i1, 0) = "U234/U238" Then e1 = True
If Req_ratios(i1, 0) = "U235/U238" Then e2 = True
If Req_ratios(i1, 0) = "U236/U238" Then e3 = True
If Req_ratios(i1, 0) = "U238/U" Then e4 = True
If Req_ratios(i1, 0) = "Pu239/U238" Then e5 = True
If Req_ratios(i1, 0) = "Pu240/U238" Then e6 = True
If Req_ratios(i1, 0) = "Pu241/U238" Then e7 = True
Next
If e1 = True And e2 = True And e3 = True And e4 = True And e5 = True And e6 =
True _
And e7 = True Then enrich1 = True
'Check to make sure at least one burnup monitor has been entered
'Pull a list from the database of all burnup monitors
length1 = DV1.Count()
Dim burnl(length1 - 1) As String
i2 = 0
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("GroupID") = 1 Then
burnl(i2) = DV1.Item(i1).Item("Isotope Ratio")
i2 += 1
End If
Next
For i1 = 0 To i2 - 1
If burnl(i1) <> Nothing Then
For i3 = 0 To 199
If burnl(i1) = Req_ratios(i3, 0) Then
b1 = True
End If
Next
End If
Next
'Check to make sure at least two age monitors have been entered
'Pull a list from the database of all age monitors
Dim agel(length1 - 1) As String
i2 = 0
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("GroupID") = 4 Then
agel(i2) = DV1.Item(i1).Item("Isotope Ratio")
i2 += 1
End If
Next
For i1 = 0 To i2 - 1
If agel(i1) <> Nothing Then
For i3 = 0 To 199
If agel(i1) = Req_ratios(i3, 0) Then
If a1 = False Then
a1 = True
6 D:\Programs\Inverse_Model3\Form1.vb
Else
a2 = True
End If
End If
Next
End If
Next
'Check to make sure at least one reactor type monitor has been entered
'Pull a list from the database of all reactor type monitors
length1 = DV1.Count()
```

```
Dim rtypel(length1 - 1) As String
i2 = 0
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("GroupID") = 2 Then
rtypel(i2) = DV1.Item(i1).Item("Isotope Ratio")
i2 += 1
End If
Next
For i1 = 0 To i2 - 1
If rtypel(i1) <> Nothing Then
For i3 = 0 To 199
If rtypel(i1) = Req_ratios(i3, 0) Then
r1 = True
End If
Next
End If
Next
'Write it to the text boxes
tbx_satc.Text = ""
tbx_urcat.Text = ""
If b1 = True Then
tbx_satc.Text += "Burnup" & CR
Else
tbx_urcat.Text += "Burnup" & CR
End If
If enrich1 = True Then
tbx_satc.Text += "Enrichment" & CR
Else
tbx_urcat.Text += "Enrichment" & CR
End If
If r1 = True Then
tbx_satc.Text += "Reactor Type" & CR
Else
tbx_urcat.Text += "Reactor Type" & CR
End If
If a2 = True Then
tbx_satc.Text += "Age" & CR
Else
tbx_urcat.Text += "Age" & CR
End If
End Function
Function Totburnup() As Single
'This function combines the burnup from all the burnup monitors and weights
'them according to their error
'burnuplist Table to temporarily hold the burnup isotopes from DV1
'column 0: Isotope Ration (name)
'column 1: Numerator
'column 2: Denominator
7 D:\Programs\Inverse_Model3\Form1.vb
'column 3: entered measured data by user
'column 4: entered measured error by user
Dim burnuplist(19, 4) As String
Dim length1, i1, i2, i3 As Short
Dim yield3, burn1, weight1 As Single
Dim found1 As Boolean = False
'Burnup for each monitor along with its weight from the user inputed error
Dim isoburn(19, 1) As Single
length1 = DV1.Count()
i2 = 0
'Pull a list from the database of all burnup monitors
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("GroupID") = 1 Then
burnuplist(i2, 0) = DV1.Item(i1).Item("Isotope Ratio")
```

```
burnuplist(i2, 1) = DV1.Item(i1).Item("Numerator")
burnuplist(i2, 2) = DV1.Item(i1).Item("Denominator")
i2 += 1
End If
Next
'Check that list against the user entered data
For i1 = 0 To i2 - 1
found1 = False
i3 = 0
Do Until found1 = True
If burnuplist(i1, 0) = Req_ratios(i3, 0) Then
burnuplist(i1, 3) = Req_ratios(i3, 1)
burnuplist(i1, 4) = Req_ratios(i3, 2)
found1 = True
End If
If i3 = 199 Then found1 = True
i3 += 1
Loop
Next
'Calculate the burnup for each of the entered ratios
i3 = 0
For i1 = 0 To i2 - 1
If burnuplist(i1, 3) <> Nothing Then
weight1 = burnuplist(i1, 4) / burnuplist(i1, 3)
yield3 = Totyield(burnuplist(i1, 1), "U235", first_xy_index)
isoburn(i3, 0) = burnup(burnuplist(i1, 0), yield3)
isoburn(i3, 1) = weight1 ' Need to fix the weight to reflex an relative
error
BurnErrors(i3, 0) = burnuplist(i1, 0)
BurnErrors(i3, 1) = isoburn(i3, 0)
i3 += 1
End If
Next
If i3 > 1 Then
'If the user didn't enter an error with the burnup monitor I will add a .01%
error
For i1 = 0 To i3 - 1
If isoburn(i1, 1) = 0 Then isoburn(i1, 1) = 0.001 'isoburn(i1, 0) * 0.
0001
Next
'Weight the burnup calculations in relaion to the measured error
Dim weight2(i3, 1) As Single
For i1 = 0 To i3 - 1
weight2(i3, 0) += isoburn(i1, 1)
Next
For i1 = 0 To i3 - 1
weight2(i1, 0) = (1 - isoburn(i1, 1) / weight2(i3, 0)) ^ 6
Next
weight2(i3, 0) = 0
8 D:\Programs\Inverse_Model3\Form1.vb
For i1 = 0 To i3 - 1
weight2(i1, 1) = isoburn(i1, 0) * weight2(i1, 0)
weight2(i3, 0) += weight2(i1, 0)
Next
weight2(i3, 1) = 0
For i1 = 0 To i3 - 1
weight2(i3, 1) += weight2(i1, 1)
Next
Totburnup = weight2(i3, 1) / weight2(i3, 0)
Else
For i1 = 0 To i2 - 1
If isoburn(i1, 0) <> Nothing Then
Totburnup = isoburn(i1, 0)
```

```
End If
Next
End If
i1 = 0
End Function
Function burnup(ByVal rname As String, ByVal Yield1 As Single) As Double
'rname is the name of the burnup isotope used (ie "Nd148/U238"), and yield1 is
it
's yield
'OEnrich2 is the enrich for Origen, used only when OIter = True
'A function that will calculate burnup
Dim Burn10, Burn11, Burn20, Burn21, Burn30, Burn31, N238_0, N238_1, N238_2 As
Double
Dim N234, N235, N236, N238, N239, N240, N241, blank As Single
Dim rindex As Short
Dim numRatio, error1 As Single
Dim found1 As Boolean = False
Find_Rat("U234/U238", N234, blank)
Find_Rat("U235/U238", N235, blank)
Find_Rat("U236/U238", N236, blank)
Find_Rat("U238/U", N238_0, blank)
Find_Rat("Pu239/U238", N239, blank)
Find_Rat("Pu240/U238", N240, blank)
Find_Rat("Pu241/U238", N241, blank)
Find_Rat(rname, numRatio, error1)
If failed = False Then
'First step
Burn10 = numRatio * Er * Na / MoU / (Yield1 / 100)
Burn11 = Burn10 * 0.00000000000016022 / 24 / 60 / 60
'First corrected NU238/NU
N238_1 = 1 / ((1 / N238_0 + N239 + N240 + N241) / (1 - Burn10 * MoU / Er /
Na))
'First corrected Burnup
Burn20 = N238_1 * Burn10
Burn21 = N238_1 * Burn11
'Second corrected NU238/NU
N238_2 = 1 / ((1 / N238_0 + N239 + N240 + N241) / (1 - Burn20 * MoU / Er /
Na))
'Second Burnup correction
Burn30 = N238_2 / N238_1 * Burn20
Burn31 = N238_2 / N238_1 * Burn21
''Test temporarily
'Burn31 = 30000
9 D:\Programs\Inverse_Model3\Form1.vb
'Burn30 = Burn31 * (24 * 60 * 60) / 0.00000000000016022
'Third corrected NU238/NU
If fU238C = False Then
NU238_NUC = 1 / ((N234 + N235 + N236 + 1) * (1 + Burn30 * MoU / (Na *
Er)) + N236 + N239 + N240 + N241)
'NU238_NUC = 1 / ((1 / N238_0 + N239 + N240 + N241) / (1 - Burn30 * MoU /
Er / Na))
If rname = "Nd148/U238" Then
fU238C = True
End If
End If
End If
burnup = Burn31
End Function
Function Totyield(ByVal isotope1 As Integer, ByVal source As String, ByVal
reactor As
Short) As Single
Dim i1, anum1, anum2, znum1, znum2, length1, index1 As Short
Dim iso3 As Integer
```

