# DEVELOPMENT OF AN APPLICATION PROGRAMMING INTERFACE FOR DEPLETION ANALYSIS (APIDA)

A Dissertation
Presented to
The Academic Faculty

by

Daniel Edgardo Lago

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Nuclear and Radiological Engineering

Georgia Institute of Technology

May 2016

# DEVELOPMENT OF AN APPLICATION PROGRAMMING INTERFACE FOR DEPLETION ANALYSIS (APIDA)

Approved by:


Dr. Farzad Rahnema, Advisor
George W. Woodruff School
*Georgia Institute of Technology*

Dr. Tom Morley
School of Mathematics
*Georgia Institute of Technology*

Dr. Bojan Petrovic
George W. Woodruff School
*Georgia Institute of Technology*

Dr. Glenn E. Sjoden
*Air Force Technical Applications Center*

Dr. Dingkang Zhang
George W. Woodruff School
*Georgia Institute of Technology*


Date Approved: March 31, 2016

*To my DANK memes. And also my lovely wife.*

# ACKNOWLEDGEMENTS

Monumental accomplishments in life are rarely achieved without the support and guidance of multiple people and this thesis is no exception. I would like to thank all my friends and family who have supported and inspired me during my tenure at Georgia Tech, as well as the people who have helped me become who I am today.

First, I'd like to thank the members of my committee – Dr. Farzad Rahnema, Dr. Glenn Sjoden, Dr. Bojan Petrovic, Dr. Tom Morley, and Dr. Dingkang Zhang – for participating in the critique of my work. I want to thank my advisor Dr. Rahnema for his support during my five years at Georgia Tech and for his belief in me to pursue and develop such a daunting graduate project. I would also like to express gratitude to Dr. Sjoden for teaching the indispensable fundamentals in developing my skillset as a nuclear engineer during my time at the University of Florida.

I would be remiss not thank the exuberant, esoteric, and somewhat mercurial members of the CRMP lab at Georgia Tech. The friendships and bonds forged during our many late evenings at the lab and several group outings will forever serve as fond memories. In an effort to avoid an endless barrage of complaints, I will thank them all by name: Andrew Holcomb, Ryan Hon, Alex Huning, Gabriel Kooreman, Christopher Chapman, Kyle Remley, and Saam Yasseri. I must also thank my longtime colleague Jessica Paul; a trusted friend and confidant who was always around to provide advice and levity.

I must also mention my family, specifically my parents Edgardo and Rosalinda Lago, for instilling the values and work ethic needed to get me to this point of my life.

While not as apparent in the midst of my youth, there have been numerous crystalizing moments in my upbringing responsible for my development as a human being and my parents are behind almost every one of them.

Most importantly, I would like to thank my wife and best friend Lily. My time in Atlanta has been exciting, challenging, and integral to my growth as a person and a researcher, and it has been an experience made immeasurably more exceptional with your accompaniment. I love you and I eagerly await the adventures ahead of us as we enter the next chapter of our life.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| ABTR | Advanced Burner Test Reactor. |
| API | Application Programming Interface. |
| APIDA | API for Depletion Analysis. |
| BC | Boundary Condition. |
| BWR | Boiling Water Reactor. |
| CANDU | Canada Deuterium Uranium reactor. |
| CASMO | Multi-group 2-D transport code developed by Studsvik. |
| CINDER'90 | Burnup solver for MCNP using Markovian chains. |
| COMET | Coarse Mesh Radiation Transport Method. |
| CPM | Collision probability method. |
| CRAM | Chebyshev Rational Approximation Method. |
| CRMP | Computational Reactor and Medical Physics Laboratory. |
| ENDF | Evaluated Nuclear Data File. |
| EPR | European Pressurized Reactor/Evolutionary Power Reactor. |
| HELIOS | MOC/CPM Lattice physics code. |
| HPC | High Performance Computing. |
| HTGR | High Temperature Gas cooled Reactor. |
| LWR | Light Water Reactor. |
| MCNP | Monte Carlo N–Particle Transport Code. |
| MOC | Method of characteristics. |
| NFY | Neutron-induced fission product yields. |
| OpenMP | An API for multi-platform shared-memory parallel programming in C/C++ and Fortran. |

| | |
|---|---|
| ORIGEN | Isotope Generation and Depletion Code Matrix Exponential Method. |
| ORNL | Oak Ridge National Laboratory. |
| PARAGON | CPM 2-D transport code developed by Westinghouse. |
| PCM | per cent mille ($10^{-5}\Delta k/k$). |
| PENBURN | Linear Chain Burnup/Depletion Solver coupled to PENTRAN or Multigroup MCNP. |
| PENTRAN | Parallel Environment Transport Code. |
| PWR | Pressurized Water Reactor. |
| RDD | Radioactive decay data. |
| RSICC | Radiation Safety Information Computational Center. |
| SCALE | Software Suite by ORNL for Nuclear Safety Analysis and Design. Code. |
| SERPENT | A 3-D continuous-energy Monte Carlo code, developed at VTT Technical Research Centre of Finland. |
| SFY | Spontaneous fission product yields. |
| SPaRC | Stochastic Particle Response Calculator. |
| ZAID | Nuclide identification number. |

# SUMMARY

A new utility has been developed with extensive capabilities in identifying nuclide decay and transmutation characteristics, allowing for accurate and efficient tracking of the change in isotopic concentrations in nuclear reactor fuel over time. This tool, named the Application Programming Interface for Depletion Analysis (APIDA), employs both a matrix exponential method and a linear chain method to solve for the end-of-time-step nuclide concentrations for all isotopes relevant to nuclear reactors. The Chebyshev Rational Approximation Method (CRAM) was utilized to deal with the ill-conditioned matrices generated during the course of lattice depletion calculations, and a complex linear chain solver was developed to handle isotopes reduced from the burnup matrix due to either radioactive stability or a sufficiently low neutron-induced reaction cross section. The entire tool is housed in a robust but simple application programming interface (API). The development of this API allows other codes, particularly numerical neutron transport solvers, to incorporate APIDA as the burnup solver in a lattice depletion code in memory, without the need to write or read from the hard disk. Specifically, APIDA was developed for coupling with the coarse mesh radiation transport method (COMET) – a numerical transport solver extensively validated and shown to provide efficient and accurate whole core solutions to host of different reactor types. The APIDA code was benchmarked using numerous decay and transmutation chains. Burnup solutions produced by APIDA were shown to provide material concentrations comparable to the analytically solved Bateman equations - well below 0.01% relative error for even the most extreme cases using isotopes with vastly different decay constants. For further benchmarking, APIDA was coupled with

the transport solver in the SERPENT code for a fuel pin cell depletion problem. A sensitivity analysis was also conducted to determine the optimal number of isotopes to track for a typical pressurized water reactor (PWR) problem in order to accurately track the change in eigenvalue of the core. Results show APIDA to be effective and efficient in solving lattice depletion problems, in addition to being successful in terms of portability for users to implement via the API.

# CHAPTER 1

# INTRODUCTION

The world's energy demands are at historic levels and the need for clean, sustainable, and dependable power is unquestionably at the forefront of modern society. Nuclear power is increasingly becoming the most powerful tool to combat the impending energy crisis, subsequently making adequate tools to safely operate nuclear reactors and accurately track the changes in fuel over the core lifetime vital to ensuring the advent of a new fleet of nuclear power plants.

Nuclear fuel in fission reactors undergoes constant and significant change during operation with criticality, radioactivity, and material performance all affected as a result. The formation of new isotopes following the absorption of a neutron in the fuel causes power shifts and flux profile changes; consequently, these phenomena need to be monitored to ensure safe and efficient reactor operation. Tracking the transmutation of fuel is also important after the core lifetime, as radiation shielding and national security become a significant concern once spent fuel is removed and stored for either permanent disposal or reprocessing.

Computer simulation is the primary tool used in reactor analysis and design. Multiple codes are deployed by both researchers and operators to monitor the neutron population and distribution in the core, calculate the core eigenvalue (k-effective), track the change and formation of isotopes, and ensure proper heat transfer and cooling. Of those

facets, accurately tracking the material composition in the fuel is relevant to reactor lifetime, fuel utilization, used nuclear fuel management, and nuclear safeguards.

The current push in reactor physics research is to create all-encompassing codes capable of capturing all the multiphysics present in the complicated core of a nuclear reactor – thermal hydraulics, material performance, radiation shielding, and kinetics being the most relevant. This work aims to create a novel burnup module with sufficient modernization to couple with current and future neutron transport solution methods.

### 1.1 Motivation

One emerging code in constant development at the Georgia Institute of Technology is the Coarse Mesh Radiation Transport Method (COMET). A hybrid deterministic-stochastic transport solution method, COMET uses incident flux response expansions to quickly and accurately calculate core eigenvalues, fuel pin fissions density distributions, and other important values. COMET has been shown to provide solutions with Monte Carlo accuracy at a fraction of the computational time for PWRs, BWRs, CANDU reactors, HTGRs, and ABTR [8, 19, 32, 33, 34, 35].

COMET presents a wholesale advancement in the archives of response matrix solution methods, particularly those applied to reactor physics and medical physics. While COMET in its current form is a powerful resource in reactor design, it also has the potential to be applied effectively as a lattice depletion tool. The purpose of this work was to develop an accurate and efficient burnup tool to couple with COMET in the future. This work was performed in coincidence with another Ph.D. project in the Computational Reactors and Medical Physics (CRMP) laboratory at The Georgia Institute of Technology. The

coinciding project involved developing a response function generation tool to work in conjunction with the burnup solver to track the change in nuclide concentration in reactor problems over time. Ultimately, these new tools will be implemented within the COMET framework to create one cohesive, deployable program capable of solving challenging reactor problems.

### 1.2 Objectives

The research in this study addresses the need for more efficient ways to accurately predict the change in isotopics in reactor fuel during the course of operation. While methods do currently exist to calculate the end-of-life-cycle nuclide concentrations in reactor fuel, they do so with a number of limitations. The majority of these limitations lie in the transport method used to generate reaction rates at each burnup time step, but several burnup tools currently in use are handcuffed in some way as well.

The purpose of this project is to develop a burnup module to couple with the COMET method, ultimately creating an efficient and accurate lattice depletion tool. En route to the ultimate goal of a fully capable lattice depletion tool, significant work was done in developing an API framework for an efficient Bateman equation solver. Numerous methods have been developed over the past 40 years to solve these equations with most of those employing numerous liberties and workarounds to circumvent the difficulties caused by the wide-range in eigenvalues present in almost every burnup calculation. This work aims to provide a tool powerful enough to solve any type of burnup problem with any number of isotopes. In doing so, the following goals were laid out:

1. Develop an efficient and accurate method for solving the decay/depletion equations (the Bateman equations).
2. Verification of this solver for a number of simple and complex decay/transmutation chains.
3. Create an API framework for this solver with a clean, but robust front-end interface.
4. Benchmark the burnup tool using a widely used transport and lattice depletion code (SERPENT).
5. Introduce the concept of coupling COMET and APIDA to be implemented in future work.

# CHAPTER 2

# BACKGROUND AND THEORY

In this section, the fundamental theory of radioactive decay is outlined and described briefly. The relevance of radioactive decay and other pathways for transmutation is also discussed, and the basis for the Bateman equations governing burnup is developed. The role of fission, with specific mention of fission product yield curves for different isotopes, is also reviewed in this section. Finally, the importance of neutron-induced reactions is considered with special note of the ones most relevant to burnup calculations.

## 2.1 Radioactive Decay

Among the different pathways for an isotope to change its given mass and energy, the most fundamental one in nuclear physics is radioactive decay. There are a number of radioactive nuclides that occur naturally on earth and a host of others made via human intervention, and all of them exhibit radioactive characteristics thanks to the lack of stability in their respective nuclei.

The stability of a particular nucleus is determined primarily by its average binding energy, BE, per nucleon. The binding energy is defined as the energy released when the atom is "created" from its constituent parts – hydrogen atoms (H) plus neutrons (N). The average binding energy can then be calculated using the known masses for each particle – hydrogen ($m_H$), neutron ($m_n$), and electron ($m_E$) [25].

$$BE_{AVG} = \frac{Zm_H + Nm_n - m_E}{N + Z} \tag{1}$$

The probability of an atom fissioning has a particularly strong correlation to the binding energy of a nucleus. The stability curve in relation to binding energy per nuclear particle seen below shows a steady decline towards a threshold where particles are capable of fissioning. As explained in the figure, once a particle's mass becomes unwieldy, the binding energy per nucleon decreases enough for the atom to undergo fission.

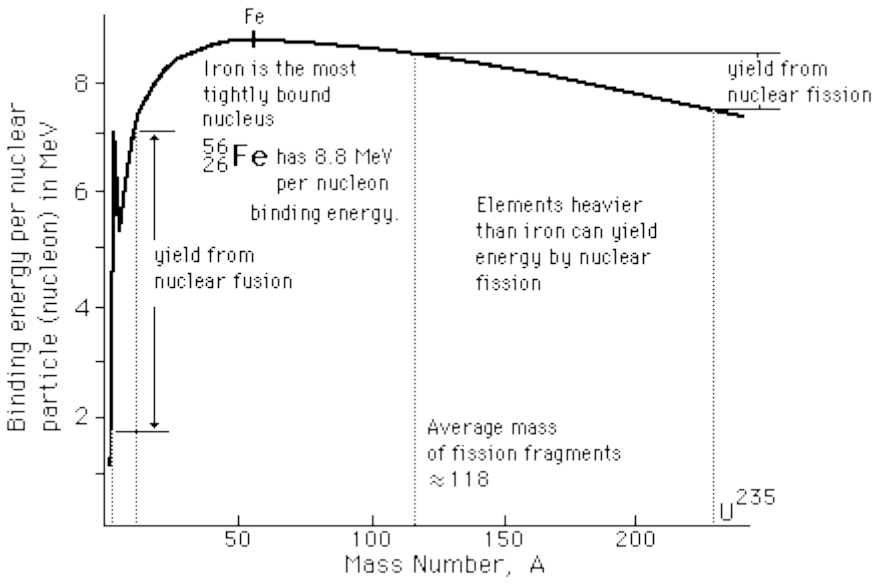


**Figure 1.** Plot of the binding energy per nucleon and related notes on nuclear stability [20].

The mode of decay varies by isotope and is determined by the quantum mechanical properties of its nucleus. There are several decay types, the most pertinent of which to burnup calculations are outlined below in Table 1.

**Table 1:** Decay modes and their associated reactions.

| Decay Mode | Reaction |
|---|---|
| Gamma (γ) | ${}^{A}_{z}X^{*} \rightarrow {}^{A}_{z}X + \gamma$ |
| Alpha (α) | ${}^{A}_{z}X \rightarrow {}^{A-4}_{z-2}Y + \alpha$ |
| Beta ($\beta^-$) | ${}^{A}_{z}X \rightarrow {}^{A}_{z+1}Y + \beta^- + \bar{\nu}$ |
| Beta ($\beta^+$) | ${}^{A}_{z}X \rightarrow {}^{A}_{z-1}Y + \beta^+ + \nu$ |
| Electron Capture (*EC*) | ${}^{A}_{z}X + e^- \rightarrow {}^{A}_{z-1}Y^{*} + \nu$ |
| Proton (*p*) | ${}^{A}_{z}X \rightarrow {}^{A-1}_{z-1}Y + p$ |
| Neutron (*n*) | ${}^{A}_{z}X \rightarrow {}^{A-1}_{z}X + n$ |
| Spontaneous fission | ${}^{A}_{z}X \rightarrow$ Fission Products |

While there are other decay modes not outlined in Table 1, they have extremely low probability of occurring and have negligible contributions in regards to burnup calculations.

In addition to the method of disintegration, the other important parameter in radioactivity is the half-life of a particular isotope. Since radioactive decay of a material is a statistical phenomenon, there is no way predicting when a single nucleus will decay. Instead, one uses an isotope's decay probability (λ) to characterize the likelihood it will disintegrate over a given time.