```
Dim yield1 As Single
Dim isotemp As String
Dim failed5 As Boolean = False
'Uses the source variable and finds the right index for the xsec_yield table
index1 = 0
If source = "Th232" Then index1 = 4
If source = "U233" Then index1 = 5
If source = "U235" Then index1 = 6
If source = "U238" Then index1 = 7
If source = "Pu239" Then index1 = 8
If source = "Pu241" Then index1 = 9
If source = "Cm245" Then index1 = 10
If source = "Cf252" Then index1 = 11
'Test to make sure the source is found
If index1 = 0 Then
MsgBox("Error in Total Yield calc. Wrong source")
failed5 = True
End If
'Breaks the isotope into Atomic and Z numbers
isotemp = Convert.ToString(isotope1)
anum1 = Convert.ToInt16(isotemp.Substring(2, 3))
znum1 = Convert.ToInt16(isotemp.Substring(0, 2))
'Calculate the total yield
If failed5 = False Then
yield1 = 0
For i1 = 0 To 1499
If xsec_yield(reactor, i1, 1) <> Nothing Then
anum2 = Convert.ToInt16(xsec_yield(reactor, i1, 1).Substring(2, 3))
znum2 = Convert.ToInt16(xsec_yield(reactor, i1, 1).Substring(0, 2))
iso3 = Convert.ToInt32(xsec_yield(reactor, i1, 1))
If anum1 = anum2 And znum1 >= znum2 And iso3 > 100000 Then
yield1 += Convert.ToSingle(xsec_yield(reactor, i1, index1))
End If
End If
Next
End If
Totyield = yield1
End Function
Function Enrich(ByVal Rindex1 As Short)
Dim i1, i2, Rindex2 As Short
10 D:\Programs\Inverse_Model3\Form1.vb
Dim Eo, Eo2, Eo3, E1, E2, E3 As Single
Dim N235_N238, N236_N238, N238_NU, N239_N238, N240_N238, N234_N238, N241_N238
As
Single
Dim blank, modburn As Single
Dim xfU238, xaU238, xrU238, xfPu239, xaPu239, xrPu239, xfPu240, xaPu240,
xrPu240,
xaPu241, xfPu241 As Single
Dim found1 As Boolean
'If no index is selected then it will use the x-sections from the first one
listed in the xyfile.csv
If Rindex1 = -1 Then
Rindex2 = first_xy_index
Else
Rindex2 = Rindex1
End If
'Change the burnup from MWD/MT to MeV/g
modburn = BurnupTable(0, 1) * 24 * 60 * 60 / 0.00000000000016022
'Load all of the needed cross-sections, ratios from the xsec_yield table
Find_Rat("U235/U238", N235_N238, blank)
Find_Rat("U236/U238", N236_N238, blank)
N238_NU = NU238_NUC
```

```
Find_Rat("Pu239/U238", N239_N238, blank)
Find_Rat("Pu240/U238", N240_N238, blank)
Find_Rat("U234/U238", N234_N238, blank)
Find_Rat("Pu241/U238", N241_N238, blank) 'Temp Pu241 check
found1 = False
i1 = 0
Do Until found1 = True Or i1 > 1399
If xsec_yield(Rindex2, i1, 1) = 922380 Then
xfU238 = xsec_yield(Rindex2, i1, 2)
xrU238 = xsec_yield(Rindex2, i1, 3)
xaU238 = xfU238 + xrU238
found1 = True
End If
i1 += 1
Loop
found1 = False
i1 = 0
Do Until found1 = True Or i1 > 1399
If xsec_yield(Rindex2, i1, 1) = 942390 Then
xfPu239 = xsec_yield(Rindex2, i1, 2)
xaPu239 = xfPu239 + xsec_yield(Rindex2, i1, 3)
xrPu239 = xsec_yield(Rindex2, i1, 3)
found1 = True
End If
i1 += 1
Loop
found1 = False
i1 = 0
Do Until found1 = True Or i1 > 1399
If xsec_yield(Rindex2, i1, 1) = 942400 Then
xfPu240 = xsec_yield(Rindex2, i1, 2)
xaPu240 = xfPu240 + xsec_yield(Rindex2, i1, 3)
xrPu240 = xsec_yield(Rindex2, i1, 3) 'Temp check Pu241
found1 = True
End If
i1 += 1
Loop
'Start Temp check Pu241
found1 = False
i1 = 0
Do Until found1 = True Or i1 > 1399
If xsec_yield(Rindex2, i1, 1) = 942410 Then
11 D:\Programs\Inverse_Model3\Form1.vb
xfPu241 = xsec_yield(Rindex2, i1, 2)
xaPu241 = xfPu240 + xsec_yield(Rindex2, i1, 3)
found1 = True
End If
i1 += 1
Loop
'End temp check Pu241
'First Step
For i1 = 0 To 1
Eo = (N235_N238 + N236_N238) * N238_NU + (modburn * MoU / (Na * Er))
MoU = 235 * Eo + 238 * (1 - Eo)
Eo = (N235_N238 + N236_N238) * N238_NU + (modburn * MoU / (Na * Er))
Eo2 = Eo
Eo3 = -(N238_NU * Na * Er * xaU238 * xaPu239 * xaPu240 * N235_N238 + _
N238_NU * Na * Er * xaU238 * xaPu239 * xaPu240 * N236_N238 + MoU * _
modburn * xaU238 * xaPu239 * xaPu240 - xfU238 * Na * Er * xaPu239 * _
xaPu240 + xfU238 * Na * Er * xaPu239 * xaPu240 * N238_NU + xfU238 * Na * _
Er * xaPu239 * xaPu240 * N234_N238 * N238_NU + xfPu239 * Na * Er * _
xaPu240 * N239_N238 * N238_NU * xaU238 - xfPu239 * Na * Er * xaPu240 * _
xrU238 + xfPu239 * Na * Er * xaPu240 * xrU238 * N238_NU + xfPu239 * Na * _
```

```
Er * xaPu240 * xrU238 * N234_N238 * N238_NU + xfPu240 * Na * Er * N240_N238 *
_
N238_NU * xaPu239 * xaU238 + xfPu240 * Na * Er * xrPu239 * N239_N238 * _
N238_NU * xaU238 - xfPu240 * Na * Er * xrPu239 * xrU238 + xfPu240 * Na * _
Er * xrPu239 * xrU238 * N238_NU + xfPu240 * Na * Er * xrPu239 * xrU238 * _
N234_N238 * N238_NU) / Na / Er / (-xaPu240 * xaPu239 * xaU238 + xfU238 * _
xaPu240 * xaPu239 + xfPu239 * xrU238 * xaPu240 + xfPu240 * xrPu239 * xrU238)
'Eo3 = -(N238_NU * Na * Er * xaU238 * xaPu239 * xaPu240 * xaPu241 * N235_N238
+ N238_NU * Na * Er * xaU238 * xaPu239 * xaPu240 * xaPu241 * N236_N238 + MoU *
modburn * xaU238 * xaPu239 * xaPu240 * xaPu241 - xfU238 * Na * Er * xaPu239 *
xaPu240
* xaPu241 + xfU238 * Na * Er * xaPu239 * xaPu240 * xaPu241 * N238_NU + xfU238 *
Na *
Er * xaPu239 * xaPu240 * xaPu241 * N234_N238 * N238_NU + xfPu239 * Na * Er *
xaPu240
* xaPu241 * N239_N238 * N238_NU * xaU238 - xfPu239 * Na * Er * xaPu240 *
xaPu241 *
xrU238 + xfPu239 * Na * Er * xaPu240 * xaPu241 * xrU238 * N238_NU + xfPu239 *
Na * Er
* xaPu240 * xaPu241 * xrU238 * N234_N238 * N238_NU + xfPu240 * Na * Er *
xaPu241 *
N240_N238 * N238_NU * xaPu239 * xaU238 + xfPu240 * Na * Er * xaPu241 * xrPu239
*
N239_N238 * N238_NU * xaU238 - xfPu240 * Na * Er * xaPu241 * xrPu239 * xrU238 +
xfPu240 * Na * Er * xaPu241 * xrPu239 * xrU238 * N238_NU + xfPu240 * Na * Er *
xaPu241 * xrPu239 * xrU238 * N234_N238 * N238_NU + xfPu241 * Na * Er *
N241_N238 *
N238_NU * xaPu240 * xaPu239 * xaU238 + xfPu241 * Na * Er * xrPu240 * N240_N238
*
N238_NU * xaPu239 * xaU238 + xfPu241 * Na * Er * xrPu240 * xrPu239 * N239_N238
*
N238_NU * xaU238 - xfPu241 * Na * Er * xrPu240 * xrPu239 * xrU238 + xfPu241 *
Na * Er
* xrPu240 * xrPu239 * xrU238 * N238_NU) / Na / Er / (-xaPu241 * xaPu240 *
xaPu239 *
xaU238 + xfU238 * xaPu241 * xaPu240 * xaPu239 + xfPu239 * xrU238 * xaPu241 *
xaPu240
+ xfPu240 * xrPu239 * xrU238 * xaPu241 + xfPu241 * xrPu240 * xrPu239 * xrU238)
'If Rindex1 = 4 Then
' N238_NU = 1 - Eo3
'End If
Next
If Rindex1 = -1 Then
For i1 = 1 To 99
EnrichTable(i1, 0) = Eo3 * 100
Next
Else
EnrichTable(Rindex1, 0) = Eo3 * 100
End If
12 D:\Programs\Inverse_Model3\Form1.vb
End Function
Function genreport()
Dim i1 As Short
BurnupTable(0, 1) = Totburnup()
For i1 = 0 To 99
If xyfiles(i1, 0) <> Nothing Then
Enrich(i1)
End If
Next
ReactType(False)
fU238C = False
GetFuelAge(False)
End Function
```

```
Function testload()
'This will load a tape6.out file from Origen with only one column of output
into
'the program input to save time testing various possibilities.
Dim i1, i2, i3, i4, length1, match As Short
Dim found1 As Boolean
Dim sum1 As Single
length1 = DV1.Count()
Dim TempRatios(length1 - 1, 3), TempS As String
'Copy the Tape6.out file to where the ParseTape6 function will expect it to be
Open1.Filter = "OUT files (*.out)|*.out|All files (*.*)|*.*"
Open1.InitialDirectory = Application.StartupPath & "\saved"
Open1.ShowDialog()
If Open1.FileName <> "" Then
TempS = Application.StartupPath & "\data\ORIGEN\TAPE6.OUT"
match = StrComp(TempS, Open1.FileName, CompareMethod.Text)
If match <> 0 Then
File.Delete(Application.StartupPath & "\data\ORIGEN\TAPE6.OUT")
File.Copy(Open1.FileName, Application.StartupPath & "\data\ORIGEN\TAPE6.
OUT")
End If
ParseTape6()
'Load all of the ratios in the database
length1 = DV1.Count()
For i1 = 0 To length1 - 1
TempRatios(i1, 0) = DV1.Item(i1).Item("Isotope Ratio")
TempRatios(i1, 1) = DV1.Item(i1).Item("Numerator")
TempRatios(i1, 2) = DV1.Item(i1).Item("Denominator")
Next
'Match the ratios in the access database to the tape6.out file and calc the
ratio
'First match the numerator
For i1 = 0 To length1 - 1
found1 = False
i2 = 0
Do Until found1 = True Or i2 > 1400
If TempRatios(i1, 1) = TTape6(i2, 0) Then
TempRatios(i1, 3) = TTape6(i2, 1)
found1 = True
End If
i2 += 1
13 D:\Programs\Inverse_Model3\Form1.vb
Loop
Next
'Next match the denominator
For i1 = 0 To length1 - 1
found1 = False
i2 = 0
Do Until found1 = True Or i2 > 1400
'Check to see if the denominator contains more than one isotope
If TempRatios(i1, 2).IndexOf(",") > 0 Then
Dim OpenT() As String = Split(TempRatios(i1, 2), ",")
sum1 = 0
For i3 = 0 To OpenT.GetLength(0) - 1
found1 = False
i4 = 0
Do Until found1 = True Or i4 > 1400
If OpenT(i3) = TTape6(i4, 0) Then
Try
sum1 += Convert.ToSingle(TTape6(i4, 1))
Catch ex As Exception
End Try
found1 = True
```

```
End If
i4 += 1
Loop
Next
TempRatios(i1, 3) = Convert.ToSingle(TempRatios(i1, 3)) / sum1
Else
'Contains only one isotope in the denominator
If TempRatios(i1, 2) = TTape6(i2, 0) Then
Try
TempRatios(i1, 3) = Convert.ToSingle(TempRatios(i1, 3)) /
Convert.ToSingle(TTape6(i2, 1))
Catch ex As Exception
End Try
found1 = True
End If
i2 += 1
End If
Loop
Next
'Enter the calc ratios into the req_ratios table
For i1 = 0 To length1 - 1
Req_ratios(i1, 0) = TempRatios(i1, 0)
Req_ratios(i1, 1) = TempRatios(i1, 3)
Req_ratios(i1, 2) = 0
Next
num_rratio.Value = 0
num_rerror.Value = 0
enterratio()
End If
End Function
Function Opensaved()
Dim TempS, TempS2 As String
Dim i1, i2, i3, space1, end1, length1 As Integer
Dim isotopes(499, 2) As String
Req_ratios.Clear(Req_ratios, 0, 600)
Open1.Filter = "inp files (*.inp)|*.inp|All files (*.*)|*.*"
Open1.InitialDirectory = Application.StartupPath & "\saved"
Open1.FileName = ""
Open1.ShowDialog()
14 D:\Programs\Inverse_Model3\Form1.vb
If Open1.FileName <> "" Then
FileOpen(1, Open1.FileName, OpenMode.Input)
i1 = 0
Do Until (EOF(1))
TempS = LineInput(1)
Dim OpenT() As String = Split(TempS, "=")
isotopes(i1, 0) = OpenT(0).Trim
TempS = OpenT(1).Trim
space1 = TempS.IndexOf(" ")
TempS2 = TempS.Substring(0, space1)
isotopes(i1, 1) = TempS2
space1 = TempS.IndexOf("-")
end1 = TempS.Length
TempS2 = TempS.Substring(space1 + 1, end1 - space1 - 1)
isotopes(i1, 2) = TempS2.Trim
i1 += 1
Loop
FileClose(1)
length1 = DV1.Count
For i2 = 0 To i1 - 1
For i3 = 0 To length1 - 1
If isotopes(i2, 0) = DV1.Item(i3).Item("Isotope Ratio") Then
Req_ratios(i3, 0) = isotopes(i2, 0)
```

```
Req_ratios(i3, 1) = isotopes(i2, 1)
If isotopes(i2, 2).Trim <> Nothing Then
Req_ratios(i3, 2) = isotopes(i2, 2)
End If
End If
Next
Next
num_rratio.Value = 0
num_rerror.Value = 0
enterratio()
End If
End Function
Function save_ratios()
Dim txtfile As String
txtfile = tbx_req.Text
Save1.Filter = "inp files (*.inp)|*.inp|All files (*.*)|*.*"
Save1.InitialDirectory = Application.StartupPath
Save1.FileName = ""
Save1.ShowDialog()
If Save1.FileName <> "" Then
FileOpen(1, Save1.FileName, OpenMode.Output)
Print(1, txtfile)
FileClose(1)
End If
End Function
Function cleartables()
MoU = 237.9
NU238_NUC = Nothing
BurnupTable.Clear(BurnupTable, 0, 22)
EnrichTable.Clear(EnrichTable, 0, 200)
RTypeConf.Clear(RTypeConf, 0, 200)
Report1.Clear(Report1, 0, 800)
IsoAge.Clear(IsoAge, 0, 10000)
FuelAge.Clear(FuelAge, 0, 100)
RTypeErrors.Clear(RTypeErrors, 0, 30000)
BurnErrors.Clear(BurnErrors, 0, 200)
AgeErrors.Clear(AgeErrors, 0, 400)
IDSelect.Clear(IDSelect, 0, 6)
End Function
15 D:\Programs\Inverse_Model3\Form1.vb
Private Sub ToolBar1_ButtonClick(ByVal sender As System.Object, ByVal e As
System.
Windows.Forms.ToolBarButtonClickEventArgs) Handles ToolBar1.ButtonClick
Dim i1 As Short
Dim run1 As Boolean = False
Select Case ToolBar1.Buttons.IndexOf(e.Button)
Case 0
Req_ratios.Clear(Req_ratios, 0, 600)
num_rratio.Value = 0
num_rerror.Value = 0
enterratio()
Case 1
Opensaved()
Case 2
save_ratios()
Case 3
If tbx_urcat.Text = "" Then
Dim frminput As New Report
genreport()
frminput.ShowDialog()
cleartables()
Else
MsgBox("First satisfy all of the required categories")
```

```
End If
End Select
End Sub
Private Sub MenuItem7_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem7.Click
Close()
End Sub
Private Sub InverseMod_Load(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MyBase.Load
CR = Chr(13) & Chr(10)
Tab1 = (Chr(9))
'Create a OleDB Connection
Dim conn As New System.Data.OleDb.OleDbConnection
conn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data source=" & _
"data\isotope_ratios.mdb"
Dim Fail1 As Boolean = False
Try
conn.Open()
Catch ex As Exception
MessageBox.Show("Failed to connect to the data source")
Fail1 = True
End Try
'Create a OleDB Data Adapter and load it to a DataSet
If Fail1 = False Then
Dim Adapter1 As New OleDb.OleDbDataAdapter
Dim Adapter2 As New OleDb.OleDbDataAdapter
Dim Adapter3 As New OleDb.OleDbDataAdapter
Dim Query1 As String = "SELECT Ratios.[Isotope Ratio], Ratios.GroupID, " & _
"Ratios.Ratio, Ratios.Numerator, Ratios.Denominator, Ratios.Error, " & _
"Ratios.Description FROM Ratios ORDER BY Ratios.[Isotope Ratio]"
Dim Query2 As String = "SELECT Categories.Category, Categories.Description,"
& _
" Categories.GroupID, Categories.Required, Categories.Satisfied " & _
"FROM(Categories) ORDER BY Categories.Category"
Dim Query3 As String = "SELECT Reactors.ID, Reactors.[Reactor Type]" & _
" FROM(Reactors) ORDER BY Reactors.ID"
16 D:\Programs\Inverse_Model3\Form1.vb
Adapter1.SelectCommand = New OleDb.OleDbCommand(Query1, conn)
Adapter2.SelectCommand = New OleDb.OleDbCommand(Query2, conn)
Adapter3.SelectCommand = New OleDb.OleDbCommand(Query3, conn)
Adapter1.Fill(DS1, "Isotope")
Adapter2.Fill(DS2, "Category")
Adapter3.Fill(DS3, "Reactor")
conn.Close()
End If
Fail1 = False
'Load the DataViews from the DataSets
Dim length1, i1 As Short
DV1 = DS1.Tables("Isotope").DefaultView
DV2 = DS2.Tables("Category").DefaultView
DV3 = DS3.Tables("Reactor").DefaultView
'Load the list box items
length1 = DV2.Count()
lbx_cat.Items.Clear()
lbx_rratio.Items.Clear()
tbx_urcat.Text = ""
tbx_uocat.Text = ""
For i1 = 0 To length1 - 1
lbx_cat.Items.Add(DV2.Item(i1).Item("Category"))
If DV2.Item(i1).Item("Required") = True Then
tbx_urcat.Text += DV2.Item(i1).Item("Category") & CR
Else
tbx_uocat.Text += DV2.Item(i1).Item("Category") & CR
```