Considering a given concentration of a radioactive nuclide *N*, the time rate of change of that nuclide over time can be defined as,

$$\frac{dN}{dt} = -\lambda N(t) \ , \tag{2}$$

the solution of which is

$$N(t) = N_0 e^{-\lambda t} \ , \tag{3}$$

where $N_0$ is the initial concentration of the nuclide.

Consequently, the exponential nature of decay allows one to calculate a useful property: the amount of time it takes for a given number of nuclides to decay to half of its initial amount, or its 'half-life'.

$$\begin{aligned} N(T_{1/2}) &= \frac{N_0}{2} = N_0 e^{-\lambda T_{1/2}} \\ T_{1/2} &= \frac{ln(2)}{\lambda} \end{aligned} \tag{4}$$

Experimental evaluation of half-lifes are extremely important as their accuracy affects burnup calculations significantly. Table 2 shows some half-lifes of interest.

**Table 2:** Half-lifes of interest [9].

| **Nuclides** | **Half-life** |
|---|---|
| $U^{235}$ | $7x10^8$y |
| $U^{238}$ | $4.5x10^9$y |
| $Pu^{238}$ | 87.7y |
| $Pu^{239}$ | 24,110y |
| $Pu^{240}$ | 6,536y |
| $Xe^{135}$ | 9.2 hours |
| $I^{135}$ | 6.57 hours |
| $Cs^{137}$ | 30.17y |

## 2.2 Bateman Equations

The formalization of tracking decay in Equation 3 is valid for one isotope, but the extension to a chain of decay products is quite natural. Tracking the decay of a nuclide and its subsequent products creates a system of coupled, first order, ordinary differential equations known as the Bateman equations. The simplest of these equations is when there are no sources for the production of isotopes outside of decay. Equation 5 shows the result of tracking the decay of an isotope being produced by a single parent isotope [16].

$$\frac{dN_i}{dt} = -\lambda_i N_i + \lambda_{i-1} N_{i-1} \tag{5}$$

$N_i =$ atom density of nuclide $i$
$\lambda_i =$ radioactive decay constant of nuclide $i$

Given an arbitrary set of nuclides, the generalized solution to $N_i$ in the proposed system of ordinary differential equations is as follows,

$$N_i(t) = N_1^0 \lambda_1 \lambda_2 \ldots \lambda_{i-1} \sum_{j=1}^{i} \frac{e^{-\lambda_j t}}{\prod_{\substack{k=1 \\ k \neq j}}^{i} (\lambda_k - \lambda_j)} \tag{6}$$

$N_i =$ atom density of nuclide $i$
$N_1^0 =$ initial atom density of the first nuclide in the chain
$\lambda_i =$ radioactive decay constant of nuclide $i$

Equation 6 describes the solution when the only driver for time rate of change is the first nuclide in the linear chain. Taking transmutation, the changing of one element into

another via nuclear bombardment, into account, the resulting solution is shown in Equation 7 [16].

$$N_i(t) = \sum_{l=1}^{i-1} \left[ N_l^0 \eta_l \eta_{l+1} \ldots \eta_{i-1} \sum_{j=1}^{i} \frac{e^{-\mu_j t}}{\prod_{\substack{k=l \\ k \neq j}}^{i} (\mu_k - \mu_j)} \right] + N_i^0 \left( \frac{e^{-\mu_i t}}{\mu_i} \right) \tag{7}$$

$N_i =$ atom density of nuclide $i$

$N_i^0 =$ initial atom density of the $i^{th}$ nuclide in the chain

$\eta_i =$ chain-linking precursor decay constant of nuclide $i$ including decay of other nuclides ($\lambda$) and reaction rates ($\sigma\phi$)

$\mu_i =$ effective decay constant of nuclide $i$

In this form of the Bateman equation, new decay constants are introduced in order to account for destruction driven by the presence of a reactor flux (φ) and the production via chain-linking precursors.

The most extensive form of the Bateman equation includes production from an external source, generally fission product generation in the presence of a reactor flux [16].

$$Q_i(t) = \sum_{l=1}^{i-1} \left[ P_l^0 \eta_l \eta_{l+1} \ldots \eta_{i-1} \sum_{j=1}^{i} \frac{\left(1 - e^{-\mu_j t}\right)/\mu_j}{\prod_{\substack{k=l \\ k \neq j}}^{i} (\mu_k - \mu_j)} \right] + P_i^0 \left( \frac{1 - e^{-\mu_i t}}{\mu_i} \right) \tag{8}$$

$$P_i^0 = \sum_{k=1}^{K} y_i^k N_k^0 \sigma_f^k \phi,$$

$y_i^k =$ $i^{th}$ fission product yield fraction from $k^{th}$ fissile nuclide

$N_k^0 =$ inital concentration of parent fissile nuclide $k$

$\sigma_f^k =$ fission cross section of parent fissile nuclide $f$ $(cm^2)$

$\phi =$ reactor flux $(n/cm^2 s)$

$P_i^0 =$ initial constant rate of formation for nuclide $i$ $(\sec^{-1})$

$Q_i(t) =$ atoms of nuclide $i$ produced at time $t$

Equation 8 was formulated assuming the initial amount of nuclide $N_i$ to be zero and a constant production represented by $P_i$. It is also important to note the units of the two terms, where $Q_i$ is the nuclide concentration in atoms, whereas $P_i$ is the constant rate of production of nuclide $i$ with units of $sec^{-1}$.

The linear nature of the Bateman equations allows these equations to be solved separately and then combined for the total solution by the property of superposition. In the case of no reactor flux, only the basic Bateman equation (Equation 6) is used. When a reactor flux is present, the Bateman equations with production and the effective decay constants are summed together (Equations 7 and 8).

Further discussed in Chapter 3, this set of Bateman equations is the basis for the commonly utilized linear chain methods present in some burnup codes.

### 2.3 Nuclear fission, fissile isotopes, and fission products

Nuclear fission is a reaction predicated by the formation of a compound nucleus. A nuclide amenable to a particular reaction, quantified by its specific reaction cross section, absorbs a neutron and its nucleus becomes unstable. Generally, nuclides are more inclined to undergo fission if they are neutron rich (i.e. large number of nucleons), an expected outcome when accounting for the importance of binding energy per nucleon. One of the most common examples in regards to fission in nuclear reactors is shown in Equation 9.

$$^{235}_{92}U + n \rightarrow {}^{236}_{92}U^{*} \rightarrow {}^{144}_{56}Ba + {}^{89}_{36}Kr + 3n + E_{Fission} \quad (9)$$

In this example, an isotope commonly used in light water reactor fuel ($^{235}$U) undergoes fission. It initially becomes a compound nucleus with an excited state, but effectively instantaneously fissions, resulting into two large fission products, an excess of neutrons with some given energy, and a release of energy, the magnitude of which is dependent on the excess mass and the excitation level of the compound nucleus. The neutrons produced as a result of the fission are monumentally important in sustaining the fission chain reaction in a nuclear reactor core. The incoming energy of the neutron and the target nucleus undergoing fission both affect the eventual fission products and the amount of excess neutrons released.

The result of one atom fissioning is not always the same. Depending on the energy of the incoming neutron and the state of the target nucleus, different isotopes can be produced as a result of the fission. Predicting which nuclides are created as the result of an atom fissioning is achieved using empirically generated fission yield curves. Compiled using experimental data, these curves have a distinct “double hump”, a phenomenon that correlates with what is observed in fission events – the formation of two large fission

products, one of which is usually 60 – 80 nucleons heavier than the other. Figure 2 shows the fission yield curves for $^{235}U$ and $^{239}Pu$ when they absorb a thermal neutron and consequently fission.

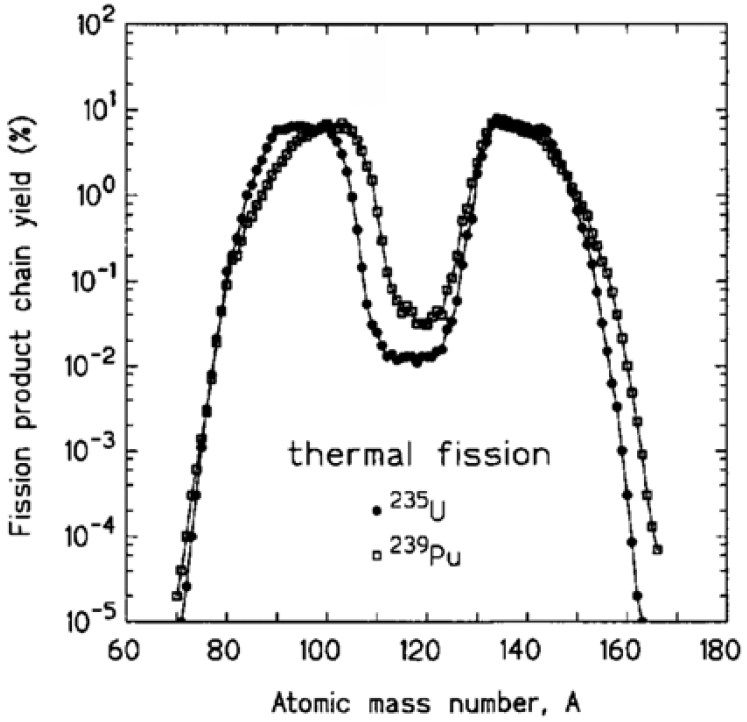


**Figure 2.** Fission product yield distribution for thermal fission of $^{235}U$ and $^{239}Pu$ [25].

Most current nuclear reactors operate with the assumption of incident neutrons with thermal energies dominating the fission chain reaction, but fission can occur at varying energies. Figure 2 shows the distribution of fission products from thermal fission – generally the boundary, or upper limit, for thermal fission is 0.0253 eV. The current set of nuclear data (ENDF/B-VII libraries) contain data for fissions in 3 different energy regions – thermal, epithermal (upper boundary of 2.0 MeV), and high energy fission (upper boundary of 14 MeV) [12]. Not all fissionable isotopes contain data for each energy region.

For isotopes with data for 2 or more energy groups, standard linear interpolation is usually implemented to calculate the resulting yield of each fission product.

### 2.4 Other neutron-nuclide interactions

Fission is the primary contributor to the reaction rates highlighted in Equations 7 and 8 for the fissionable nuclides tracked in a given problem, but reaction rates constitute all neutron-induced reactions in burnup calculations. Isotopes that don't undergo fission still contribute to the production of other isotopes via other reactions. Table 3 outlines the neutron-nuclide interactions of interest.

**Table 3.** Neutron-nuclide interactions relevant to burnup calculations.

| Decay Mode | Reaction |
|---|---|
| $(n,2n)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-1}_{Z}X + 2n$ |
| $(n,3n)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-2}_{Z}X + 3n$ |
| $(n,4n)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-2}_{Z}X + 4n$ |
| $(n,\gamma)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A+1}_{Z}X + \gamma$ |
| $(n,p)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A}_{Z-1}Y + p$ |
| $(n,d)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-1}_{Z-1}Y + d$ |
| $(n,t)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-2}_{Z-1}Y + t$ |
| $(n,{}^{3}He)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-2}_{Z-2}Y + {}^{3}He$ |
| $(n,\alpha)$ | ${}^{A}_{Z}X + n \rightarrow {}^{A-3}_{Z-2}Y + \alpha$ |

In terms of applications to burnup up problems and Bateman equations, neutron-nuclide interactions are treated effectively as constants – a phase-space integral of the microscopic cross section times the scalar flux.

$$RR = \int dX \phi(X) \sigma_i(X) \tag{10}$$

The phase-space X constitutes the independent variables acknowledged in formulating the original problem; generally, space, energy, angle. The result is effectively an average reaction rate with the same units as the decay constant ($seconds^{-1}$).

**2.5 Decay and fission yield data**

As with any nuclear engineering application, a topic worthy of consideration is the methods used in acquiring and validating the accuracy of the prerequisite data used for calculations. Cross sections are important in any transport calculation but burnup solution methods uniquely require decay data and fission product yield data.

The primary source of most radioactive decay data is from the Evaluated Nuclear Data Files (ENDF/B-*). ENDF/B libraries, the latest and most commonly used being ENDF/B-VII, contain an enormous amount of data for every nuclide relevant to most calculations [7]. ENDF/B-VII is comprised for 14 sublibraries with 3 libraries being the most pertinent in regards to burnup – radioactive decay data (RDD), spontaneous fission product yields (SFY), and neutron-induced fission product yields (NFY).

The ENDF/B-VII library is widely used and validated by numerous high-pedigree simulation codes both in industry and research laboratories. Oak Ridge National Lab

(ORNL) utilizes the ENF/B-VII libraries to construct their own set of decay and yield libraries, publicly distributed as part of the Scale code package via the Radiation Safety Information Computational Center (RSICC). For sake of comparison and ease of use, the decay libraries from the Scale code package were used in this study, but future work will implement a data reader to extract data directly from the ENDF libraries.

### 2.6 Burnup problems, systems of differential equations, and matrix structure

The formalizations of the decay equations and the solutions via Bateman equations in the previous sections are the foundation for burnup and depletion analysis. For the purposes of this thesis and the development of a burnup tool, the subsequent generalized equation is used to begin building the foundation for the solution method implemented in this work.

$$\frac{dN_i}{dt} = \underbrace{\sum_{j=1}^{m} b_{ij}\lambda_j N_j + \sum_{k=1}^{m} y_{ik} RR_k N_k}_{\text{Production}} \underbrace{-(\lambda_i + RR_i)N_i}_{\text{Destruction}} \tag{11}$$

$$RR_i = \int dX \phi(X)\sigma_i(X)$$

$N_i =$ atom density of nuclide $i$
$\lambda_i =$ radioactive decay constant of nuclide $i$
$\sigma_i(X) =$ neutron absorption cross-section for nuclide $i$ over phase-space $X$
$\phi(X) =$ neutron angular flux over phase-space $X$
$b_{ij} =$ branching ratio of all other nuclides to nuclide $i$
$y_{ik} =$ branching ratio for neutron absorption by other nuclides that lead to nuclide $i$

Equation 11 shows the time rate of change of the concentration of nuclide $N_i$ as a balance equation with the net result being the sum of the destruction and production of the nuclide of interest. The necessary assumptions in forming Equation 11 are as follows: a homogenous medium; space-averaged and energy-integrated reaction rate over one energy group; sufficiently small time step to assume a constant flux. In most lattice depletion applications, these assumptions are acceptable and provide accurate solutions if utilized appropriately.

Production of $N_i$ can result from the decay of another nuclide $N_j$ into $N_i$ with the probability of said reaction expressed by its associated branching ratio $b_{ij}$, or production can come from nuclide $N_k$ participating in a reaction under the influence of a flux resulting in the production of nuclide $N_i$. Destruction of $N_i$ is determined by two factors, both of which are dependent on the type of problem and isotope. If the nuclide is unstable, its decay probability ($\lambda$) determines the removal rate. If there is irradiation, implying a flux ($\phi$), then the nuclide's reaction rate determines the removal rate. If both principles are appropriate, then the decay probability and reaction rate are summed to constitute the removal coefficient for the nuclide $N_i$.

Equation 11 is valid for all isotopes being tracked in a given problem, the end result being a system of first order, ordinary differential equations. The structure of the matrix in burnup problems is distinct and consistent – extremely sparse with the non-zero elements bunched near the diagonal, save for the fission products which are bunched up to the right side of the matrix. This matrix structure assumes the nuclides are arranged in ascending order by atomic mass.