```
End If
Next
length1 = DV1.Count()
For i1 = 0 To length1 - 1
lbx_rratio.Items.Add(DV1.Item(i1).Item("Isotope Ratio"))
Next
'Load the isotope information from the file elements.csv in the data folder
Dim filen, TempS As String
Dim Abort1 As Boolean = False
Dim i2 As Short
filen = "Data\elements.csv"
Try
FileOpen(1, filen, OpenMode.Input)
Catch
MsgBox("Data\elements.csv file is missing or locked")
Abort1 = True
End Try
If Abort1 = False Then
i1 = 0
Do Until (EOF(1))
TempS = LineInput(1)
If i1 > 1 Then
Dim OpenT() As String = Split(TempS, ",")
For i2 = 0 To 5
IsoInfo(i1 - 2, i2) = OpenT(i2)
Next
End If
i1 += 1
Loop
End If
FileClose(1)
'Load the cross-section and yield values for each rector
XYTableLoad()
End Sub
17 D:\Programs\Inverse_Model3\Form1.vb
Private Sub lbx_rratio_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As
System.EventArgs) Handles lbx_rratio.SelectedIndexChanged
enterratio()
End Sub
Private Sub rbn_isotope_CheckedChanged(ByVal sender As System.Object, ByVal e
As
System.EventArgs) Handles rbn_isotope.CheckedChanged
tbx_descript.Text = ""
If rbn_isotope.Checked = True Then
'Load the list box items for the categories
lbl_cat.Text = "Isotope Description"
Dim length1, i1 As Short
length1 = DV1.Count()
lbx_cat.Items.Clear()
For i1 = 0 To length1 - 1
lbx_cat.Items.Add(DV1.Item(i1).Item("Isotope Ratio"))
Next
Else
'Load the list box items for the categories
lbl_cat.Text = "Category Description"
Dim length1, i1 As Short
length1 = DV2.Count()
lbx_cat.Items.Clear()
For i1 = 0 To length1 - 1
lbx_cat.Items.Add(DV2.Item(i1).Item("Category"))
Next
End If
```

```
End Sub
Private Sub lbx_cat_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As
System.EventArgs) Handles lbx_cat.SelectedIndexChanged
Dim index1, i1, length1 As Short
index1 = lbx_cat.SelectedIndex
'If isotope option is checked do the following
If rbn_isotope.Checked = True Then
tbx_descript.Text = DV1.Item(index1).Item("Description")
tbx_descript.Text += CR & CR & "This Isotope is in the category "
length1 = DV2.Count()
For i1 = 0 To length1 - 1
If DV1.Item(index1).Item("GroupID") = DV2.Item(i1).Item("GroupID") Then
tbx_descript.Text += CR & DV2.Item(i1).Item("Category")
End If
Next
'If the category option is checked do the following
Else
tbx_descript.Text = DV2.Item(index1).Item("Description") & CR & CR
tbx_descript.Text += "Isotopes in this category"
length1 = DV1.Count()
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("GroupID") = DV2.Item(index1).Item("GroupID") Then
tbx_descript.Text += CR & DV1.Item(i1).Item("Isotope Ratio")
End If
Next
End If
End Sub
Private Sub btn_EnterR_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btn_EnterR.Click
If lbx_rratio.SelectedIndex >= 0 Then enterratio()
End Sub
Private Sub btn_DeleteR_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btn_DeleteR.Click
Dim index1 As Short
18 D:\Programs\Inverse_Model3\Form1.vb
num_rratio.Value = 0.0
num_rerror.Value = 0.0
index1 = lbx_rratio.SelectedIndex
Req_ratios(index1, 0) = Nothing
Req_ratios(index1, 1) = Nothing
Req_ratios(index1, 2) = Nothing
enterratio()
End Sub
Private Sub btn_Test_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles btn_Test.Click
OrigenPrep(num_RID.Value, num_burn.Value, num_e.Value, num_temp.Value * 365.24)
RunOrigen()
File.Copy(Application.StartupPath & "\data\Origen\TAPE6.OUT", Application.
StartupPath & "\saved\TAPE6.OUT", True)
End Sub
Private Sub MenuItem3_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem3.Click
Opensaved()
End Sub
Private Sub MenuItem4_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem4.Click
save_ratios()
End Sub
Private Sub MenuItem2_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem2.Click
Req_ratios.Clear(Req_ratios, 0, 600)
enterratio()
```

```
End Sub
Private Sub MenuItem13_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem13.Click
testload()
End Sub
Private Sub MenuItem8_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem8.Click
Dim about1 As New AboutM
about1.ShowDialog()
End Sub
Private Sub MenuItem10_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles MenuItem10.Click
If pnl_origen.Visible = True Then
pnl_origen.Visible = False
Else
pnl_origen.Visible = True
End If
End Sub
End Class


D:\Programs\Inverse_Model3\Module1.vb
Module Module1
'Version
Public version1 As Single = 0.902
Public reldate As String = "March 5, 2005"
Public Req_ratios(199, 2) As String 'The ratios inputed by the user ratios
Public CR As String 'Carriage Return
Public Tab1 As String 'Tab
Public DS1, DS2, DS3 As New DataSet
Public DV1, DV2, DV3 As New DataView
Public Er As Single = 200 'Average Enery per fission 200MeV/fission
Public Na As Single = 6.02214E+23 'Avogadro's number
Public MoU As Single = 237.9 ' ???
Public NU238_NUC 'U238/U corrected for burnup
Public BurnupTable(10, 1) As Single 'Will keep track of the Burnup for up to 10
reactor types
'Will keep track of the Enrichment for each reactor type
'Index matchs the reactor type ID
'0 Column: Enrichment
'1 Column: Error
Public EnrichTable(99, 1) As Single
'Keeps track of how close it matches the reactor type
'0 Column: Reactor ID
'1 Column: Confidence
Public RTypeConf(99, 1) As Single
'This is the table used to fill in the report
'0 reactor ID
'1 Column is the reactor type
'2 Column is the age
'3 Column is the Conifidence (error)
Public Report1(99, 7) As String
'First dimen reactor type ID
'Second dimen is the isotopes, it can handle up to 1,500 isotopes
'Third dimen holds the cross-section and yield data
'3rd dimen: 0=isotope name (U238) 1=isotope number(601480) ,x-section,
2=fission 3=
radiative capture ,Yields, 4=Th232 5=U233 6=U235 7=U238 8=Pu239 9=Pu241
10=Cm245 11=
Cf252)
Public xsec_yield(99, 1499, 11) As String
'List of information of each isotope imported from the file elements.csv
Public IsoInfo(999, 5)
```

```
'1st column: Location of the Origen Origen x-sections and yields for each
reactor
type
'2nd column: Location of the skeleton tape5.inp file
'3rd column: Location of the tape9.inp file
'The index matches the ID for each reactor type
Public xyfiles(99, 2) As String
'Temporary file to hold the moles of each isotope from the Origen tape6.out
file
Public TTape6(1499, 1) As String
2 D:\Programs\Inverse_Model3\Module1.vb
'Temporary file to hold the age isotopes so that Origen doesn't have to run
again
'after running it to find the reactor type
'First dimen. reactor type ID
'Second dimen. can handle up to 20 age isotopes
'Third dimen. 0 = isotope name , 1 = numerator (eg 922380) , 2 = denominator
(eg
922380)
'Third dimen. 3 = Origen calculated value, 4 = propagrated error
Public IsoAge(99, 19, 4)
'Age for each reactor, the index matches the reactor ID
Public FuelAge(99)
'Records the errors on the reactor type isotopes for the ratios report
'First Dimen matches the reactor ID
'Second Dimen can hold 100 reactor type isotopes
'Third Dimen: 0: Isotope Name, 1: Enter Ratio, 2: Calculated Ratio
Public RTypeErrors(99, 99, 2) As String
'Records the errors on the burnup monitors for the ratios report
'First Dimen can hold 100 burnup monitors
'Third Dimen: 0: Isotope Name, 1: Calculated Burnup
Public BurnErrors(99, 1) As String
'Records the errors on the age monitors for the ratios report
'First Dimen matches the reactor ID
'Second Dimen can hold 19 age monitors
'Third Dimen: 0: Isotope Name, 1: Calculated Age
Public AgeErrors(99, 1, 1) As String
'Tells the ratio window what reactor ID was selected
Public IDSelect(5) As Short
'Origen Iteration report
'The first row is the enrichment everything else is burnup
'The First column is the isotope ratio name
'The second column is the value
Public OIReport(99, 1) As String
'Keeps track of any errors that occur, so the programs exits instead of
continuing to
try an run
Public failed As Boolean = False
'A temporary table to keep track of the burnup after you run an origen
iteration
'1st Dimen = Reactor index
'2nd Dimen = can handle 99 Burnup isotopes
'3rd Dimen: 0 Column = monitor name, 1 Column = burnup value
Public IBurns(99, 99, 1) As String
Function RunOrigen()
Dim Origenlocation As String
Dim Exist As Boolean = False
Dim Exist2 As Boolean = False
Dim i1 As Integer
Dim drive1, argue1 As String
'Create bat file to run origen with the correct run path
drive1 = Application.StartupPath
drive1 = drive1.Substring(0, 2)
```