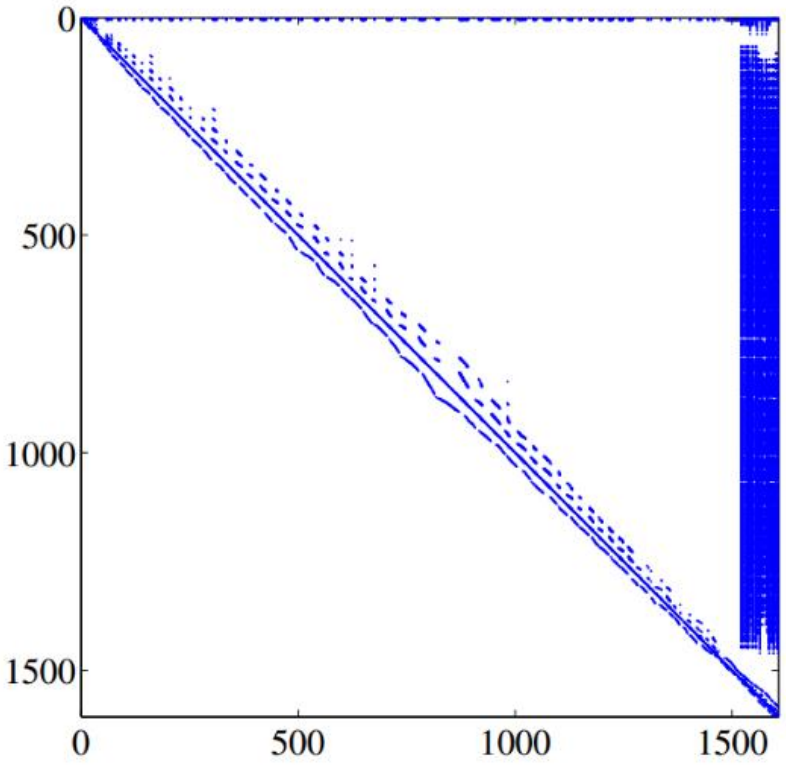


**Figure 3.** Structure of a burnup matrix with 1606 nuclides in ascending order [22].

In the preceding figure, the sparsity pattern of a typical burnup matrix is shown given the presence of a neutron flux. The fission product distribution is condensed on the right hand side, rendering the matrix 'almost' upper triangular in structure and behavior. This serves as an advantage when using certain matrix solution methods such as LU factorization.

# CHAPTER 3

# CURRENT TECHNOLOGIES AND METHODOLOGIES

Burnup and depletion codes previously developed or currently in development support numerous options in regards to both solution methodologies and applications for reactor operations and nuclear safeguards. Among the applications currently of interest and being pursued by developers are:

- Calculating the change in eigenvalue over the core lifetime;
- Tracking fissile isotopes as well as major and minor actinides inside of fuel elements;
- Optimizing the utilization of nuclear fuel;
- Studying open and closed fuel cycles;
- Estimating the nuclide concentration at end-of-life fuel cycle for nonproliferation and national security.

This section will focus on the two main methods used to solve the depletion Bateman equations – matrix exponential methods and linear chain methods. This section will also discuss current methods for coupling burnup and transport.

## 3.1 Linear Chain Methods

In terms of utilizing the generalized solutions to burnup problems represented by the Bateman equations described in Chapter 2, linear methods are the most basic and

straightforward to apply. At their core, linear chain methods are based on identifying the relevant nuclide decay chains and calculating the solution for each chain. While simple to implement for a low number of chains, the method becomes more complicated when many chains are involved. An example of such a chain is illustrated in Figure 4.

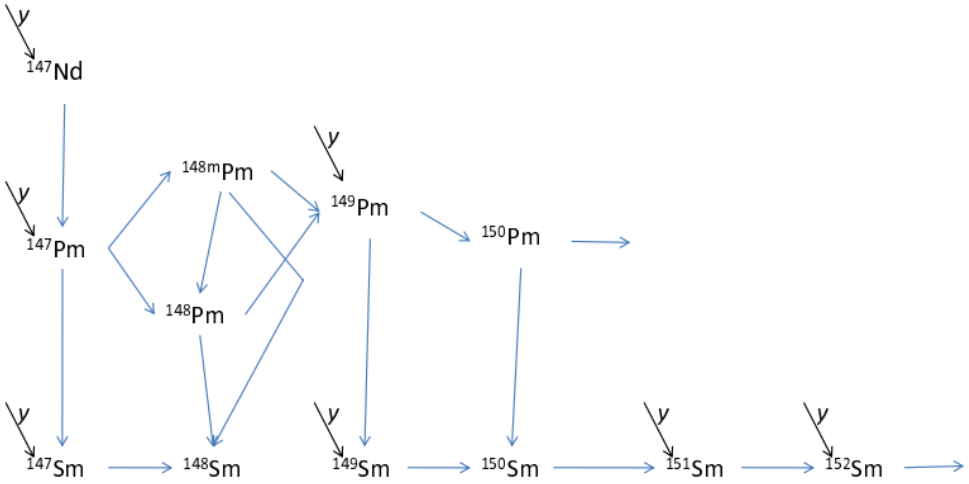


**Figure 4.** Decay chain pathways for Sm [16].

Isolating the pathways of production for $^{150}$Sm, there are 6 unique pathways for $^{147}$Nb to eventually decay into each particular isotope of Sm. Enumerating each of these pathways into linear chains is the most taxing facet of linear chain methods. Figure 5 illustrates the 6 pathways in the previous decay scheme for $^{150}$Sm.

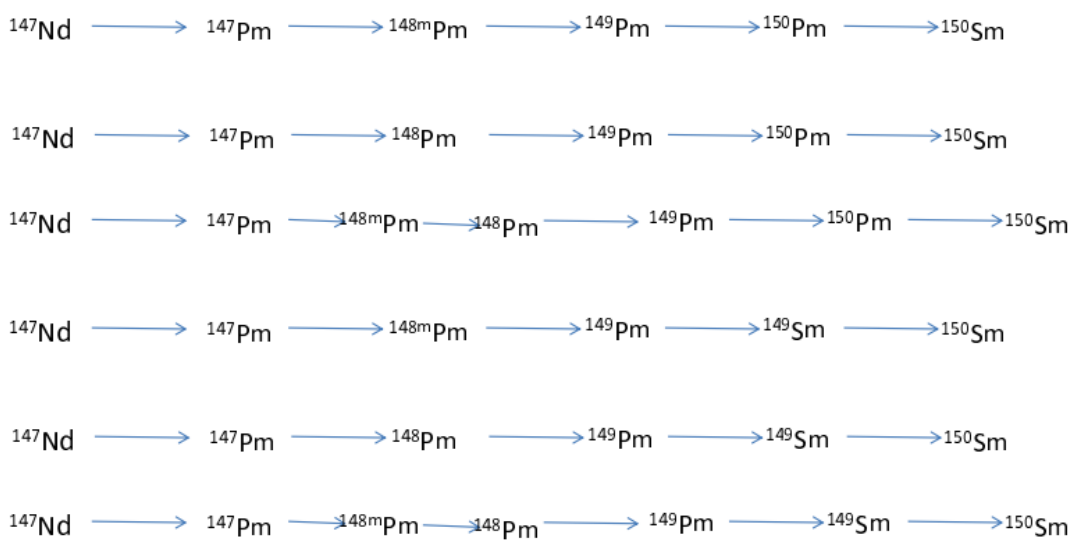


**Figure 5.** Enumerated linear chain pathways for $^{150}$Sm [16].

Given the particular pathways defined by the nuclides being tracked, the Bateman equations outlined in Chapter 2 (Equations 6 – 8) can be utilized appropriately to solve for the final concentration of each nuclide after a prescribed time interval. Once the solution for each linear chain is generated, they can be summed together as a result of the superposition principle.

The primary advantage of a linear chain method is the ability to directly and explicitly find solutions – each linear chain is solved for using the Bateman equations, minimizing the computational overhead. This approach works effectively for a problem involving a low number of nuclides (< 200), but the need to explicitly model each pathway for each nuclide being tracked hinders the extension of these methods to problems involving several hundreds of nuclides.

Some current codes utilizing the linear chain methods are PENBURN (the burnup module associated with PENTRAN), HELIOS, and CINDER'90 [17, 25, 29].

### 3.2 Matrix Exponential Methods

The alternative approach to dealing with burnup problems is to embrace the natural development of a system of ordinary differential equations and use a numerical matrix solver. Historically, matrix exponential methods have been applied with varied success in multiple fuel decay and transmutation codes.

For a chosen set of isotopes, Equation 11 can be set up in matrix notation as follows,

$$N' = \mathbf{A}N \ . \tag{12}$$

$\boldsymbol{A}$ = transition matrix containing coefficients for decay and transmutation
$N$ = nuclide concentration vector, *s.t.* $N = N_i$ for all $i$
$N'$ = first derivative in time of nuclide concentration

The solution can be given in the form of an exponential as follows,

$$N = \exp\left(\boldsymbol{A}t\right) N\left(0\right) \ . \tag{13}$$

$N(0)$ = initial nuclide concentration vector

Consider the solution proposed in Equation 13 – the exponential term can be represented by an infinite series expansion as in Equation 13, and plugged back into the solution in Equation 14.

$$\exp\left(\mathbf{A}t\right) = I + \mathbf{A}t + \frac{\left(\mathbf{A}t\right)^2}{2!} + ... = \sum_{m=0}^{\infty} \frac{\left(\mathbf{A}t\right)^m}{m!} \ . \tag{14}$$

$$N = \left[ \boldsymbol{I} + \mathbf{A}t + \frac{(\mathbf{A}t)^2}{2!} + \frac{(\mathbf{A}t)^3}{3!} + \ldots \right] N(0) \ . \tag{15}$$

Now, a recursion relation can be developed by looking at an arbitrary $N_i$,

$$\begin{aligned} N_i = N_i(0) + t\sum_j a_{ij} N_j(0) + \frac{t}{2}\sum_k \left[ a_{ik} t \sum_j a_{kj} N_j(0) \right] \\ + \frac{t}{3}\sum_m \left[ a_{im} \left( \frac{t}{2}\sum_k \left[ a_{mk} t \sum_j a_{kj} N_j(0) \right] \right) \right] + \ldots \end{aligned} \ . \tag{16}$$

Considering the definition for $N_i$ in Equation 14, the recursion relation below can be applied:

$$\begin{aligned} R_i^0 &\equiv N_i(0) \\ R_i^{n+1} &\equiv \frac{t}{n+1}\sum_j a_{ij} R_j^n \ , \end{aligned} \tag{17}$$

$$N_i = \sum_{n=0}^{\infty} R_i^n \ , \tag{18}$$

Given the construction of the solution in Equation 17, this method only requires the storage of two successive vectors - $R^n$ and $R^{n+1}$ - in addition to the updated value for the solution and the transition matrix [11]. This method of representing the exponential of the matrix as a Taylor series expansion has been the main solver in the ORIGEN for the past several decades [3, 4].

The implementation described in the preceding equations is the most naïve of the matrix exponential methods. Most burnup problems require tracking up to 1000 or so isotopes with a wide range of half-lifes, potentially spanning 30 orders of magnitude. Consequently, the resulting transition matrix has a wide spectrum of eigenvalues, introducing numerical instabilities and false solution convergence. Matrix exponentiation is a valid way to solve Equation 13, but the process used to effectively exponentiate the matrix must handle the difficulties presented by burnup problems [18].

Historically, burnup codes have used this naïve approximation to the exponential of the matrix with some modifications. Generally a valid way to approximate an exponential, the Taylor series expansion method breaks down catastrophically for a matrix with an even modestly wide range of eigenvalues. Some codes attempt to remedy this by using a scaling and squaring method, shown in the following equation,

$$e^{At} = \left( e^{\frac{At}{n}} \right)^n \quad . \tag{19}$$

While using this method does induce the desired effect of minimizing the norm of the matrix multiplied by the time step, it does a poor job of handling isotopes if they are produced at a faster rate than they decay. These methods also suffer from a limitation in terms of length of time step, generally breaking down for any time interval greater than $10^6$ seconds [21].

One of the most recent advancements in matrix exponentiation and burnup solution methodologies is the Chebyshev Rational Approximation Method (CRAM), the primary

burnup solver in the Monte Carlo-based lattice-depletion code SERPENT [15]. CRAM provides multiple advantages, allowing one to compute more accurate solutions with some computational speedup and without the potential breakdowns associated with poorly conditioned matrices. For those reasons, CRAM was chosen as the solver for this study.

# CHAPTER 4

# DEVELOPMENT OF BURNUP SOLVER

Development of the burnup solver in the APIDA code involved studies into multiple linear chain methods and matrix exponential methods. Literature review and implementation of proof-of-concept algorithms led to the conclusion CRAM was the most effective and flexible method for solving burnup equations.

While CRAM, and inherently any numerical matrix solver, handles the requisite transition matrices in burnup problems well, it cannot be applied to nuclides that introduce zeros into the diagonals of said matrices. Consequently, a novel decay chain solver needs to be applied to solve for the time rate of change of nuclide concertation for those particular isotopes.

The APIDA tool is the result of combining the two preceding methods – a hybrid matrix exponentiation and linear chain solver for burnup problems. The following section briefly outlines the theory and foundation of CRAM. For a more complete analysis, the reader is encouraged to refer to the work done by Pusa, et al in references [23].

## 4.1 The Chebyshev Rational Approximation Method (CRAM)

As discussed in earlier sections, the nature of burnup problems and the inherent properties of the isotopes of interest leads to the generation of poorly conditioned matrices. The inclusion of both short- and long-lived isotopes results in a large spectrum of eigenvalues in the problem matrix. Additionally, the time interval for each burnup

calculation has a significant effect on the behavior of the matrix. Historically, these problems have been circumvented by reducing the transition matrix and removing the short-lived or stable isotopes which inflate the norm of the problem (the product of the transition matrix and the time interval). While effective, this method requires computational overhead to check which nuclides must be removed from the matrix and consequently tracked as a decay chain.

The matrices produced in burnup problems exhibit one unique property which can be exploited to produce more accurate answers in a relatively quick time period – the eigenvalues of the burnup matrix have been found to be bounded near the negative real axis [21]. Physically, this is a natural outcome of generating the burnup matrix; the diagonal elements represent the removal coefficients for a particular isotope and are always negative. For methods requiring solutions near the origin (the Taylor series expansion of the matrix exponential), this presents a mathematical hurdle.

CRAM takes advantage of this property and allows burnup matrices to be solved accurately without the removal of short-lived isotopes. Like most burnup matrix exponential solvers, CRAM utilizes an approximation to the given solution for the following system of differential equations.

$$\begin{aligned} N'(t) &= \boldsymbol{A}N(t) \\ N(t) &= \exp(\boldsymbol{A}t)N(0) \end{aligned} \tag{20}$$

Given a rational function capable of approximating the exponential of a value, the matrix exponential can be computed if the approximation is valid in the complex plane

[23]. Applying the Cauchy integral formula, the matrix exponential can be approximated as,

$$\exp(\boldsymbol{A}t) = \frac{1}{2\pi i}\oint_{\Gamma} exp(z)\left(z\boldsymbol{I} - \boldsymbol{A}t\right)^{-1} dz, \tag{21}$$

where $\boldsymbol{I}$ is the identity matrix, $z$ is the independent variable of the rational function $R(z)$ approximating the exponential function, and $\Gamma$ is the closed contour around the spectrum of $\boldsymbol{A}t$.

Using resolvent formalism, one can define the following relation,

$$\left(z\boldsymbol{I} - \boldsymbol{A}t\right)^{-1} = \frac{\boldsymbol{B}(z)}{\det\left(z\boldsymbol{I} - \boldsymbol{A}t\right)}, \tag{22}$$

where the series $\boldsymbol{B}(z)$ is defined as

$$\boldsymbol{B}(z) = z^{n-1}\boldsymbol{B}_0 + z^{n-2}\boldsymbol{B}_1 + \ldots + z^{n}\boldsymbol{B}_{n-2} + \boldsymbol{B}_{n-1} \tag{23}$$

where $\boldsymbol{B}_i$ is a matrix independent of the variable $z$ [28].

Consequently, the eigenvalues of the matrix $\boldsymbol{A}t$ are equivalent to the poles of the rational functions in the formalism of the resolvent. Using the matrix exponential approximation with the Cauchy integral formula, the following holds,

$$\exp(\boldsymbol{A}t)_f = \frac{1}{2\pi i}\oint_{\Gamma} exp(z)R_f(z)dz,$$
$$R(z) = \left(z\boldsymbol{I} - \boldsymbol{A}t\right)^{-1}, \tag{24}$$

where the asymptotic behavior of $R_f(z)$ as z → -∞ is defined $R_f = \mathcal{O}(1)$ and the singularities of $R_f$ correspond to the eigenvalues of the problem matrix $\boldsymbol{A}t$. As mentioned earlier, the eigenvalues are confined to the negative real axis, thus the contour Γ can be extended to the complex plane as a hyperbolic/parabolic function.