```
argue1 = drive1 & " && cd " & Application.StartupPath & "\data\Origen" & " &&
origen.exe && exit"
FileOpen(1, Application.StartupPath & "\data\Origen\runorigen.bat", OpenMode.
Output)
Print(1, argue1)
FileClose(1)
3 D:\Programs\Inverse_Model3\Module1.vb
'Run the origen executable
System.IO.File.Delete(Application.StartupPath & "\data\Origen\tape6.out")
Origenlocation = Application.StartupPath & "\data\Origen\runorigen.bat"
Exist = System.IO.File.Exists(Origenlocation)
If Exist = True Then
Dim myProcess As Process = New Process
myProcess.StartInfo.WindowStyle = ProcessWindowStyle.Hidden
myProcess.StartInfo.CreateNoWindow = True
'myProcess.Start(Application.StartupPath & "\data\Origen\runorigen.bat")
Shell(Application.StartupPath & "\data\Origen\runorigen.bat", AppWinStyle.
Hide, True, -1)
Exist2 = System.IO.File.Exists(Application.StartupPath & "\data\Origen\tape6.
out")
'Make the process wait until origen finishes running
i1 = 0
Do Until Exist2 = True
Exist2 = System.IO.File.Exists(Application.StartupPath & "\data\Origen\
tape6.out")
If i1 > 10000000 Then
Exist2 = True
MsgBox("Error running origen: can't find tape6.out")
End If
i1 += 1
Loop
Else
MsgBox("Can't find Origen.exe")
End If
End Function
Function OrigenPrep(ByVal Rindex As Short, ByVal burn1 As Single, ByVal enrich1
As
Single, ByVal decay As Single)
'Find the tape5.inp and tape9.inp files for the reactor type and copy into the
correct place
'And modify the tape5.inp according to the burnup and enrichment thus known
Dim PathT5, PathT9, TempS, TempT5 As String
Dim filechk1, filechk2, found1 As Boolean
Dim Burn5, Days(15), eU235, eU238, enrich2 As Single
Dim index1, length1, i1, i2, index2 As Short
Dim finished, found2 As Boolean
PathT5 = xyfiles(Rindex, 1)
PathT9 = xyfiles(Rindex, 2)
filechk1 = System.IO.File.Exists(PathT5)
filechk2 = System.IO.File.Exists(PathT9)
If filechk1 And filechk2 = True Then
finished = False
Do Until finished = True
Try
System.IO.File.Delete(Application.StartupPath & "\data\Origen\TAPE5.
INP")
System.IO.File.Delete(Application.StartupPath & "\data\Origen\TAPE9.
INP")
finished = True
Catch ex As Exception
End Try
Loop
System.IO.File.Copy(PathT5, Application.StartupPath & "\data\Origen\TAPE5.INP
```

```
")
System.IO.File.Copy(PathT9, Application.StartupPath & "\data\Origen\TAPE9.INP
")
FileOpen(1, Application.StartupPath & "\data\Origen\TAPE5.INP", OpenMode.
Input)
4 D:\Programs\Inverse_Model3\Module1.vb
'Find the burnup in the Tape5.inp file
found1 = False
Burn5 = 0
Do Until (EOF(1))
TempS = LineInput(1)
index1 = TempS.IndexOf("***1")
If index1 > 0 Then
Dim OpenT() As String = Split(TempS, " ")
length1 = OpenT.GetLength(0)
For i1 = 0 To length1 - 1
If OpenT(i1) = "***1" Then
For i2 = i1 + 1 To length1 - 1
If OpenT(i2) <> "" And found1 = False Then
found1 = True
Burn5 = Convert.ToSingle(OpenT(i2))
End If
Next
i2 = 0
End If
Next
End If
Loop
If found1 = False Or Burn5 = 0 Then
MsgBox("Error with skeleton TAPE5.INP file")
End If
'Calculate the days for each step
Days(0) = Math.Round((burn1 * 0.0666667) / Burn5, 1)
Days(1) = Math.Round((burn1 * 0.1333333) / Burn5, 1)
Days(2) = Math.Round((burn1 * 0.2) / Burn5, 1)
Days(3) = Math.Round((burn1 * 0.2666667) / Burn5, 1)
Days(4) = Math.Round((burn1 * 0.3333333) / Burn5, 1)
Days(5) = Math.Round((burn1 * 0.4) / Burn5, 1)
Days(6) = Math.Round((burn1 * 0.4666667) / Burn5, 1)
Days(7) = Math.Round((burn1 * 0.5333333) / Burn5, 1)
Days(8) = Math.Round((burn1 * 0.6) / Burn5, 1)
Days(9) = Math.Round((burn1 * 0.6666667) / Burn5, 1)
Days(10) = Math.Round((burn1 * 0.733333) / Burn5, 1)
Days(11) = Math.Round((burn1 * 0.8) / Burn5, 1)
Days(12) = Math.Round((burn1 * 0.8666667) / Burn5, 1)
Days(13) = Math.Round((burn1 * 0.9333333) / Burn5, 1)
Days(14) = Math.Round((burn1) / Burn5, 1)
Days(15) = Days(14) + Math.Round(decay, 1)
'Calculate the enrichment in atom percent
enrich2 = Math.Abs(enrich1) / 100
'eU235 = Math.Abs(enrich1) * 10000
'eU238 = 1000000 * (1 - Math.Abs(enrich1) / 100)
'Correct for the change from atom to weight percent enrichment
eU235 = -1000000.0 * 235 * (enrich2 / (3 * enrich2 - 238))
eU238 = 1000000.0 * 238 * ((enrich2 - 1) / (enrich2 - 238))
'Add the days to the Tape5.inp file and enrichment.
FileClose(1)
FileOpen(1, Application.StartupPath & "\data\Origen\TAPE5.INP", OpenMode.
Input)
TempT5 = ""
found2 = False
Do Until (EOF(1))
TempS = LineInput(1)
```

```
index2 = TempS.IndexOf("***1")
If index2 > 0 Then
found2 = True
End If
TempS = TempS.Replace("***1 ", Days(0) & " ")
TempS = TempS.Replace("***2", Days(1))
5 D:\Programs\Inverse_Model3\Module1.vb
TempS = TempS.Replace("***3", Days(2))
TempS = TempS.Replace("***4", Days(3))
TempS = TempS.Replace("***5", Days(4))
TempS = TempS.Replace("***6", Days(5))
TempS = TempS.Replace("***7", Days(6))
TempS = TempS.Replace("***8", Days(7))
TempS = TempS.Replace("***9", Days(8))
TempS = TempS.Replace("***10", Days(9))
TempS = TempS.Replace("***11", Days(10))
TempS = TempS.Replace("***12", Days(11))
TempS = TempS.Replace("***13", Days(12))
TempS = TempS.Replace("***14", Days(13))
TempS = TempS.Replace("***15", Days(14))
TempS = TempS.Replace("***16", Days(15))
TempS = TempS.Replace("**922350**", eU235)
TempS = TempS.Replace("**922380**", eU238)
TempT5 += TempS & CR
Loop
If found2 = False Then
MsgBox("Problem with skeleton tape5.inp file")
End If
FileClose(1)
'Write the results to the Tape5.inp file
FileOpen(1, Application.StartupPath & "\data\Origen\TAPE5.INP", OpenMode.
Output)
PrintLine(1, TempT5)
FileClose(1)
Else
MsgBox("Can't find reactor index " & Rindex & " Tape5 or Tape9 file")
End If
End Function
Function ParseTape6()
'Opens the Tape6.out file after Origen has been run and puts the moles of each
isotope into the TTape6 file
Dim found1, found2, done1 As Boolean
Dim i1, i2 As Integer
Dim TempS, elem1, iso1 As String
Dim index1, index2, length2, match1, match2 As Short
TTape6.Clear(TTape6, 0, 3000)
found1 = False
found2 = False
done1 = False
i2 = 0
found1 = System.IO.File.Exists(Application.StartupPath &
"\data\Origen\TAPE6.OUT"
)
If found1 = True Then
i1 = 0
Do Until done1 = True
Try
FileOpen(1, Application.StartupPath & "\data\Origen\TAPE5.INP",
OpenMode.Append)
done1 = True
Catch ex As Exception
i1 += 1
done1 = False
```

```
If i1 > 100000 Then
done1 = True
MsgBox("Origen has hung and hasn't release the output file.")
End If
End Try
Loop
FileClose(1)
done1 = False
6 D:\Programs\Inverse_Model3\Module1.vb
i1 = 0
Do Until done1 = True
Try
FileOpen(1, Application.StartupPath & "\data\Origen\TAPE6.OUT",
OpenMode.Input)
done1 = True
Catch ex As Exception
i1 += 1
done1 = False
If i1 > 100000 Then
done1 = True
MsgBox("Tape6.out can't be opened")
End If
End Try
Loop
done1 = False
Do Until (EOF(1)) Or done1 = True
Do Until (EOF(1)) Or found2 = True
TempS = LineInput(1)
index1 = TempS.IndexOf("ACTINIDES+DAUGHTERS")
If index1 > 0 Then found2 = True
Loop
If found2 = True And done1 = False Then
If TempS.IndexOf("NEUTRON SOURCE") > 0 Then done1 = True
TempS = LineInput(1)
index1 = TempS.IndexOf("POWER")
If index1 < 0 Then index1 = TempS.IndexOf("ORIGEN")
If index1 < 0 Then index1 = TempS.IndexOf("TOTAL")
If index1 < 0 Then index1 = TempS.IndexOf("AP+FP")
If index1 < 0 Then index1 = TempS.IndexOf("ACT+FP")
index2 = -1
If index1 < 0 And TempS.Length > 7 Then
index2 = TempS.IndexOf("+", 7)
If index2 < 0 Then index2 = TempS.IndexOf("-")
End If
If index2 > 0 And index2 < 23 Then
length2 = TempS.Length
TTape6(i2, 0) = TempS.Substring(0, 7)
TTape6(i2, 0) = TTape6(i2, 0).TrimEnd
TTape6(i2, 1) = TempS.Substring(7, length2 - 7)
TTape6(i2, 1) = TTape6(i2, 1).Trim
i2 += 1
End If
End If
Loop
FileClose(1)
'Changes the first column from a isotope format of U238M to 922381
Dim Temp3, Temp4 As String
For i1 = 0 To 1499
If TTape6(i1, 0) <> Nothing Then
elem1 = TTape6(i1, 0).Substring(0, 2)
elem1 = elem1.Trim
iso1 = TTape6(i1, 0).Substring(2, 3)
iso1 = iso1.Trim
```

```
For i2 = 0 To 999
Temp3 = IsoInfo(i2, 0)
Temp4 = IsoInfo(i2, 2)
match1 = StrComp(elem1, Temp3, CompareMethod.Text)
match2 = StrComp(iso1, Temp4, CompareMethod.Text)
If match1 = 0 And match2 = 0 Then
If IsoInfo(i2, 2).Length = 1 Then
TempS = "00" & IsoInfo(i2, 2)
ElseIf IsoInfo(i2, 2).Length = 2 Then
TempS = "0" & IsoInfo(i2, 2)
7 D:\Programs\Inverse_Model3\Module1.vb
Else
TempS = IsoInfo(i2, 2)
End If
'If TTape6(i1, 0).Length = 6 Then
If TTape6(i1, 0).IndexOf("M", 3) > 0 Then
TTape6(i1, 0) = IsoInfo(i2, 1) & TempS & "1"
Else
TTape6(i1, 0) = IsoInfo(i2, 1) & TempS & "0"
End If
End If
Next
End If
Next
Else
MsgBox("TAPE6.OUT not found")
End If
End Function
Function Find_Rat(ByVal Ratio1 As String, ByRef numRatio As Single, ByRef
numError As
Single)
'Function that will look up a needed ratio in the req_ratios table
Dim j1 As Short
Dim found1 As Boolean = False
j1 = 0
Find_Rat = True
Do Until found1 = True Or j1 > 198
If Req_ratios(j1, 0) <> Nothing Then
If Req_ratios(j1, 0).ToUpper = Ratio1.ToUpper Then
numRatio = Convert.ToSingle(Req_ratios(j1, 1))
numError = Convert.ToSingle(Req_ratios(j1, 2))
found1 = True
End If
End If
j1 += 1
Loop
If found1 = False Then
failed = True
MsgBox("Failed to find value for needed ratio")
End If
End Function
Function MatchRType(ByVal Rindex As Short) As Single
'Takes the info from the tape6.out file and determines how close it matches the
entered data for the reactor type
'Added on afterwards to tally age isotopes for calculating the age later on so
we
don't have to run Origen twice
'rtypelist Table to temporarily hold the reactor type isotopes from DV1
'column 0: Isotope Ratio (name)
'column 1: Numerator
'column 2: Denominator
'column 3: entered measured data by user
'column 4: entered measured error by user
```

```
'column 5: ratio calculated from Origen
Dim rtypelist(99, 5)
Dim i1, i2, i3, length1 As Short
Dim Found1, Found2 As Boolean
Dim numer1, denom1 As Single
Dim warning As Boolean = False
'Pull a list from the database of all burnup monitors
length1 = DV1.Count()
i2 = 0
i3 = 0
For i1 = 0 To length1 - 1
8 D:\Programs\Inverse_Model3\Module1.vb
If DV1.Item(i1).Item("GroupID") = 2 Then
rtypelist(i2, 0) = DV1.Item(i1).Item("Isotope Ratio")
rtypelist(i2, 1) = DV1.Item(i1).Item("Numerator")
rtypelist(i2, 2) = DV1.Item(i1).Item("Denominator")
i2 += 1
End If
Next
'Check that list against the user entered data
For i1 = 0 To i2 - 1
Found1 = False
i3 = 0
Do Until Found1 = True
If rtypelist(i1, 0) = Req_ratios(i3, 0) Then
rtypelist(i1, 3) = Req_ratios(i3, 1)
rtypelist(i1, 4) = Req_ratios(i3, 2)
Found1 = True
End If
If i3 = 199 Then Found1 = True
i3 += 1
Loop
Next
'Calculates the ratio from the tape6.out file for the burnup
warning = False
For i1 = 0 To 99
If rtypelist(i1, 3) <> Nothing Then
Found1 = False
Found2 = False
For i2 = 0 To 1499
If rtypelist(i1, 1) = TTape6(i2, 0) Then
numer1 = Convert.ToSingle(TTape6(i2, 1))
Found1 = True
End If
If rtypelist(i1, 2) = TTape6(i2, 0) Then
denom1 = Convert.ToSingle(TTape6(i2, 1))
Found2 = True
End If
Next
If Found1 = True And Found2 = True Then
rtypelist(i1, 5) = numer1 / denom1
Else
If warning = False Then
MsgBox("reactor type ratio not found in Origen output or elements
.csv")
warning = True
End If
End If
End If
Next
'Calculates how far the measured values differ from the origen calculated
values
Dim measure1, calc1 As Double
```

```
Dim errors(100) As Double
i2 = 0
errors(100) = 0
For i1 = 0 To 99
If rtypelist(i1, 3) <> Nothing Then
measure1 = rtypelist(i1, 3)
If measure1 <= 0 Then measure1 = 0.0000000000001
calc1 = rtypelist(i1, 5)
errors(i2) = Math.Abs(measure1 - calc1) / measure1
errors(i2) = Math.Exp(-errors(i2))
errors(100) += errors(i2)
i2 += 1
End If
9 D:\Programs\Inverse_Model3\Module1.vb
Next
errors(100) = errors(100) / i2
MatchRType = errors(100) * 100
For i1 = 0 To 99
If rtypelist(i1, 0) <> Nothing Then
RTypeErrors(Rindex, i1, 0) = rtypelist(i1, 0)
RTypeErrors(Rindex, i1, 1) = rtypelist(i1, 3)
RTypeErrors(Rindex, i1, 2) = rtypelist(i1, 5)
End If
Next
End Function
Function ReactType(ByVal OIon As Boolean)
Dim i1, i2, i3, i4 As Short
Dim burn1 As Single
i2 = 0
For i1 = 0 To 99
If xyfiles(i1, 0) <> Nothing Then
If OIon = True Then
i4 = 0
For i3 = 0 To 99
burn1 += IBurns(i1, i3, 1)
i4 += 1
Next
burn1 = burn1 / i4
BurnupTable(0, 1) = burn1
End If
OrigenPrep(i1, BurnupTable(0, 1), EnrichTable(i1, 0), 183)
RunOrigen()
ParseTape6()
RTypeConf(i2, 0) = i1
RTypeConf(i2, 1) = MatchRType(i1)
i2 += 1
End If
Next
End Function
Function GetFuelAge(ByVal OIon As Boolean)
'Calculates the fuel age for each reactor type
Dim AgeList(19), TH, No, NT, error1, sum1, numer1, denom1 As Single
Dim err1, err2, terr As Single
Dim i1, i2, i3, length1, Rindex, rtypes As Short
Dim AIso As String
Dim found1, found2, found3 As Boolean
Dim HalfL(19, 1) As Single
Dim AgeIso1, AgeIso2 As Integer
'Loop that will run Origen for each reactor type to collect isotope info at
time
of discharge
For Rindex = 0 To 99
If xyfiles(Rindex, 0) <> Nothing Then
```

```
'Pull a list from the database of all burnup monitors
length1 = DV1.Count()
i2 = 0
i3 = 0
For i1 = 0 To length1 - 1
'Pull the list of age monitors
If DV1.Item(i1).Item("GroupID") = 4 Then
IsoAge(Rindex, i3, 0) = DV1.Item(i1).Item("Isotope Ratio")
IsoAge(Rindex, i3, 1) = DV1.Item(i1).Item("Numerator")
IsoAge(Rindex, i3, 2) = DV1.Item(i1).Item("Denominator")
i3 += 1
10 D:\Programs\Inverse_Model3\Module1.vb
If i3 > 19 Then
i3 = 0
MsgBox("Error: only 19 age monitors are allowed")
End If
End If
Next
'Check that list against user entered data. If there is no match delete
the info
For i1 = 0 To i3 - 1
If IsoAge(Rindex, i1, 0) <> Nothing Then
found3 = False
For i2 = 0 To 199
If IsoAge(Rindex, i1, 0) = Req_ratios(i2, 0) Then
found3 = True
End If
Next
If found3 = False Then
For i2 = 0 To 2
IsoAge(Rindex, i1, i2) = Nothing
Next
End If
End If
Next
i2 = 0
Dim Burn23, Enrich23 As String
Dim i11, i12 As Short
Dim Burn1 As Single
If OIon = True Then
i12 = 0
For i11 = 0 To 99
Burn1 += IBurns(Rindex, i11, 1)
i12 += 1
Next
Burn1 = Burn1 / i12
BurnupTable(0, 1) = Burn1
End If
Burn23 = BurnupTable(0, 1)
Enrich23 = EnrichTable(Rindex, 0)
OrigenPrep(Rindex, BurnupTable(0, 1), EnrichTable(Rindex, 0), 30)
RunOrigen()
ParseTape6()
'Calculates the ratio from the tape6.out file for the age monitors
For i1 = 0 To 19
If IsoAge(Rindex, i1, 0) <> Nothing Then
found1 = False
found2 = False
For i2 = 0 To 1499
If IsoAge(Rindex, i1, 1) = TTape6(i2, 0) Then
numer1 = Convert.ToSingle(TTape6(i2, 1))
found1 = True
End If
```

```
If IsoAge(Rindex, i1, 2) = TTape6(i2, 0) Then
denom1 = Convert.ToSingle(TTape6(i2, 1))
found2 = True
End If
Next
If found1 = True And found2 = True Then
IsoAge(Rindex, i1, 3) = numer1 / denom1
Else
MsgBox("Age monitor isotope not found in Origen output or
elements.csv")
End If
11 D:\Programs\Inverse_Model3\Module1.vb
End If
Next
End If
Next
'Calculates an average age from each of the isotopes for each reactor
For i1 = 0 To 99
i3 = 0
sum1 = 0
For i2 = 0 To 19
If IsoAge(i1, i2, 0) <> Nothing Then
AIso = IsoAge(i1, i2, 0)
TH = GetHalfLife(IsoAge(i1, i2, 1))
HalfL(i2, 1) = TH
HalfL(i2, 0) = IsoAge(i1, i2, 1)
No = IsoAge(i1, i2, 3)
Find_Rat(IsoAge(i1, i2, 0), NT, error1)
AgeList(i3) = -TH / Math.Log(2) * Math.Log(NT / No)
sum1 += AgeList(i3)
i3 += 1
End If
Next
If i3 > 0 Then
FuelAge(i1) = sum1 / i3
End If
Next
sum1 = 0
i3 = 0
For i1 = 0 To 99
If FuelAge(i1) <> Nothing Then
sum1 += FuelAge(i1)
i3 += 1
End If
Next
If i3 > 0 Then
sum1 = sum1 / i3
End If
If sum1 < 1 Then sum1 = 1
'Find the isotopes that have a halflife closest to the average
err1 = 100000000
err2 = 100000000
For i1 = 0 To 19
If HalfL(i1, 1) <> 0 Then
terr = Math.Abs(sum1 - HalfL(i1, 1))
If terr < err1 Then
AgeIso1 = HalfL(i1, 0)
err1 = terr
End If
End If
Next
For i1 = 0 To 19
If HalfL(i1, 1) <> 0 Then
```