Using these properties, the contour integral in Equation 23 can be approximated with rational functions. With the poles and residues of the integral representing the nodes and weights, numerical integration can effectively approximate the matrix exponential.

Defining the rational function with an order *k*, the partial fraction decomposition of the approximation is as follows,

$$r_{k,k}(z) = \alpha_0 + \sum_{j=1}^{k} \frac{\alpha_j}{z - \theta_j}, \tag{25}$$

where $\alpha_0$ is the limit of the rational function as z → ∞ and the residues, $\alpha_j$, correspond to the poles $\alpha_j$. Knowing the poles can form conjugate pairs, the real coefficients and forming the real-valued rational function *r(x)* can be applied as follows,

$$r_{k,k}(x) = \alpha_0 + 2 * \mathrm{Re}\left(\sum_{j=1}^{k/2} \frac{\alpha_j}{x - \theta_j}\right). \tag{26}$$

This real, rational function serves as the approximation to the matrix exponential and provides the solution to the problem proposed in Equation 19 as follows:

$$N(t) = \exp(\boldsymbol{A}t)N(0) = r_{k,k}(x)N(0) = \\ \alpha_0 N(0) + 2 * \mathrm{Re}\left( \sum_{j=1}^{k/2} \alpha_j \left( \boldsymbol{A}t - \theta_j \boldsymbol{I} \right)^{-1} N(0) \right). \quad (27)$$

In practice, any rational function can be used to approximate the exponential. The Chebyshev rational approximation itself is well defined, but the difficulty comes in generating the coefficients ($\alpha_j$ and $\theta_j$) to a precision sufficient enough to provide accurate answers.

### 4.2 Partial Fraction Coefficients (PFD) Generation

The rational function coefficients can be computed via any method but are more practically applied in the form of partial fraction coefficients. These coefficients can be computed directly by solving for the roots of the polynomials, but numerical difficulties arise for higher order approximations [10]. Precomputed coefficients are available in current literature, but even the most widely used sets suffer from round-off errors.

One method for generating these coefficients is through the application of quadrature formulas to the contour integrals over the left complex plane. As highlighted in Equation 23, the computation of the matrix exponential *exp(**A**t)* can be approximated with high accuracy by contour integrals when the eigenvalues of the matrix ***A**t* are confined to the negative real axis. Given the exponential nature of the function, the integral

asymptotically decays as the function approaches -∞, allowing the integral to be approximated by a quadrature set [29].

The work by Weideman presents some useful options for rational approximations to the exponential functions using quadrature rules [29]. One of the simplest and most effective quadrature rules for approximating exponentials are parabolic sets. The analyses done by Weideman includes optimizations for the parameters in each quadrature set, including balancing the error terms of each approximation.

Considering the integral term in Equation 23 illustrating the properties outlined for matrix exponential approximations, any optimized contours can be used. The one proposed by Gallopoulos and used in this work to generate quadrature coefficients is shown below [10].

$$\phi(x) = N\left(0.1309 - 0.1149x^2 + 0.2500ix\right), \tag{28}$$

where $\phi$ is defined from the real plane to the complex plane. This particular parabola is shown to yield a convergence rate of $2.85^{-N}$ [10].

Using the proposed parabola, the exponential can now be approximated using the following rational approximation,

$$r(z) = \sum_{k=1}^{N} \frac{\alpha_k}{z - \theta_k}, \tag{29}$$

where the two coefficients, $\alpha_k$ and $\theta_k$, are defined as

$$\theta_k = \phi(x_k),$$
$$\alpha_k = -\frac{h}{2\pi i} e^{\theta(x_k)} \theta'(x_k). \tag{30}$$

The term *h* in Equation 29 is the discretization length for the quadrature scheme used in approximating the contour integral.

### 4.3 Direct matrix solver using LU factorization

Once the partial coefficients are generated for the rational approximation, the solution to Equation 19 can be generated with a direct matrix solver. For this study, a matrix solver was developed based on a block-LU factorization. A direct solver was chosen over an iterative solver due to the ill-conditioned nature of the matrix and to take advantage of the well-known structure of the burnup matrix. Of the direct methods, the most well-known is LU factorization. Consider the generalized problem,

$$\boldsymbol{A}x = b, \tag{31}$$

and a factorization of $\boldsymbol{A}$ such that $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ is an upper triangular matrix as follows,

$$\boldsymbol{A}x = \boldsymbol{L}\boldsymbol{U}x = b. \tag{32}$$

With the LU factorization available, the problem can now be solved in two steps,

$$\begin{aligned} \boldsymbol{L}y &= b \\ \boldsymbol{U}x &= y. \end{aligned} \tag{33}$$

Solving these two matrix problems is extremely efficient due to the convenient structures of $\boldsymbol{U}$ and $\boldsymbol{L}$. The computational overhead of generating L and U can be minimized by implementing an efficient factorization algorithm, such as the "block" LU algorithm outlined in the MATLAB script in Figure 6.

```
function [L,U] = block_LU(A)
n = size(A, 1); I = eye(n); O = zeros(n);
L = I;  % Identity matrix
U = O;  % Matrix of zeros
    for k = 1:n
        if k == 1
            v(k:n) = A(k:n,k);
        else
            z = L(1:k-1,1:k-1)\A(1:k-1,k);
            size(z);
            U(1:k-1,k) = z;
            v(k:n) = A(k:n,k)-L(k:n,1:k-1)*z;
        end
        if k < n, L(k+1:n,k) = v(k+1:n)/v(k); end
        U(k,k) = v(k);
    end

end
```

**Figure 6.** MATLAB script for the block LU factorization algorithm implemented in APIDA.

The implementation of this algorithm in APIDA differs from the script above as it explicitly handles the complex variables introduced by the Chebyshev rational approximation and uses optimized matrix-vector operations written specifically for burnup matrices. The efficiency of the algorithm in APIDA is remarkable, solving problems

tracking over 1000 isotopes on the order of seconds with partial coefficient orders up to 30.

### 4.4 Linear chain solver for stable nuclides

Burnup problems produce strictly structured and highly sparse matrices. Specifically, the resulting transition matrix containing all of the coefficients corresponding to the set of ordinary differential equations governing the problem can be generated with recursive logic.

For the problem presented below,

$$N'(t) = \boldsymbol{A}N(t), \tag{34}$$

where $N$ is the vector of nuclide concentrations and $A$ is the transition matrix holding all the relevant coefficients. The elements of $A$ are defined as follows,

$$a_{ij} = \begin{cases} l_{ij}\lambda_j + \left( f_{ij}\sigma_j^f + nn_{ij}\sigma_j^{nn} \right)\phi , & i \neq j \\ -\left( \lambda_i + \sigma_i\phi \right) , & i = j \end{cases} , \tag{35}$$

where, $l_{ij}$ is the decay branching ratio of nuclide $j$ to nuclide $i$, $\lambda_j$ is the disintegration constant of nuclide $j$, $f_{ij}$ is the yield fraction of the fission of nuclide $j$ yielding nuclide $i$, $\sigma_j^f$ is the fission cross section of nuclide $j$, $\sigma_j^{nn}$ is the cross section for non-fission neutron reactions for nuclide $j$, $nn_{ij}$ is the branching ratio of a non-fission neutron reaction with

nuclide $j$ yielding nuclide $i$, $\sigma_j$ is the total cross section of nuclide $j$, and $\phi$ is the neutron flux.

Observing the elements along the diagonal matrix, there are scenarios in which the condition number of $A$ could be cumbersome or even infinite in the case of an extremely small magnitude along the diagonal or a zero element (i.e. a stable isotope with little to no reactions). In this case, the transition matrix needs to be reduced – the row and column associated with the isotope in question must be removed from the matrix. This allows the transition matrix to be full and amenable to exponentiation. Consequently, the isotopes reduced from the transition matrix need to have their final concentrations solved for in a different manner.

In order to find the final concentrations for the reduced nuclides, a linear chain method was utilized in APIDA. For each nuclide not included in the transition matrix, solutions provided by the appropriate Bateman equations are applied to each nuclide. For cases where there is no irradiation and no external sources of production of nuclide $N_i$, the following equation is used,

$$N_i(t) = N_1^0 \lambda_1 \lambda_2 \ldots \lambda_{i-1} \sum_{j=1}^{i} \frac{e^{-\lambda_j t}}{\prod_{\substack{k=1 \\ k \neq j}}^{i} (\lambda_k - \lambda_j)} + N_i^0, \tag{36}$$

where $\lambda_j$ is the disintegration constant of nuclide $j$ and $N_1^0$ is the initial concentration of the first parent nuclide of the chain.

In the case of neutron irradiation and the production of nuclide $N_i$ with precursors, the following equation is used,

$$N_i(t) = \sum_{l=1}^{i-1} \left[ N_l^0 \eta_l \eta_{l+1} \ldots \eta_{i-1} \sum_{j=1}^{i} \frac{e^{-\mu_j t}}{\prod_{\substack{k=l \\ k \neq j}}^{i} (\mu_k - \mu_j)} \right] + N_i^0, \tag{37}$$

where $\eta_i$ is the chain-linking precursor decay constant of nuclide $N_i$ (including decay from other nuclides and reaction transmutation) and $\mu_i$ is the effective decay constant of nuclide $i$ (total removal rate including decay and reactions).

In both cases described above, the branching ratio calculator in the APIDA code is utilized to cycle backwards through each nuclide's decay scheme and enumerate the linear chains for each chain. Expectedly, the resulting set of decay constants and reaction rates are strongly associated with the coinciding coefficients in the transition matrix.

# CHAPTER 5

# API DEVELOPMENT FOR APIDA

Traditionally, lattice depletion codes have been some transport solution method of choice (discrete ordinates, Monte Carlo, collision probability method, etc.) and an associated burnup solver specifically designed and tailored for the prescribed neutronics code. Historically, these the methods are inextricably linked – specifically, the burnup solver cannot be extracted and implemented with another transport solver. Widely used lattice depletion codes like HELIOS, SERPENT, CASMO, and PARAGON all use self-developed depletion modules virtually impossible to utilize without their associated transport codes [15, 26]. This presents numerous hurdles, chiefly the inability to independently validate the burnup solver and the impracticality of attempting to integrate the module with a novel transport solution method. APIDA presents a significant step in providing a universal burnup solver capable of integration with any code via a simple Application Programming Interface (API).

## 5.1 Communication between transport solver and burnup module

The basic procedure of any lattice depletion method is first to determine the physical characteristics (flux) at one discrete time with a transport solver, and then to use that information to calculate the generation, destruction, and change in nuclide concentrations over one or more prescribed time steps with a burnup solver. Once the new nuclide concentrations in the fuel have been calculated, the information is relayed back to the transport solver to calculate the new physical characteristics (flux, eigenvalue, etc.).

Efficient iteration between the transport and burnup modules in a lattice depletion code is key to minimizing its computation time. Historically, these modules were coupled using text files. In essence, the two codes were treated as "black boxes" in relation to one another. Figure 7 illustrates a typical iterative cycle between a transport solver and a burnup module.

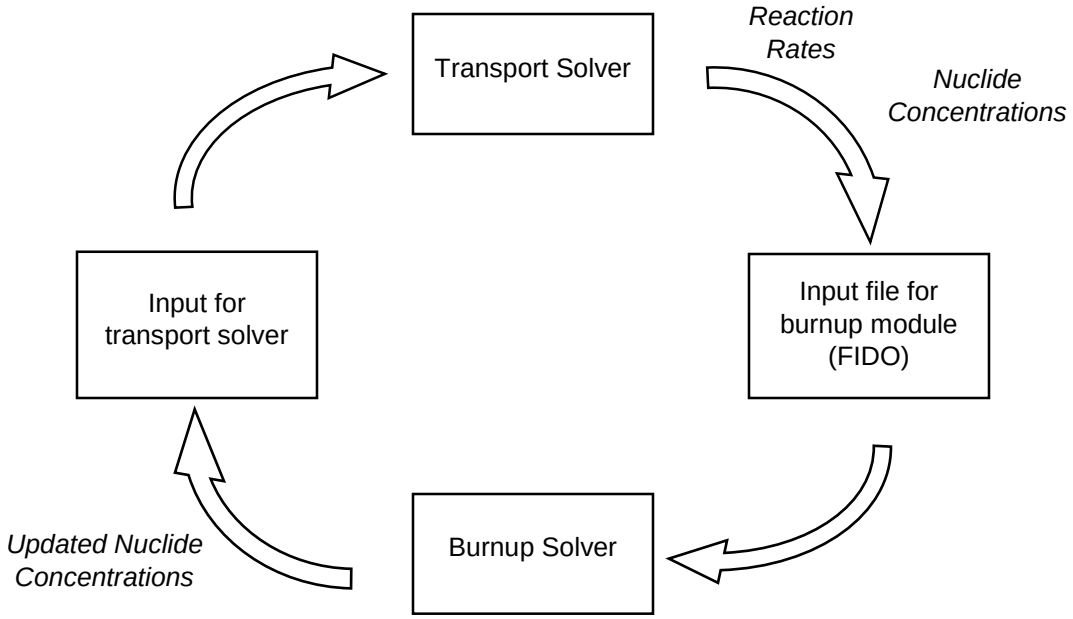


**Figure 7.** Flowchart of communication between transport solver and burnup module using text inputs.

While this method has allowed for relatively seamless independent code development, it presents some serious problems in modern implementations. The most prominent issue this setup presents is difficulty with large-scale embedding. Cluster computing has become an integral part of reactor modeling, especially in regards to multi-physics applications. The need for high performance computing has forced developers to abandon the model in Figure 7, since the "black box" model requires writing an input to

the hard disk. The preferred option in coupling two codes is to do so “in-memory” with the use of an Application Programming Interface (API). Rather than generating text inputs to be read by both of the code modules, an API allows the “parent” program to call upon the secondary code module in-memory without the need to stop and generate a text file or touch the hard disk.

### 5.2 Object-oriented framework in APIDA

APIDA is a code written in the C++ programming language and contains numerous features for users interested in all aspects of burnup calculations. All the features are housed within classes, each with their own API suited to the particular information being retrieved.

The key to the portability and utility of APIDA in terms of seamless integration with other codes is the object-oriented framework of the API. The data pertinent to the burnup problem is stored in exposed data containers where the user only needs to provide the data necessary to run a calculation. Programming these structures in C++ allows for these objects to be instantiated simultaneously in a massively parallel environment (multiple processors over multiple nodes). The C++ programming language allows for memory to be allocated on the heap, making it possible to instantiate multiple instances without threatening the integrity of the data with overwrites.

Discussed in the next section, the APIDA code is separated into multiple objects representing the important variables in a burnup calculation. The data for each important variable is encapsulated within different objects available to the user with “get” and “set”

functions. Each object, or "class," contains multiple functions publicly available to the user, allowing for the manipulation of private data.

### 5.3 Classes and features in APIDA

As mentioned in the previous section, the APIDA framework is broken down into multiple classes, each responsible for handling the requisite steps for a burnup calculation. The main classes inherently instantiated for a burnup calculation are the following:

- 'apida.h' – container for the front-end user API.
- 'library_builder.h' – container for library data (decay and fission yields).
- 'tran_mat.h' – container for transition matrix.
- 'depletion.h' – container for solver methods for burnup problems.
- 'output_proc.h' – container for output processing methods.

Each class contains a suite of methods used to handle the operations required by each step of a burnup calculation. The 'apida' class serves as the front-end interface, allowing user to run a calculation with only the data absolutely required to complete a calculation – the list of nuclides being tracked, their corresponding initial concentrations, the reaction rates of each type by nuclide, and the time step. If desired, users can access burnup data explicitly (half-lifes, fission yields, decay modes, etc.) via the methods available in each class. Each class, along with their corresponding methods, are described in Appendix A.

### 5.4 Implementation of APIDA

The purpose of APIDA is to provide a simple and straightforward way to incorporate a burnup solver using the C++ interface. Future work for APIDA includes incorporating compatibility with more programming languages (C, FORTRAN, etc.) using "wrappers" to make the code interface amenable to other languages.

The implementation of an API is non-trivial for users inexperienced in coupling codes – consequently, a simple example of the use of APIDA to solve a burnup problem is shown below in Figure 8. Benchmark problem #6 (described in Chapter 6) follows the fission and capture of $^{238}U$ to produce $^{135}Xe$ and several isotopes of Pu. In order to run the calculation, the steps are as follows:

1. Initialize the instance of the 'apida' class.
2. Provide a vector containing the ZAID of the isotopes to be tracked.
3. Provide a vector containing the initial concentrations of the isotopes to be tracked.
4. Provide a vector with the time steps (cumulative) for the calculation.
5. Provide the reaction rates for each isotope (description given in comments of code example).
6. Initialize the library to gather data for the calculation.
7. Set the initial concentrations, time steps, and reaction rates.
8. Run the calculation.
9. Retrieve the final concentrations for each nuclide at each time step.

APIDA contains numerous other features to allow for more robust calculations or to access specific data for other applications. A description of publicly available functions in certain APIDA classes is available in Appendix A.

```
#include <vector>
#include "apida/apida.h"

using namespace std;

int main(int argc, char** argv) {

    // initialize apida class
    apida* case1 = new apida;

    vector<int> nuclides; // nuclides
    vector<double> conc_i; // initial concentrations

    nuclides.push_back(922380); // U238
    nuclides.push_back(922390); // U239
    nuclides.push_back(932390); // Np239
    nuclides.push_back(942390); // Pu239
    nuclides.push_back(942400); // Pu240
    nuclides.push_back(531350); // I135
    nuclides.push_back(541350); // Xe135
    nuclides.push_back(551350); // Cs135

    conc_i.push_back(1E12); // U238
    conc_i.push_back(0); // U239
    conc_i.push_back(0); // Np239
    conc_i.push_back(0); // Pu239
    conc_i.push_back(0); // Pu240
    conc_i.push_back(0); // I135
    conc_i.push_back(0); // Xe135
    conc_i.push_back(0); // Cs135

    vector<double> time; // time steps

    time.push_back(0);
    time.push_back(1E6);

    vector< vector<double> > rxns;
    vector<double> row_hold(10, 0.0); // 10 types of rxns
//   /* column neutron-nuclide interactions (n_i to n_j)
//    * 1. (n,gamma)  (+000010)
//    * 2. (n,2n)     (-000010)
//    * 3. (n,3n)     (-000020)
//    * 4. (n,4n)     (-000030)
//    * 5. (n,p)      (-010000)
//    * 6. (n,d)      (-010010)
//    * 7. (n,t)      (-010020)
//    * 8. (n,He-3)   (-020020)
//    * 9. (n,alpha)  (-020030)
//    * 10. Fission
//    */
```

**Figure 8.** Example input using APIDA to solve Benchmark Problem #6.

```
    // ------------------------------------------------------------------
    // START OF API FUNCTIONALITY
    // initialize the library and read in decay + fission yield data
    case1->initialize_library(nuclides, conc_i, Avg_FE);

    // set the initial concentrations
    case1->set_initial_concentrations(conc_i);

    // set times
    case1->set_times(time);

    // set reaction rates
    case1->set_rxn_rates(rxns);

    // run the burnup calculation
    case1->run();

    vector< vector<double> > final_conc;

    // retrieve the new concentrations
    case1->get_concentrations(final_conc);

    // ----> send new concentrations to transport code

    return 0;
}
```

**Figure 8 continued**

### 5.5 Coupling APIDA and COMET

As highlighted in Chapter 1, the motivation for this work was to expand the utility and capability of the COMET method. Extensively validated to provide accurate and efficient numerical transport solutions to whole core reactor problems, extending the capabilities of COMET to include depletion at the assembly level (and eventually the whole core level) would provide an invaluable resource in terms of reactor design and fuel cycle analysis.

Section 5.1 includes a discussion of the general communication between a transport solver and a burnup module, but this section will expound on the specific information needed in a COMET calculation to perform a lattice depletion burnup step.

The COMET method takes advantage of the natural structure of a modern nuclear reactor core – a lattice of square assemblies organized into some Cartesian geometry. The basis of COMET is the generation of incident flux response function coefficients for unique coarse meshes – fuel assemblies are generally modeled for each coarse mesh. For each coarse mesh, a fixed source calculation (with the fission source scaled by $1/k_{eff}$) is performed. The boundary condition is an incident neutron flux with a phase space distribution that is the tensor product of a delta function in energy and Legendre polynomials in space (x, y) and direction (azimuthal and polar angles) on the mesh boundary. Historically, the energy variable has been treated discretely similar to multigroup theory but recent work has been done to expand the energy treatment into the continuous regime.

Since the boundary conditions are not known a priori, vacuum boundaries in the fixed source calculations are used to pre-compute the response function expansion coefficient library needed to perform an iterative deterministic sweep to find the core solution (e.g., $k_{eff}$ and the pin fission density distribution in the entire core) for an arbitrary arrangement of the unique coarse meshes in the core. For fuel coarse meshes, the response functions depend on the core eigenvalue ($k_{eff}$) which is not known a priori. As a result, the response library is generated for a grid of $k_{eff}$. Recent work has been implemented in COMET using a new method which does not require interpolation of the library as well [33]. The truncation of Legendre expansions and the interpolation in $k_{eff}$ are the only

approximations in COMET. For a more thorough description of the coarse mesh transport method consult the work done by Zhang and Rahnema [32].

Traditionally, COMET has been employed for whole core problems with the assemblies acting as the coarse meshes. For the purposes of this study, pin cells were chosen for the coarse meshes in order to conduct assembly-level lattice depletion problems. Using a smaller volume for a coarse mesh introduces difficulties in terms of statistics when using stochastic methods to generate the response functions. Recent work done by Hon to develop a Monte Carlo based response function generator shows promise in terms of quickly pre-computing response functions [13]. The code developed by Hon, a Stochastic Particle Response Calculator (SPaRC) is another step being taken to make COMET amenable to coupling with APIDA.

In order to provide APIDA the necessary reaction rates, specifically the neutron-induced reactions in Table 3, response functions need to be generated for each type of reaction. Once the deterministic sweep is performed to generate the final solution, the currents for each coarse mesh and each surface are used with the pre-computed reaction-dependent response functions to generate the reaction rates. Note that these reaction rates need to be given for each isotope; consequently, response functions must be generated and tracked for each isotope as well.

Once these reaction rates are given, they are scaled according to the power and used by APIDA to generate new material concentrations. These new material concentrations are then used to generate new response functions and the calculation can be looped until the final burnup step. A flowchart outlining the general communication and order of operations in a COMET-based lattice depletion calculation is shown in Figure 9.

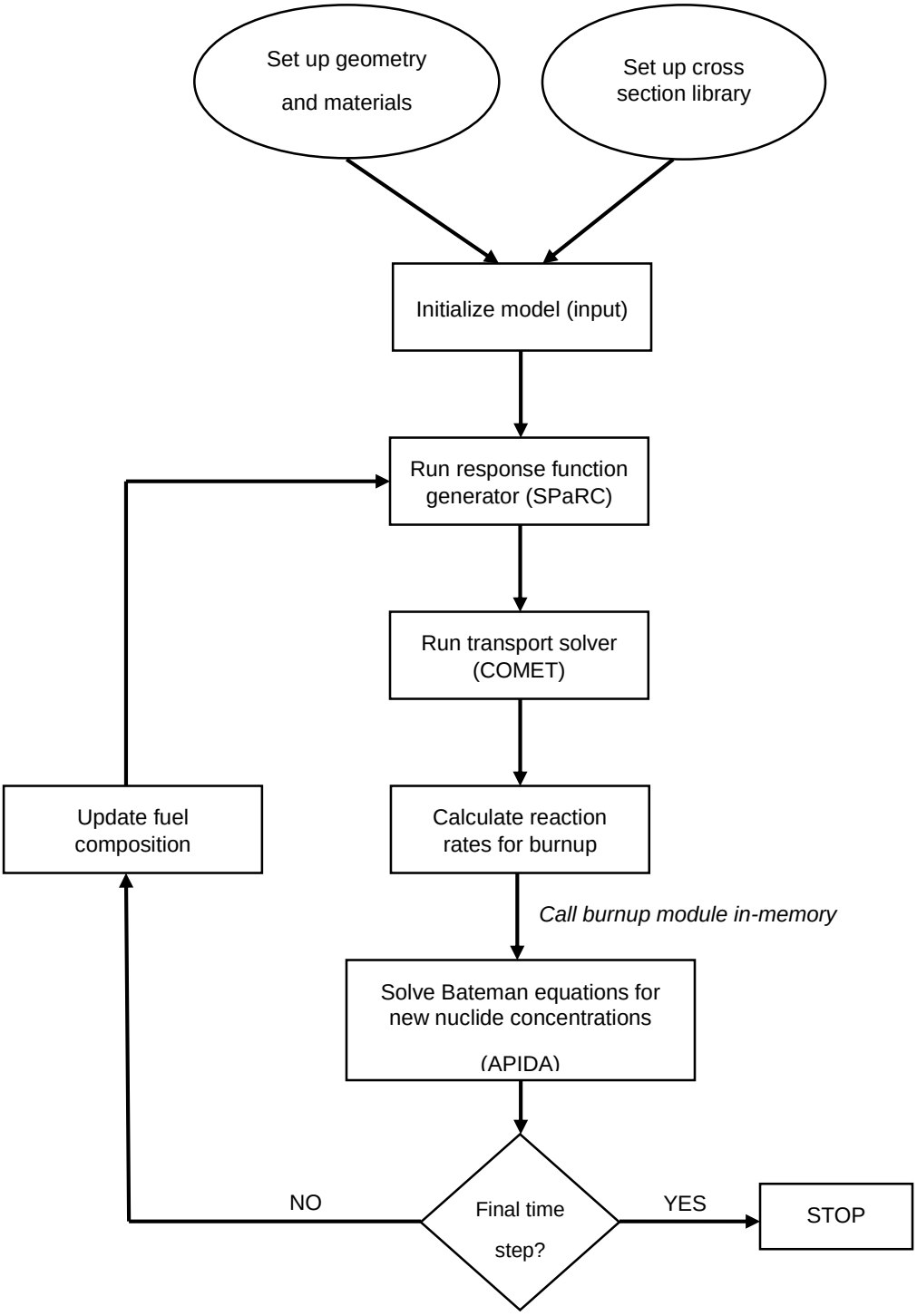


**Figure 9.** Flowchart of COMET-burnup coupling.

# CHAPTER 6

# DESCRIPTION OF BENCHMARK PROBLEMS

In order to validate the methods in APIDA, the burnup solver was benchmarked against several decay chains solved using analytical methods. Once verified for explicit decay chains, APIDA was then applied to lattice depletion problems and benchmarked for a pin cell case.

## 6.1 Analytical benchmarks with Mathematica

Initial benchmarking of APIDA was done with analytical solutions generated in the Wolfram Alpha tool Mathematica [30]. Numerous decay chains were chosen, with and without reactions and fission yields, over a wide range of decay probabilities. The problems were chosen with the purpose of challenging the methods in APIDA to ensure accuracy of solutions for any given set of isotopes. The proceeding tables and figures describe the burnup problems model by APIDA and validated in Mathematica. The corresponding results are in Chapter 7.

Benchmark problem #1 is a simple decay scheme – $^{238}U$ alpha decays into $^{234}Th$, which then itself decays. While lacking complexity, this decay scheme tests the methods in APIDA due to the wide range of eigenvalues in the transition matrix.

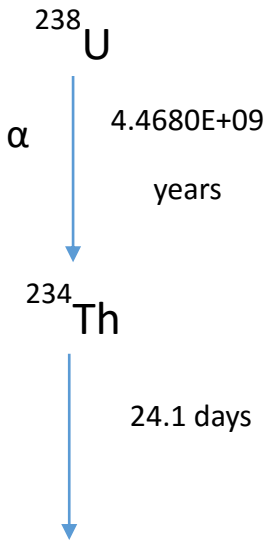


**Figure 10.** Description of Benchmark #1.

Benchmark problem #2 is another fairly simple decay scheme, but now introduces two types of decay – alpha decay and beta decay.

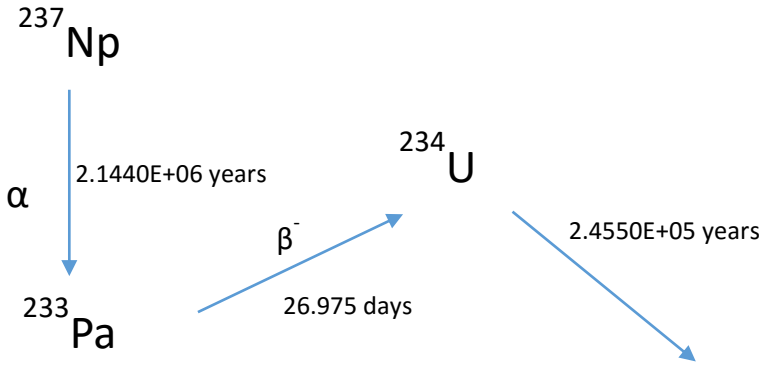


**Figure 11.** Description of Benchmark #2.

Benchmark problem #3 is similar to the first two benchmarks, but the introduction of a stable isotope ($^{207}$Pb) tests the linearized chain method in APIDA used to solve for the concentrations of stable isotopes.

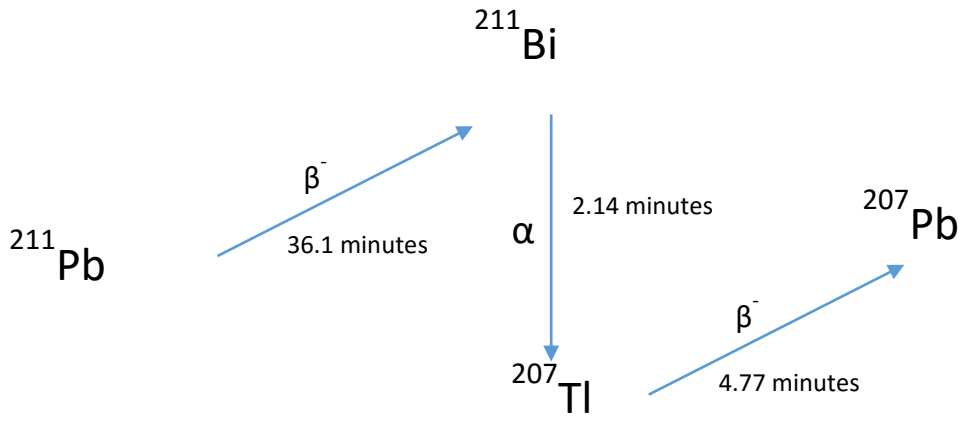


**Figure 12.** Description of Benchmark #3.

Benchmark problem #4 follows the production of $^{237}$Np via $^{235}$U. This problem includes the decay of each isotope, but now incorporates reaction rates – specifically the (n,γ) reactions involved in producing $^{237}$Np. This method tests the reaction rate branching ratio calculator in APIDA.

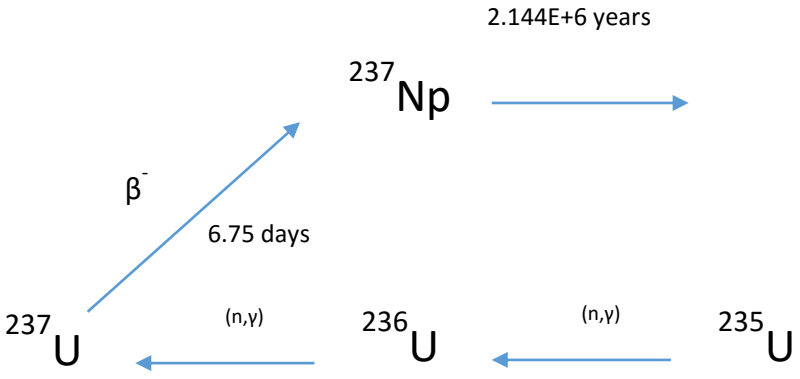


**Figure 13.** Description of Benchmark #4.

Benchmark problem #5 is a complex actinide chain following the decay and transmutation of $^{238}U$. This problem includes a wide range of decay probabilities, reaction rates, and a closed decay loop from $^{244}Cm$ to $^{240}Pu$.