```
terr = Math.Abs(sum1 - HalfL(i1, 1))
If terr < err2 And HalfL(i1, 0) <> AgeIso1 Then
AgeIso2 = HalfL(i1, 0)
err2 = terr
End If
End If
Next
'Calculates a preferential age from the isotopes that have the closest halflife
to the average
For i1 = 0 To 19
12 D:\Programs\Inverse_Model3\Module1.vb
AgeList(i1) = Nothing
Next
For i1 = 0 To 99
If xyfiles(i1, 0) <> Nothing Then
i3 = 0
sum1 = 0
For i2 = 0 To 19
If IsoAge(i1, i2, 1) = AgeIso1 Or IsoAge(i1, i2, 1) = AgeIso2 Then
AIso = IsoAge(i1, i2, 0)
TH = GetHalfLife(IsoAge(i1, i2, 1))
No = IsoAge(i1, i2, 3)
Find_Rat(IsoAge(i1, i2, 0), NT, error1)
AgeList(i3) = -TH / Math.Log(2) * Math.Log(NT / No)
sum1 += AgeList(i3)
AgeErrors(i1, i3, 0) = IsoAge(i1, i2, 0)
AgeErrors(i1, i3, 1) = AgeList(i3)
i3 += 1
End If
Next
If i3 > 0 Then
FuelAge(i1) = sum1 / i3
End If
End If
Next
i1 = 0
End Function
Function GetHalfLife(ByVal isotope1 As Integer) As Single
'Finds the halflife for an isotope in the IsoInfo Table
Dim Anum, Mnum, Meta As Short
Dim i1, i2 As Short
Dim TempS As String
If isotope1 > 99999 Then
TempS = Convert.ToString(isotope1)
Anum = Convert.ToInt16(TempS.Substring(0, 2))
Mnum = Convert.ToInt16(TempS.Substring(2, 3))
Meta = Convert.ToInt16(TempS.Substring(5, 1))
ElseIf isotope1 < 99999 And isotope1 > 9999 Then
TempS = Convert.ToString(isotope1)
Anum = Convert.ToInt16(TempS.Substring(0, 1))
Mnum = Convert.ToInt16(TempS.Substring(1, 3))
Meta = Convert.ToInt16(TempS.Substring(4, 1))
Else
MsgBox("Can't get half life of Isotope " & isotope1)
End If
For i1 = 0 To 999
If Anum = IsoInfo(i1, 1) And Mnum = IsoInfo(i1, 2) And Meta = IsoInfo(i1, 3)
Then
GetHalfLife = IsoInfo(i1, 4) / 365.24
End If
Next
End Function
End Module
```

```
D:\Programs\Inverse_Model3\Report.vb
Imports System
Imports System.Globalization
Public Class Report
Inherits System.Windows.Forms.Form
Windows Form Designer generated code
Public Iterated As Boolean = False
Function loadreport()
Dim i1, i2, index1, index2, index3, index4, index5, length1 As Short
Dim er1, er2, er3, er4, er5, compare1, enrich1(4) As Single
' Displays a value with the default separator (",").
Dim nfi As NumberFormatInfo = New CultureInfo("en-US", False).NumberFormat
tbx_error.Visible = False
'Load the RTypeConf table to the Report1 table
i2 = 0
For i1 = 0 To 99
If RTypeConf(i1, 1) <> Nothing Then
Report1(i2, 3) = RTypeConf(i1, 1)
Report1(i2, 0) = RTypeConf(i1, 0)
i2 += 1
End If
Next
'Determine the top 5 Reactors with the highest confidence level
er1 = 0
er2 = 0
er3 = 0
er4 = 0
er5 = 0
For i1 = 0 To 99
compare1 = Convert.ToSingle(Report1(i1, 3))
If compare1 > er1 Then
er1 = Report1(i1, 3)
index1 = i1
End If
Next
For i1 = 0 To 99
compare1 = Convert.ToSingle(Report1(i1, 3))
If compare1 > er2 And compare1 < er1 Then
er2 = Report1(i1, 3)
index2 = i1
End If
Next
For i1 = 0 To 99
compare1 = Convert.ToSingle(Report1(i1, 3))
If compare1 > er3 And compare1 < er2 Then
er3 = Report1(i1, 3)
index3 = i1
End If
Next
For i1 = 0 To 99
compare1 = Convert.ToSingle(Report1(i1, 3))
If compare1 > er4 And compare1 < er3 Then
er4 = Report1(i1, 3)
index4 = i1
End If
Next
For i1 = 0 To 99
compare1 = Convert.ToSingle(Report1(i1, 3))
If compare1 > er5 And compare1 < er4 Then
er5 = Report1(i1, 3)
2 D:\Programs\Inverse_Model3\Report.vb
```

```
index5 = i1
End If
Next
'Match the reactor ID with the name from the access database
length1 = DV3.Count()
For i1 = 0 To 99
For i2 = 0 To length1 - 1
If Report1(i1, 0) = DV3.Item(i2).Item("ID") Then
Report1(i1, 1) = DV3.Item(i2).Item("Reactor Type")
End If
Next
Next
'Write the Age to the Report table
For i1 = 0 To 19
For i2 = 0 To 99
If Report1(i2, 0) = i1 Then
Report1(i2, 2) = FuelAge(i1)
End If
Next
Next
IDSelect(0) = Report1(index1, 0)
IDSelect(1) = Report1(index2, 0)
IDSelect(2) = Report1(index3, 0)
IDSelect(3) = Report1(index4, 0)
IDSelect(4) = Report1(index5, 0)
'Write the report1 table to the textboxes
Try
tbx_rtype1.Text = Report1(index1, 1)
tbx_rtype2.Text = Report1(index2, 1)
tbx_rtype3.Text = Report1(index3, 1)
tbx_rtype4.Text = Report1(index4, 1)
tbx_rtype5.Text = Report1(index5, 1)
tbx_burn1.Text = BurnupTable(0, 1).ToString("N", nfi)
tbx_age1.Text = Report1(index1, 2)
tbx_age2.Text = Report1(index2, 2)
tbx_age3.Text = Report1(index3, 2)
tbx_age4.Text = Report1(index4, 2)
tbx_age5.Text = Report1(index5, 2)
tbx_enrich1.Text = EnrichTable(0, 1)
tbx_error1.Text = Report1(index1, 3)
tbx_error2.Text = Report1(index2, 3)
tbx_error3.Text = Report1(index3, 3)
tbx_error4.Text = Report1(index4, 3)
tbx_error5.Text = Report1(index5, 3)
tbx_enrich1.Text = EnrichTable(Report1(index1, 0), 0)
tbx_enrich2.Text = EnrichTable(Report1(index2, 0), 0)
tbx_enrich3.Text = EnrichTable(Report1(index3, 0), 0)
tbx_enrich4.Text = EnrichTable(Report1(index4, 0), 0)
tbx_enrich5.Text = EnrichTable(Report1(index5, 0), 0)
Catch ex As Exception
End Try
enrich1(0) = EnrichTable(Report1(index1, 0), 0)
enrich1(1) = EnrichTable(Report1(index2, 0), 0)
enrich1(2) = EnrichTable(Report1(index3, 0), 0)
enrich1(3) = EnrichTable(Report1(index4, 0), 0)
enrich1(4) = EnrichTable(Report1(index5, 0), 0)
For i1 = 0 To 4
If enrich1(i1) > 10 Then
tbx_error.Visible = True
tbx_error.Text = "Warning! On high enrichments Pu240 can cause " & _
"significant errors when determining the reactor type. Please rerun " &
3 D:\Programs\Inverse_Model3\Report.vb
_
```

```
" this scenario after removing Pu240 from the measurements"
End If
Next
rbn_rtype1.Checked = True
If Iterated = True Then
SelBurn()
End If
End Function
Function OIterate()
Dim enrich1, burn1 As Single
Dim RI As Short
OIReport.Clear(OIReport, 0, 200)
RI = IDSelect(5)
burn1 = BurnupTable(0, 1)
tbx_prog.Visible = True
progress(0)
Refresh()
Application.DoEvents()
enrich1 = EnrichTable(IDSelect(5), 0)
enrich1 = OEnrich(RI, burn1, enrich1)
progress(20)
Refresh()
burn1 = OBurnup(RI, enrich1, burn1)
progress(40)
Refresh()
enrich1 = OEnrich(RI, burn1, enrich1)
progress(60)
Refresh()
burn1 = OBurnup(RI, enrich1, burn1)
progress(80)
Refresh()
enrich1 = OEnrich(RI, burn1, enrich1)
tbx_prog.Visible = False
End Function
Function progress(ByVal perc As Short)
tbx_prog.Text = "Origen Iteration Progress"
tbx_prog.Text += CR & "Progress = " & perc & " %"
Refresh()
End Function
Function OEnrich(ByVal Rindex As Short, ByVal Burn1 As Single, ByVal Enrich0 As
Single) As Single
'Find the enrichment by iterating Origen
Dim blank, i1, i2, enrich1, enrich2, error1 As Single
Dim numer1, denom1, ratio1, ratio2 As Single
Dim converge As Boolean
blank = 0
enrich1 = Enrich0
If enrich1 < 0 Then enrich1 = 1
Find_Rat("U235/U238", ratio1, blank)
i2 = 0
converge = False
'Iterate Origen until it converges
Do Until converge = True Or i2 = 10
OrigenPrep(Rindex, Burn1, enrich1, 90)
RunOrigen()
4 D:\Programs\Inverse_Model3\Report.vb
ParseTape6()
i1 = 0
For i1 = 0 To 1499
If Convert.ToString(922350) = TTape6(i1, 0) Then
numer1 = TTape6(i1, 1)
End If
If Convert.ToString(922380) = TTape6(i1, 0) Then
```