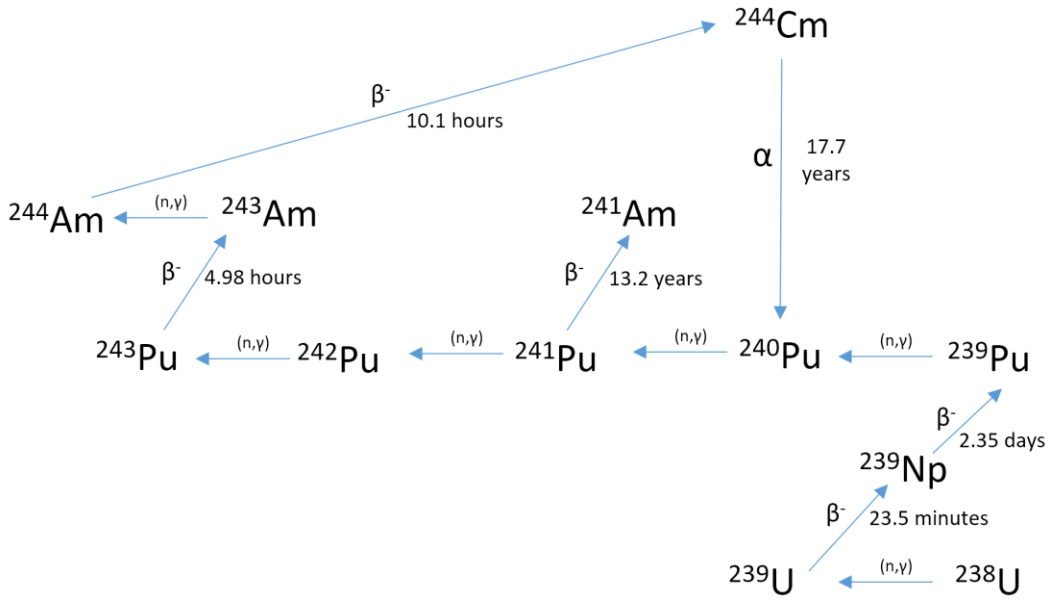


**Figure 14.** Description of Benchmark #5.

The final analytical benchmark problem follows the production of a fission product important to reactivity – $^{135}Xe$. This problem tests the fission yield calculator in the APIDA code.

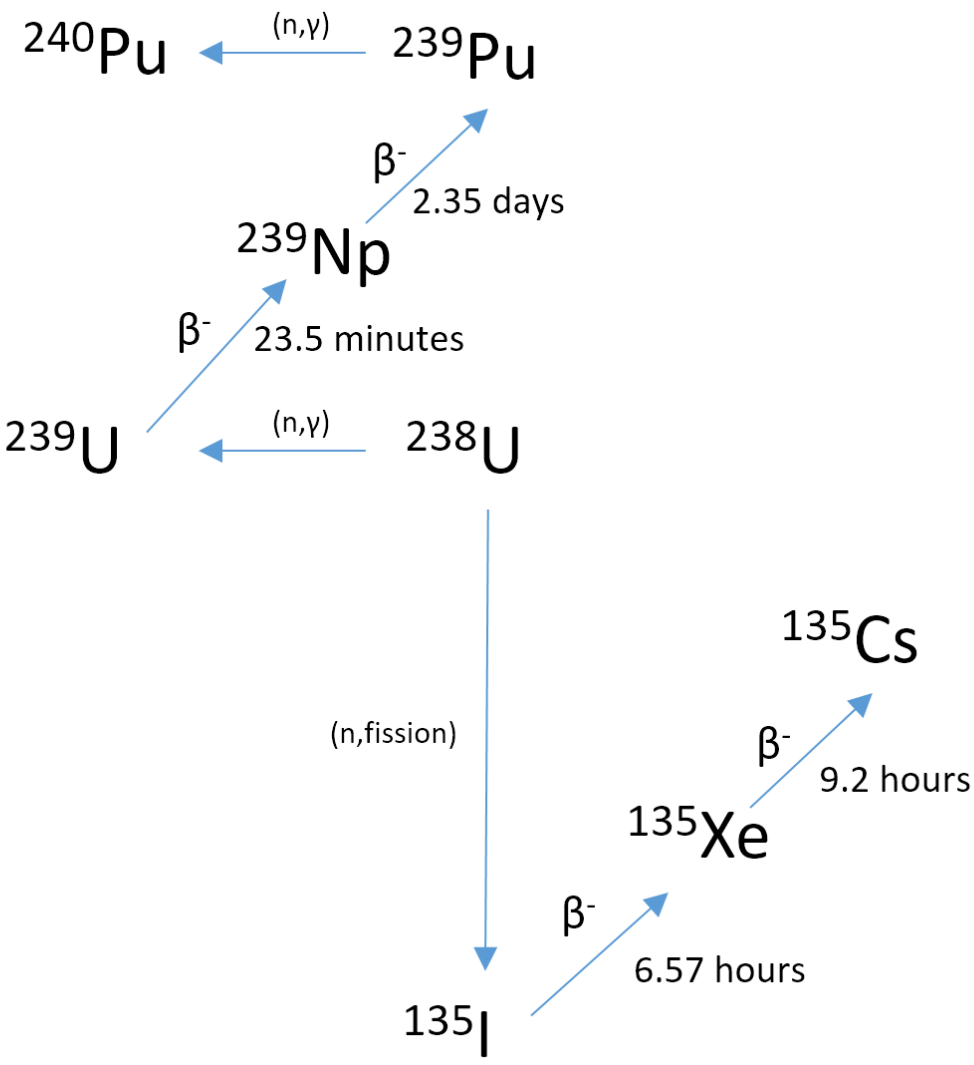


**Figure 15.** Description of Benchmark #6.

### 6.2 Lattice depletion pin cell benchmark

As a first order application to test the methods in APIDA, a single fuel pin cell was modeled and depleted. Two fuel pin cells were modeled – one without burnable absorber and one with gadolinium integrated into the fuel. The parameters for each pin cell were based on the AREVA European Pressurized Reactor design [1, 2]. It is a standard

pressurized water reactor (PWR) fuel pin design – a cylindrical fuel pin surrounded by zirconium-based cladding with a small gap, encapsulated in a moderator (water) with boron. The fuel pin design is illustrated in Figure 16 and the geometric parameters for the pin cell are shown in Table 4.

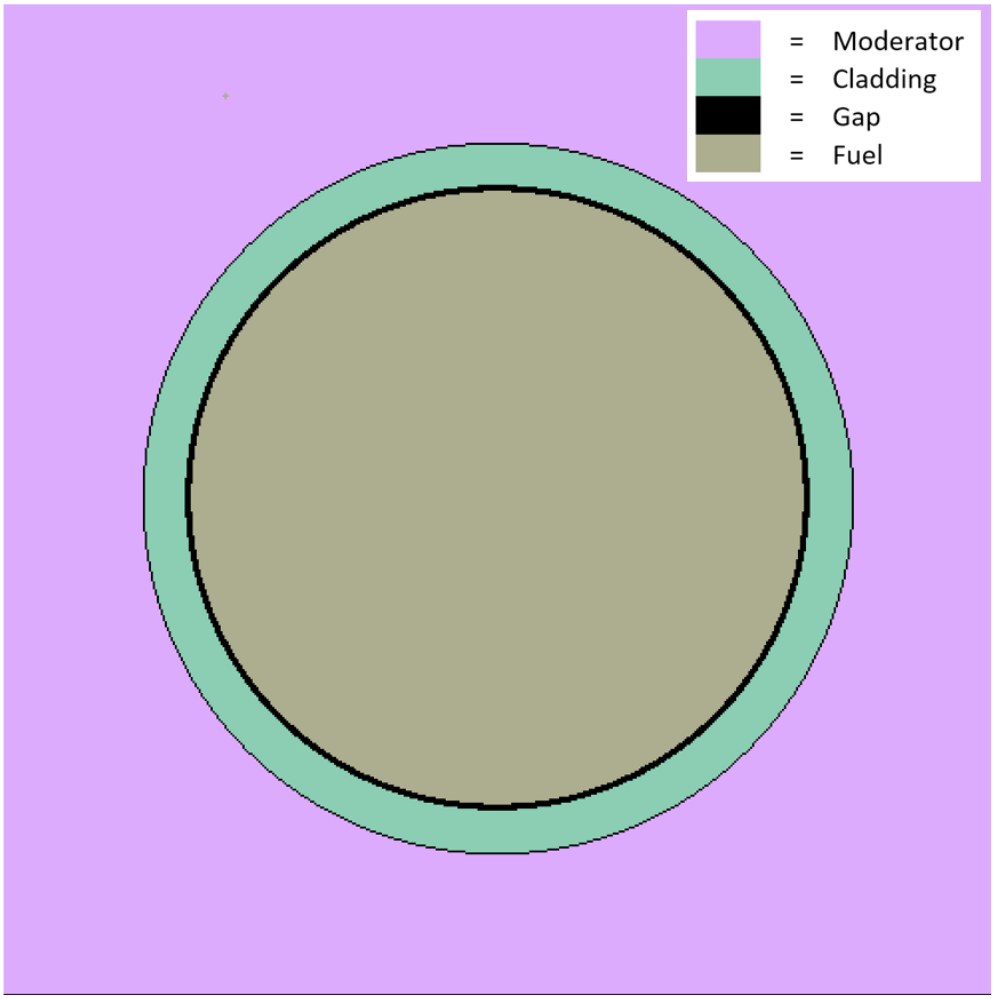


**Figure 16.** Illustration of fuel pin cell depletion problem.

**Table 4.** Geometric parameters for fuel pin cell.

| | |
|---|---|
| Pin Pitch | 1.259840 cm |
| Cladding Outer Radius | 0.474980 cm |
| Cladding Inner Radius | 0.417957 cm |
| Fuel Pellet Radius | 0.409575 cm |
| Boron Concentration | 1000 ppm |
| Moderator Density | 0.7 g/cm$^3$ |
| Fuel Density | 10.4 g/cm$^3$ |
| Fuel Temperature | 900 K |
| Moderator Temperature | 600 K |

Two types of fuel pins were modeled – one with $UO_2$ enriched to 3.5wt% and another with integrated fuel burnable absorber in the form of $Gd_2O_3$. The fuel parameters for each pin cell problem are shown in Tables 5 and 6.

**Table 5.** Fuel parameters for $UO_2$ pin cell.

| | |
|---|---|
| Fuel Composition | $UO_2$ |
| Fuel Enrichment | 3.5 wt% $^{235}U$ |
| Fuel Density | 10.4 g/cm$^3$ |
| Cladding Composition | Natural Zr |
| Cladding Density | 6.514 g/cm$^3$ |
| Moderator Density | 0.7 g/cm$^3$ |
| Soluble Boron Concentration | 1000 ppm |

**Table 6.** Fuel parameters for pin cell with gadolinium.

| | |
|---|---|
| Fuel Composition | $UO_2$ + $Gd_2O_3$ |
| Fuel Enrichment | 2.27 wt% $^{235}U$ |
| Gad Enrichment | 8.0 wt% $Gd_2O_3$ |
| Fuel Density | 10.4 g/cm$^3$ |
| Cladding Composition | Natural Zr |
| Cladding Density | 6.514 g/cm$^3$ |
| Moderator Density | 0.7 g/cm$^3$ |
| Soluble Boron Concentration | 1000 ppm |

In regards to isotopics, a sensitivity analysis was conducted using the fuel pin cell problem to determine the proper number of nuclides to track during a depletion calculation in order effectively capture the physics of transmutation in a reactor core. By default, the SERPENT code tracks 1094 isotopes regardless of how many nuclides the user specifies in the input file. While all encompassing, this may not be necessary for all calculations and could potentially lead to numerical instability in both depletion and transport calculations if material concentrations are too low.

For the sensitivity analysis, a 'first-step' investigation was conducted to see how the number of isotopes being tracked affected the change in eigenvalue over one burnup step, in this case 250 MWD/MTU. The SERPENT calculation was run with the codes built-in burnup solver to produce a reference solution for the first burnup step. Then the transport solution from the initial steady state calculation in SERPENT was used to run a burnup calculation for the first step using APIDA. The APIDA calculation was run several times,

each with a different number of isotopes ranging from 250 to 1049. Once a reasonable number of isotopes was determined, APIDA was then coupled with the transport solver in SERPENT to run fuel pin cell lattice depletion calculations over several time steps.

In addition to the results generated by SERPENT and the results generated by coupling APIDA to serpent, HELIOS was also used to produce pin cell depletion results for further comparison. HELIOS is an extensively validated lattice depletion code utilizing method of characteristics (MOC) and collision probability (CPM) solvers in 2D general geometry for transport solutions and a linearized chain method for burnup. An illustration of the fuel pin cell model in HELIOS is shown in Figure 17.

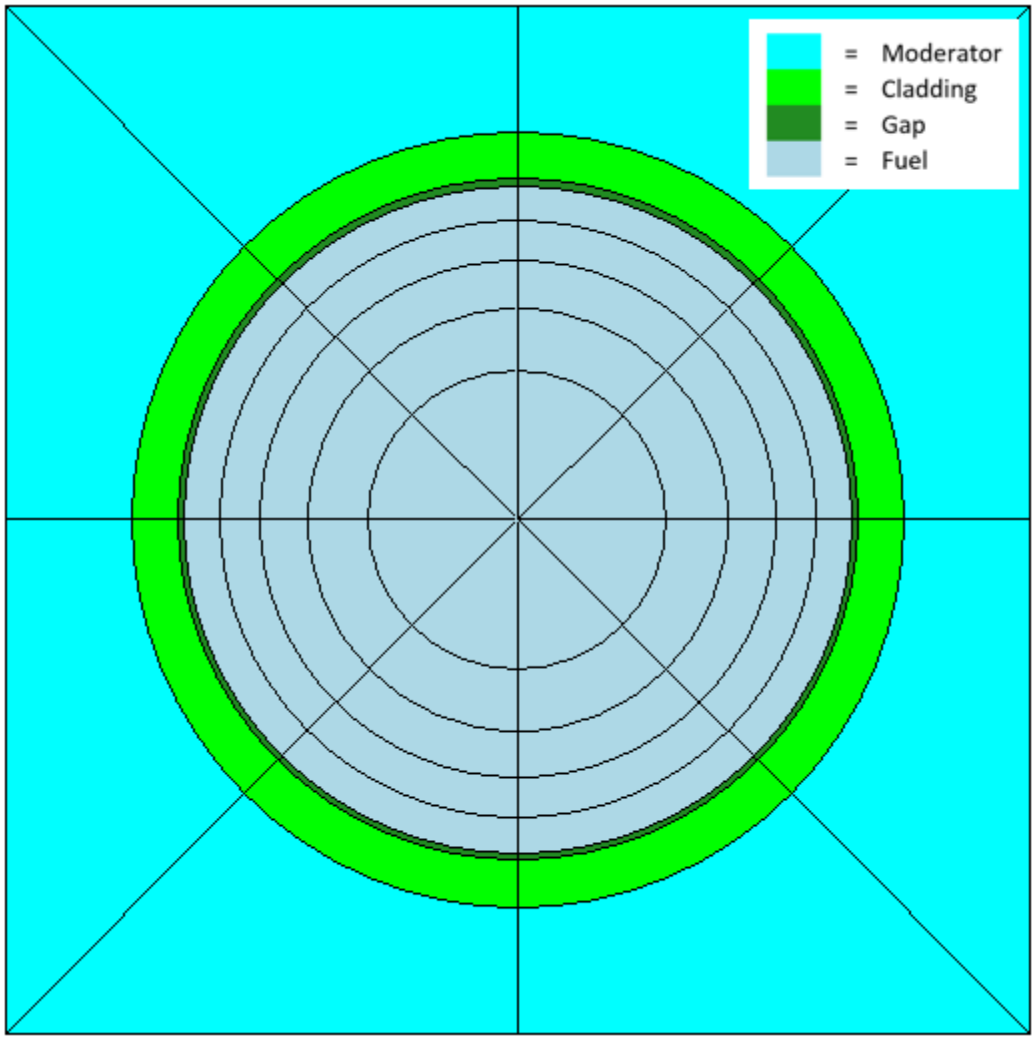


**Figure 17.** Geometry of fuel pin cell in HELIOS.

# CHAPTER 7

# RESULTS

## 7.1 Analytical benchmarks with Mathematica

The results from the analytical benchmark problem described in Chapter 6 validated the APIDA method in terms of solving well-defined decay and transmutation schemes. Tables 7, 8, and 9 show the percent relative error in the final concentration generated by APIDA for each benchmark. The mean weighted error is defined below.

$$MWE = \frac{\sum_{i}^{N} \left| RE_i \right| \cdot FD_i}{N \cdot FD_{avg}} \tag{38}$$

**Table 7.** APIDA results compared to Mathematica solutions for benchmarks #1 and #2.

| **Benchmark #1** | | **Benchmark #2** | |
|---|---|---|---|
| | **Relative Error (%)** | | **Relative Error (%)** |
| **U-238** | 0.0058 | **Np-237** | 8.02E-13 |
| **Th-234** | 0.0034 | **Pa-233** | 5.76E-13 |
| | | **U-234** | 3.54E-07 |

The results from Benchmark #1 exhibit the largest errors for material concentrations of all the benchmarks when compared to the analytical solution produced in Mathematica, but they are still acceptable by a significant margin. The errors produced in benchmark #1 are inherent in all numerical solvers, as each benchmark was also validated using the intrinsic

matrix exponential function in MATLAB. The mean weighted error for Benchmark #1 is 5.84E-3% and the mean weighted error for Benchmark #2 is 3.38E-9%.

**Table 8.** APIDA results compared to Mathematica solutions for benchmarks #2 and #3.

| Benchmark #3 | | Benchmark #4 | |
|---|---|---|---|
| **Relative Error (%)** | | **Relative Error (%)** | |
| **Pb-211** | 9.91E-12 | **U-235** | 5.84E-05 |
| **Bi-211** | 7.31E-07 | **U-236** | 3.12E-05 |
| **Tl-207** | 2.76E-01 | **U-237** | 6.37E-08 |
| **Pb-207** | 8.92E-03 | **Np-237** | 1.08E-08 |

For benchmarks #3 and #4, the relative percent errors remain extremely low and more than acceptable. The linear chain solver for stable isotopes in APIDA is also shown to be effective, solving for the final concentration of the stable nuclide Pb-207. The mean weighted error for Benchmark #3 is 1.92E-3% and the mean weighted error for Benchmark #4 is 1.18E-7%.

**Table 9.** APIDA results compared to Mathematica solutions for benchmarks #5 and #6.

| Benchmark #5 | | Benchmark #6 | |
|---|---|---|---|
| **Relative Error (%)** | | **Relative Error (%)** | |
| **U-238** | 5.12E-06 | **U-238** | 1.27E-04 |
| **U-239** | 5.21E-06 | **U-239** | 1.27E-04 |
| **Np-239** | 2.52E-09 | **Np-239** | 2.38E-09 |
| **Pu-239** | 5.96E-09 | **Pu-239** | 1.34E-08 |
| **Pu-240** | 1.28E-08 | **Pu-240** | 1.41E-08 |
| **Pu-241** | 3.21E-09 | **I-135** | 1.48E-07 |
| **Pu-242** | 4.50E-06 | **Xe-135** | 2.19E-10 |
| **Pu-243** | 1.54E-05 | **Cs-135** | 3.59E-11 |
| **Am241** | 5.59E-06 | | |
| **Am-243** | 2.91E-06 | | |
| **Am-244** | 9.76E-07 | | |
| **Cm-244** | 5.01E-07 | | |

The relative percent errors remain remarkably low for benchmark problems #5 and #6, validating the capabilities in APIDA to incorporate reaction rates and to properly apply fission yield fractions. In benchmark #6, the production of Xe-135 is of particular importance to reactor operations as it is a strong neutron absorber. The mean weighted error for Benchmark #5 is 4.65E-8% and the mean weighted error for Benchmark #6 is 8.27E-9%.

In order to validate the capability of APIDA to solve for nuclide concentrations over multiple time steps, benchmark problems #4 and #5 were solved over a number of time intervals. Plots showing the APIDA solutions imposed on the analytical solutions are shown in Figures 18 and 19.

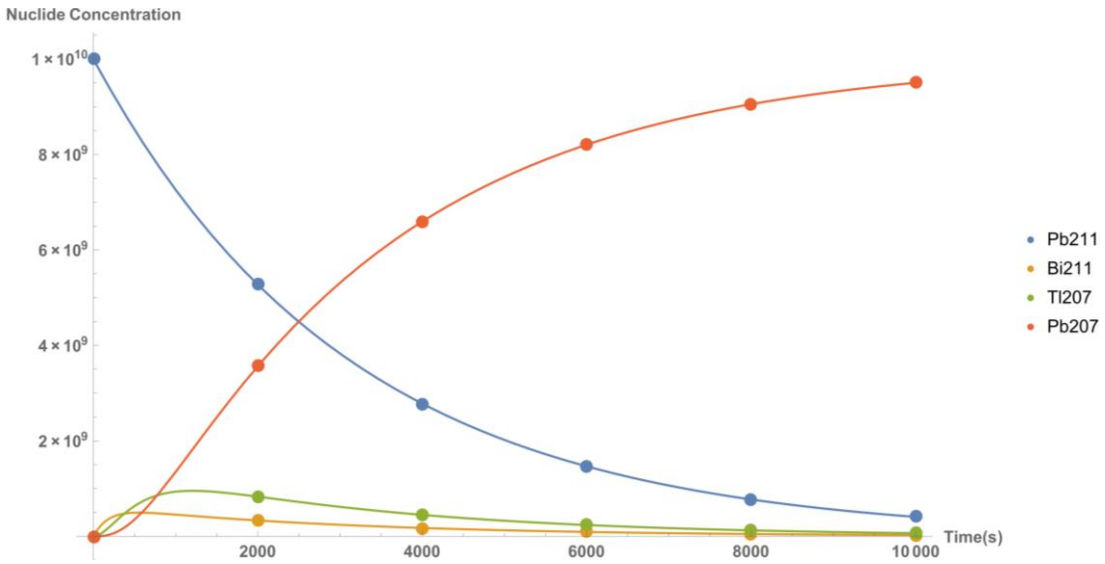


**Figure 18.** APIDA solution points plotted over the analytical solution to benchmark #4.

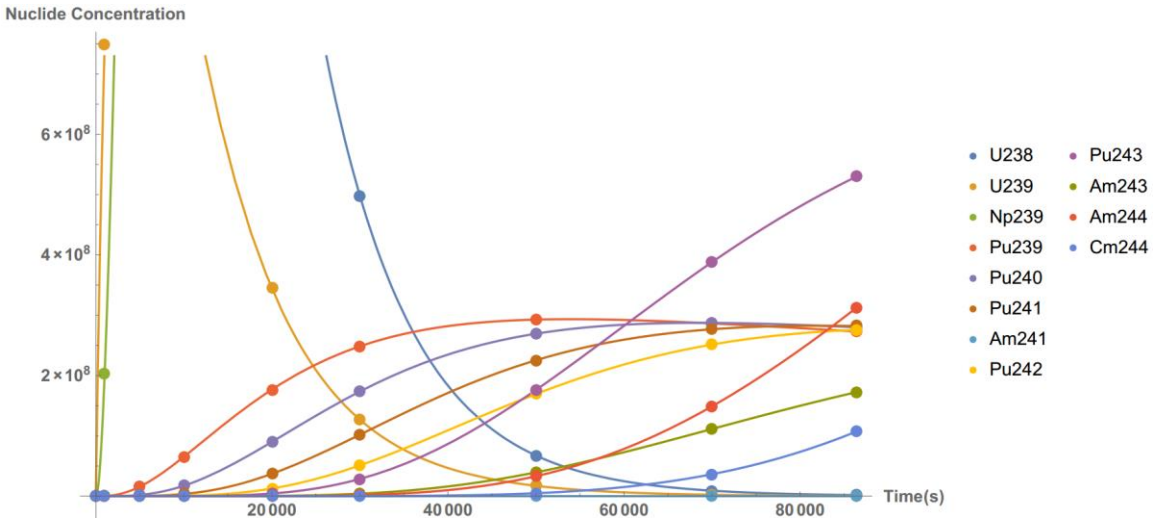


**Figure 19.** APIDA solution points plotted over the analytical solution to benchmark #5.

### 7.2 Isotopic sensitivity analysis with a fuel pin cell

As discussed in Chapter 6, the number of nuclides tracked in a problem is of considerable interest given the effect of problem size on both memory and solution accuracy. To provide some perspective on the implications of increasing the number of nuclides tracked in a problem, the sparsity pattern of burnup matrix produced when tracked 274 isotopes is shown in Figure 20 and the coinciding sparsity pattern from tracking 1049 isotopes is shown in Figure 21.

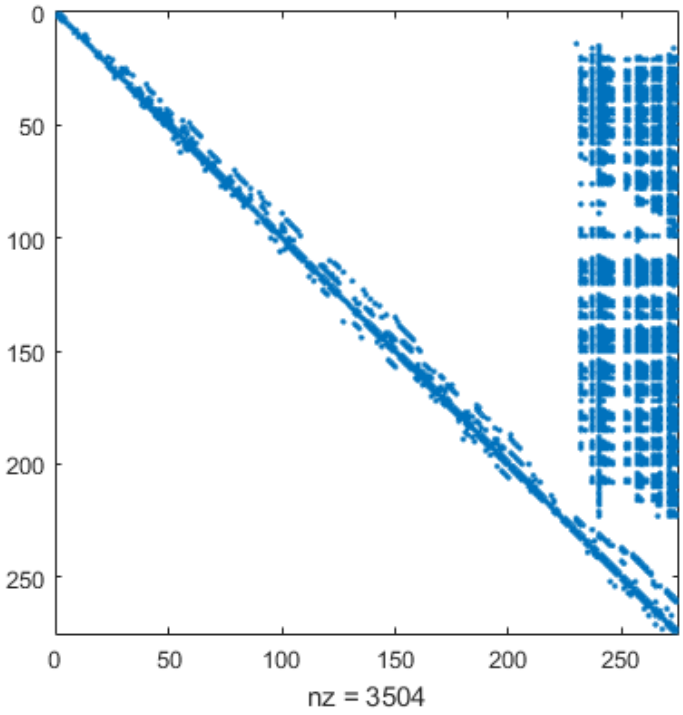


**Figure 20.** Sparsity pattern of the burnup matrix when tracking 274 isotopes.

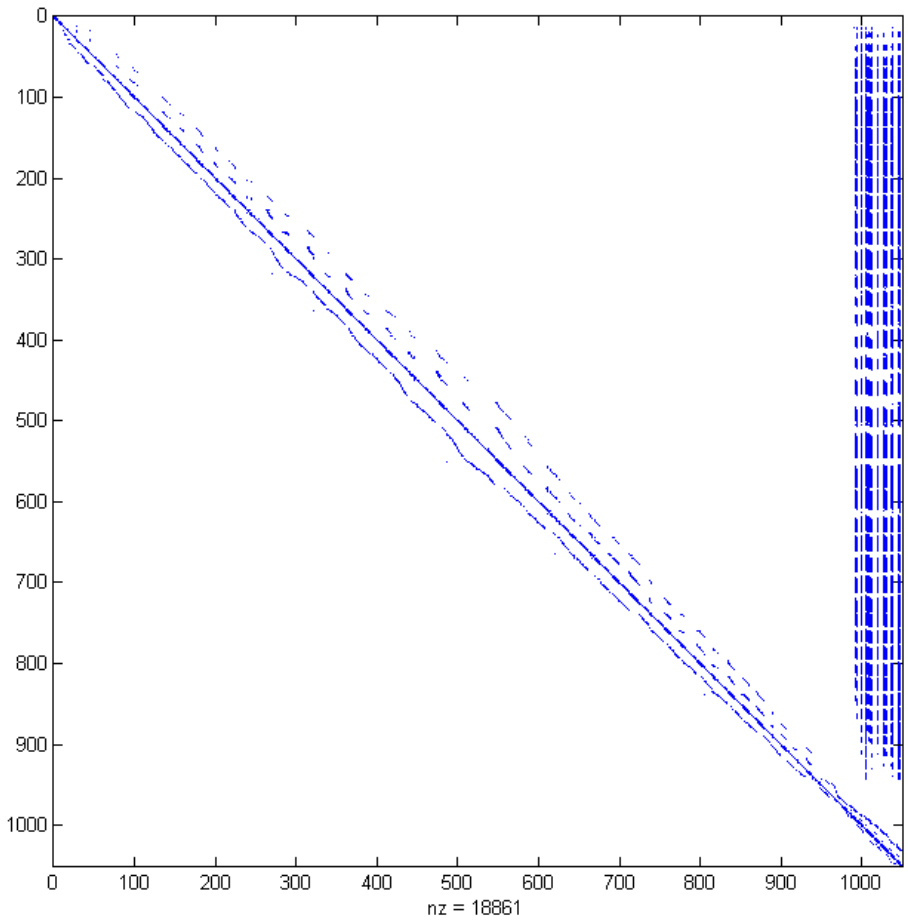


**Figure 21.** Sparsity pattern of the burnup matrix when tracking 1049 isotopes.

The fuel pin cell described in Chapter 6 was modeled in SERPENT and burned for one time step – 250 MWD/MTU. The eigenvalue produced by SERPENT using their burnup solver produced an eigenvalue of 1.20613 with a standard deviation (S.D.) of 27 pcm. The transport solution, namely the reaction rates, produced by the steady state calculation, in SERPENT was then used in APIDA to solve for the change in nuclide concentration. Calculations were run using a different number of isotopes to observe the

effect on the eigenvalue once the new concentrations were applied. The results comparing the SERPENT eigenvalue to the eigenvalues produced using the APIDA material concentrations for a varying number of isotopes are shown in Tables 10 and 11. A listing of the specific isotopes used for each calculation is available in Appendix B. SERPENT transport calculations were run 15000 histories per cycle, 1000 total cycles with 200 cycles skipped.