```
denom1 = TTape6(i1, 1)
End If
Next
ratio2 = numer1 / denom1
'Calculate the error
error1 = Math.Abs(ratio1 - ratio2) / ratio1
If error1 > 0.05 Then
If ratio1 > ratio2 Then
enrich1 = enrich1 * (error1 + 1)
Else
enrich1 = enrich1 * Math.Exp(-error1 / 1.5)
End If
ElseIf error1 < 0.05 And error1 > 0.0005 Then
If ratio1 > ratio2 Then
enrich1 = enrich1 * (error1 / 2 + 1)
Else
enrich1 = enrich1 * (1 - error1 / 2)
End If
Else
converge = True
End If
i2 += 1
Loop
'Write the result to the table
OIReport(0, 1) = enrich1
OEnrich = enrich1
i2 = 0
End Function
Function OBurnup(ByVal Rindex As Short, ByVal Enrich0 As Single, ByVal Burn0 As
Single) As Single
'Find the burnup by iterating Origen
Dim burn40, length1, numer1, denom1, ratio1, ratio2, enrich1 As Single
Dim blank As Single
Dim i1, i2, i3, i4 As Short
Dim iso1, iso2 As Integer
Dim Converge, flip As Boolean
Dim rname, error1 As String
blank = 0
enrich1 = Enrich0
If enrich1 < 0 Then enrich1 = 1
i4 = 0
burn40 = Burn0
'Run through each burnup monitor
For i3 = 0 To 99
If BurnErrors(i3, 0) <> Nothing Then
rname = BurnErrors(i3, 0)
i2 = 0
Converge = False
flip = False
'Iterate Origen until it converges
Do Until Converge = True Or i2 = 10
OrigenPrep(Rindex, burn40, enrich1, 90)
RunOrigen()
ParseTape6()
length1 = DV1.Count()
5 D:\Programs\Inverse_Model3\Report.vb
i1 = 0
'Find the ratios numerator and denominator
For i1 = 0 To length1 - 1
If DV1.Item(i1).Item("Isotope Ratio") = rname Then
iso1 = DV1.Item(i1).Item("Numerator")
iso2 = DV1.Item(i1).Item("Denominator")
End If
```

```
Next
i1 = 0
For i1 = 0 To 1499
If Convert.ToString(iso1) = TTape6(i1, 0) Then
numer1 = TTape6(i1, 1)
End If
If Convert.ToString(iso2) = TTape6(i1, 0) Then
denom1 = TTape6(i1, 1)
End If
Next
ratio2 = numer1 / denom1
'Calculate the error
Find_Rat(rname, ratio1, blank)
error1 = Math.Abs(ratio1 - ratio2) / ratio1
If error1 > 0.01 Then
If ratio1 > ratio2 Then
burn40 = burn40 * (error1 + 1)
Else
burn40 = burn40 * Math.Exp(-error1 / 1.5)
End If
ElseIf error1 < 0.01 And error1 > 0.0005 Then
If ratio1 > ratio2 Then
burn40 = burn40 * (error1 + 1)
Else
burn40 = burn40 * (1 - error1)
End If
Else
Converge = True
End If
i2 += 1
Loop
i4 += 1
'Write the result to the table
OIReport(i4, 0) = rname
OIReport(i4, 1) = burn40
OBurnup += burn40
End If
Next
OBurnup = OBurnup / i4
i2 = 0
End Function
Function FillR(ByVal Rindex As Short)
Dim i1 As Short
For i1 = 1 To 99
If OIReport(i1, 0) <> Nothing Then
IBurns(Rindex, i1 - 1, 0) = OIReport(i1, 0)
IBurns(Rindex, i1 - 1, 1) = OIReport(i1, 1)
End If
Next
EnrichTable(Rindex, 0) = OIReport(0, 1)
End Function
Function OIAll()
6 D:\Programs\Inverse_Model3\Report.vb
rbn_rtype1.Checked = True
Refresh()
IDSelect(5) = IDSelect(0)
OIterate()
FillR(IDSelect(5))
rbn_rtype2.Checked = True
Refresh()
IDSelect(5) = IDSelect(1)
OIterate()
FillR(IDSelect(5))
```

```
rbn_rtype3.Checked = True
Refresh()
IDSelect(5) = IDSelect(2)
OIterate()
FillR(IDSelect(5))
rbn_rtype4.Checked = True
Refresh()
IDSelect(5) = IDSelect(3)
OIterate()
FillR(IDSelect(5))
rbn_rtype5.Checked = True
Refresh()
IDSelect(5) = IDSelect(4)
OIterate()
tbx_prog.Visible = True
Refresh()
FillR(IDSelect(5))
progress(90)
Refresh()
ReactType(Iterated)
GetFuelAge(Iterated)
loadreport()
tbx_prog.Visible = False
Iterated = True
End Function
Function SelBurn()
'I initally set it up to only have one burnup for every reactor
'With the new option of iterating origen there is a different burnup
'for every reactor, this changes the burnup numbers in the tables as
'the different reactors are selected. So everything functions properly
Dim Rindex, i1, i2 As Short
Dim burn1 As Single
Dim nfi As NumberFormatInfo = New CultureInfo("en-US", False).NumberFormat
If rbn_rtype1.Checked = True Then
Rindex = IDSelect(0)
ElseIf rbn_rtype2.Checked = True Then
Rindex = IDSelect(1)
ElseIf rbn_rtype3.Checked = True Then
Rindex = IDSelect(2)
ElseIf rbn_rtype4.Checked = True Then
Rindex = IDSelect(3)
ElseIf rbn_rtype5.Checked = True Then
Rindex = IDSelect(4)
End If
BurnErrors.Clear(BurnErrors, 0, 200)
i2 = 0
For i1 = 0 To 99
If IBurns(Rindex, i1, 0) <> Nothing Then
burn1 += Convert.ToSingle(IBurns(Rindex, i1, 1))
BurnErrors(i2, 0) = IBurns(Rindex, i1, 0)
7 D:\Programs\Inverse_Model3\Report.vb
BurnErrors(i2, 1) = IBurns(Rindex, i1, 1)
i2 += 1
End If
Next
burn1 = burn1 / i2
tbx_burn1.Text = burn1.ToString("N", nfi)
BurnupTable(0, 1) = burn1
End Function
Private Sub ToolBar1_ButtonClick(ByVal sender As System.Object, ByVal e As
System.
Windows.Forms.ToolBarButtonClickEventArgs) Handles ToolBar1.ButtonClick
Select Case ToolBar1.Buttons.IndexOf(e.Button)
```

```
Case 0
Dim frminput As New Ratios
If rbn_rtype1.Checked = True Then
IDSelect(5) = IDSelect(0)
ElseIf rbn_rtype2.Checked = True Then
IDSelect(5) = IDSelect(1)
ElseIf rbn_rtype3.Checked = True Then
IDSelect(5) = IDSelect(2)
ElseIf rbn_rtype4.Checked = True Then
IDSelect(5) = IDSelect(3)
ElseIf rbn_rtype5.Checked = True Then
IDSelect(5) = IDSelect(4)
End If
frminput.ShowDialog()
Case 1
Case 2
Case 3
Dim frmOIR As New FrmOIReport
If rbn_rtype1.Checked = True Then
IDSelect(5) = IDSelect(0)
ElseIf rbn_rtype2.Checked = True Then
IDSelect(5) = IDSelect(1)
ElseIf rbn_rtype3.Checked = True Then
IDSelect(5) = IDSelect(2)
ElseIf rbn_rtype4.Checked = True Then
IDSelect(5) = IDSelect(3)
ElseIf rbn_rtype5.Checked = True Then
IDSelect(5) = IDSelect(4)
End If
OIReport.Clear(OIReport, 0, 200)
OIterate()
frmOIR.ShowDialog()
Case 4
IBurns.Clear(IBurns, 0, 20000)
OIAll()
Case 5
End Select
End Sub
Private Sub Report_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs)
Handles MyBase.Load
loadreport()
End Sub
Private Sub rbn_rtype1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rbn_rtype1.CheckedChanged
If Iterated = True Then
SelBurn()
End If
End Sub
8 D:\Programs\Inverse_Model3\Report.vb
Private Sub rbn_rtype2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rbn_rtype2.CheckedChanged
If Iterated = True Then
SelBurn()
End If
End Sub
Private Sub rbn_rtype3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rbn_rtype3.CheckedChanged
If Iterated = True Then
SelBurn()
End If
End Sub
Private Sub rbn_rtype4_CheckedChanged(ByVal sender As System.Object, ByVal e As
```

```
System.EventArgs) Handles rbn_rtype4.CheckedChanged
If Iterated = True Then
SelBurn()
End If
End Sub
Private Sub rbn_rtype5_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rbn_rtype5.CheckedChanged
If Iterated = True Then
SelBurn()
End If
End Sub
End Class
```

VITA

Mark R. Scott graduated from Brigham Young University in August 2003 with a Bachelor of Science in Physics. After graduation he accepted employment with N-3 at Los Alamos National Laboratory. Future contact can be made by e-mail at marks@scophoto.com or by mail forwarded through the Department of Nuclear Engineering, c/o Dr. William Charlton, Texas A&M University, College Station, TX 77843-3133.


Los Alamos
NATIONAL LABORATORY
EST.1943