**Table 10.** Comparison of APIDA eigenvalues to the SERPENT eigenvalue for one burnup step tracking 274 nuclides to 665 nuclides.

| Number of isotopes | k-eff | S.D. | Diff. from SERPENT (pcm) |
|---|---|---|---|
| 274 | 1.234 | 0.00026 | -2781 |
| 298 | 1.23312 | 0.00027 | -2693 |
| 322 | 1.23281 | 0.00026 | -2662 |
| 347 | 1.22936 | 0.00026 | -2317 |
| 372 | 1.22862 | 0.00027 | -2243 |
| 397 | 1.22822 | 0.00026 | -2203 |
| 422 | 1.21075 | 0.00026 | -456 |
| 447 | 1.21004 | 0.00027 | -385 |
| 471 | 1.21033 | 0.00027 | -414 |
| 496 | 1.20777 | 0.00027 | -158 |
| 519 | 1.20715 | 0.00027 | -96 |
| 544 | 1.20789 | 0.00026 | -170 |
| 569 | 1.20794 | 0.00027 | -175 |
| 594 | 1.20774 | 0.00027 | -155 |
| 619 | 1.20738 | 0.00027 | -119 |
| 644 | 1.20605 | 0.00027 | 14 |
| 665 | 1.20655 | 0.00026 | -36 |

**Table 11.** Comparison of APIDA eigenvalues to the SERPENT eigenvalue for one burnup step tracking 687 nuclides to 1049 nuclides.

| Number of isotopes | k-eff | S.D. | Diff. from SERPENT (pcm) |
|---|---|---|---|
| 687 | 1.20604 | 0.00026 | 15 |
| 713 | 1.20608 | 0.00026 | 11 |
| 738 | 1.20614 | 0.00027 | 5 |
| 762 | 1.20615 | 0.00027 | 4 |
| 788 | 1.20612 | 0.00027 | 7 |
| 813 | 1.20597 | 0.00028 | 22 |
| 836 | 1.20552 | 0.00026 | 67 |
| 861 | 1.20571 | 0.00027 | 48 |
| 885 | 1.20568 | 0.00029 | 51 |
| 909 | 1.20668 | 0.00029 | -49 |
| 931 | 1.20591 | 0.00029 | 28 |
| 952 | 1.20576 | 0.00029 | 43 |
| 972 | 1.20567 | 0.00027 | 52 |
| 991 | 1.20584 | 0.00029 | 35 |
| 1013 | 1.20596 | 0.00026 | 23 |
| 1036 | 1.20597 | 0.00026 | 22 |
| 1049 | 1.20599 | 0.00027 | 20 |

As expected, increasing the number of isotopes yields a solution closer the reference calculation provided by SERPENT, which tracks over 1000 isotopes for each calculation. The threshold for the minimum number of isotopes appears to begin around 644 nuclides. The grouping of the isotopes also plays a considerable part, as the isotopes important to fully realizing the transmutation pathways for important nuclides needs to be included. This phenomenon is illustrated for in Figure 22. After approximately 600 isotopes, the oscillation of difference between the APIDA solution and the SERPENT solution is within the standard deviation of the calculation.

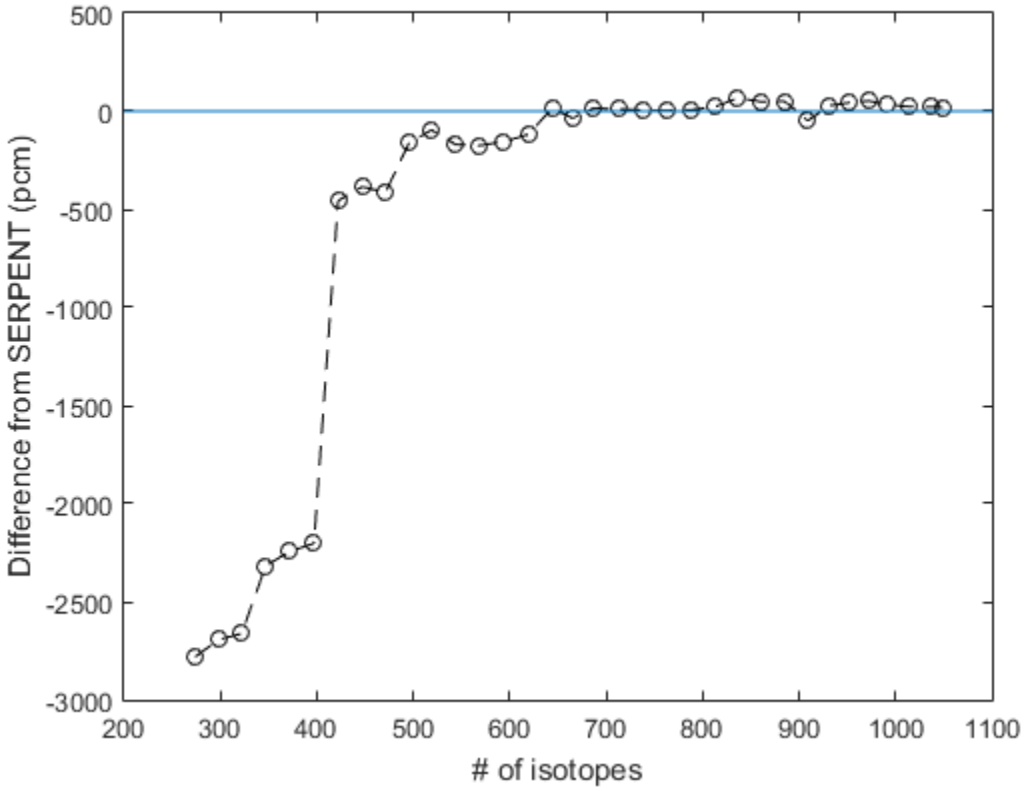


**Figure 22.** Convergence of APIDA generated eigenvalues in SERPENT to the reference solution given different numbers of isotopes.

This process was repeated for the fuel pin with gadolinium described in Chapter 6. The fuel pin cell was modeled in SERPENT and burned for one time-step – 250 MWD/MTU. The eigenvalue produced by SERPENT using their burnup solver produced an eigenvalue of 0.212881 with a standard deviation (S.D.) of 86 pcm. The results comparing the SERPENT eigenvalue to the eigenvalues produced using the APIDA material concentrations for a varying number of isotopes are shown in Tables 12 and 13. A listing of the specific isotopes used for each calculation is available in Appendix B. SERPENT transport calculations were run 15000 histories per cycle, 1000 total cycles with 200 cycles skipped.

**Table 12.** Comparison of APIDA eigenvalues to the SERPENT gadded fuel pin eigenvalue for one burnup step tracking 274 nuclides to 665 nuclides.

| Number of isotopes | k-eff | S.D. | Diff. from SERPENT (pcm) |
|---|---|---|---|
| 274 | 0.212909 | 0.00086 | -2.8 |
| 298 | 0.212744 | 0.00086 | 13.7 |
| 322 | 0.213017 | 0.00086 | -13.6 |
| 347 | 0.213021 | 0.00089 | -14 |
| 372 | 0.212901 | 0.00087 | -2 |
| 397 | 0.213046 | 0.00089 | -16.5 |
| 422 | 0.21247 | 0.00086 | 41.1 |
| 447 | 0.212942 | 0.00090 | -6.1 |
| 471 | 0.21315 | 0.00088 | -26.9 |
| 496 | 0.213171 | 0.00083 | -29 |
| 519 | 0.213054 | 0.00089 | -17.3 |
| 544 | 0.212823 | 0.00090 | 5.8 |
| 569 | 0.212944 | 0.00087 | -6.3 |
| 594 | 0.212567 | 0.00085 | 31.4 |
| 619 | 0.213158 | 0.00087 | -27.7 |
| 644 | 0.21276 | 0.00090 | 12.1 |
| 665 | 0.212723 | 0.00090 | 15.8 |

**Table 13.** Comparison of APIDA eigenvalues to the SERPENT gadded fuel pin eigenvalue for one burnup step tracking 687 nuclides to 1049 nuclides.

| Number of isotopes | k-eff | S.D. | Diff. from SERPENT (pcm) |
|---|---|---|---|
| 687 | 0.212795 | 0.00083 | 8.6 |
| 713 | 0.212818 | 0.00086 | 6.3 |
| 738 | 0.213179 | 0.00087 | -29.8 |
| 762 | 0.212368 | 0.00085 | 51.3 |
| 788 | 0.212982 | 0.00085 | -10.1 |
| 813 | 0.212633 | 0.00085 | 24.8 |
| 836 | 0.21298 | 0.00086 | -9.9 |
| 861 | 0.212771 | 0.00085 | 11 |
| 885 | 0.213166 | 0.00085 | -28.5 |
| 909 | 0.212603 | 0.00087 | 27.8 |
| 931 | 0.212986 | 0.00085 | -10.5 |
| 952 | 0.21308 | 0.00089 | -19.9 |
| 972 | 0.212924 | 0.00086 | -4.3 |
| 991 | 0.213116 | 0.00088 | -23.5 |
| 1013 | 0.212672 | 0.00089 | 20.9 |
| 1036 | 0.212924 | 0.00085 | -4.3 |
| 1049 | 0.213096 | 0.00081 | -21.5 |

Unlike the fuel pin modeled with no gadolinium, the eigenvalue does not vary significantly given the number of isotopes tracked over one time-step in the gadded fuel pin. The low eigenvalue and the slightly higher standard deviation does factor into the analysis, but the depressed amount of fissions and reactions in the isotopes important to criticality likely decreases the dependence on the number of isotopes for one time-step. The behavior of the eigenvalue is plotted in Figure 23.

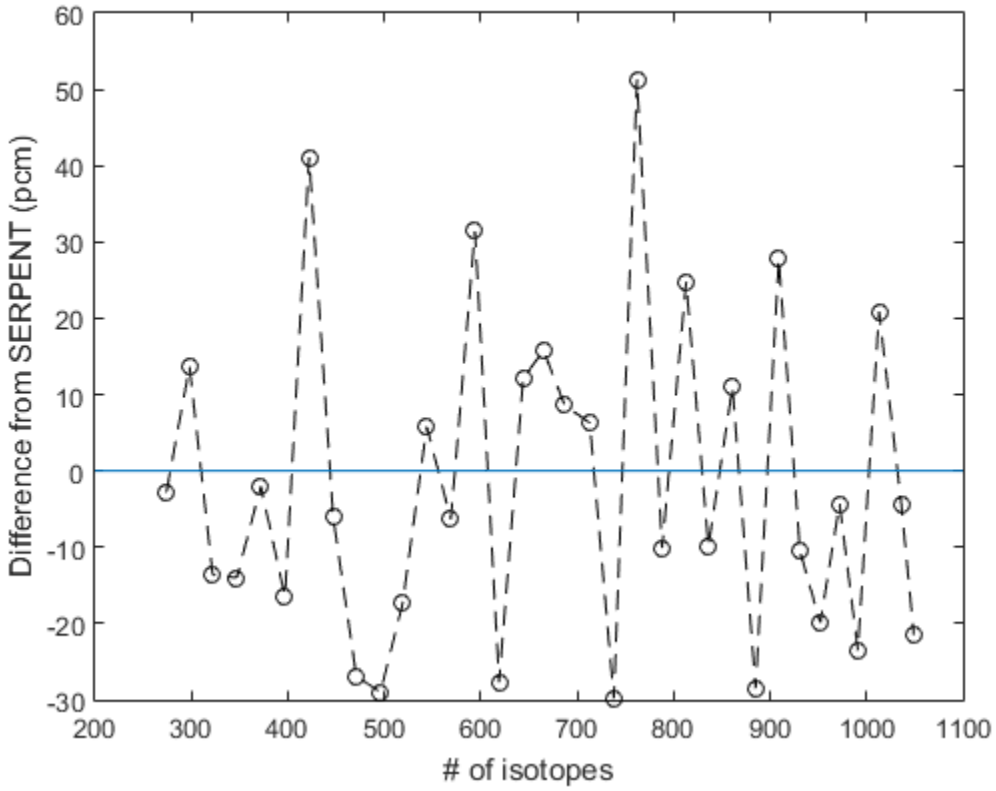


**Figure 23.** Convergence of APIDA generated eigenvalues in SERPENT to the reference solution of the gadded fuel pin given different numbers of isotopes.

Figure 24 does not show the same asymptotic behavior as Figure 23, and the oscillations around 0 are statistical and well within the standard deviations reported by SERPENT. This likely results from the eigenvalue of the gadded fuel pin staying relatively flat over the first few burnup steps. While somewhat illuminating, these results are not entirely conclusive and further work should be performed with more particle histories to provide more insight.

### 7.3 Multi-step pin cell depletion calculation

The $UO_2$ fuel pin cell used in the sensitivity study was also used in the multi-step depletion calculation to compare APIDA to SERPENT and HELIOS. Results were generated with both SERPENT and HELIOS up to a total burnup of 10,000 MWD/MTU at intervals of 500 MWD/MTU, with the first two steps at 250 MWD/MTU for higher fidelity. Two sets of SERPENT results are shown. The transport solutions from SERPENT were used at each steps by APIDA to generate the new material concentrations for the next step. The power was fixed at 34 W/g and calculations were run without xenon equilibrium and with predictor-corrector turned off. The results using SERPENT were generated running 15000 histories per cycle with 1000 total cycles and 200 inactive cycles. Figure 24 shows the change in the eigenvalue (k-effective) over these burnup steps using SERPENT, HELIOS, and APIDA, with APIDA results generated with 738 isotopes tracked and 1049 isotopes tracked.

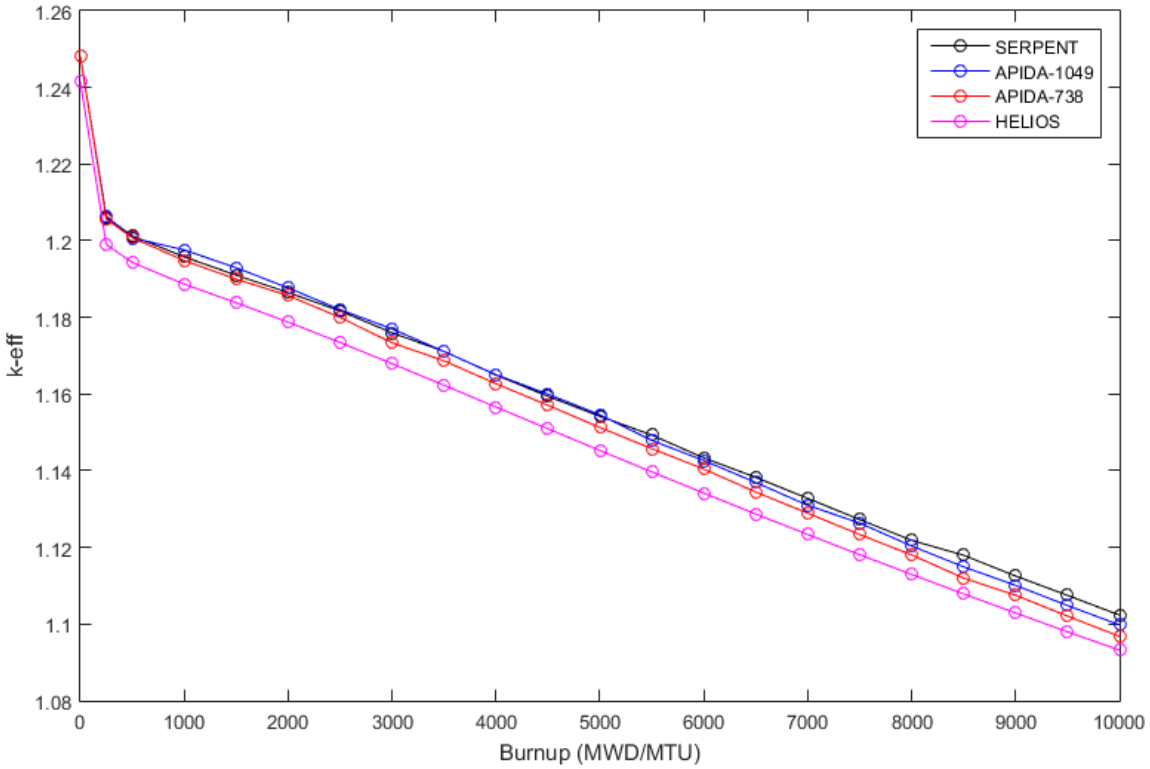


**Figure 24.** Eigenvalue (k-eff) as a function of burnup resulting from SERPENT, HELIOS, and APIDA calculations.

The depletion curve generated by APIDA shows excellent agreement with SERPENT, especially when tracking 1049 isotopes. The curve generated when tracking 738 isotope seems to diverge as the fuel pin is burned more, indicating added importance to some of the less dominant actinides in terms of pathways of transmutation. One noticeable difference between the burnup methods used in SERPENT and APIDA is the imposed limit used by SERPENT for fission product yields. SERPENT cuts off the tracking of fission products if the fission yield fraction is $1x10^{-6}$ or less, a variable which could have significant effects in terms of pathways of transmutation and the production of important fission products. The statistical nature of Monte Carlo calculations, especially

when it's not clear if the source has converged to the correct solution, also contributes to some of the differences in the eigenvalue.

In addition to the core eigenvalue, the atom density of certain isotopes is of considerable interest in fuel cycle analysis. One of the most important fission products related to criticality analysis is $^{135}$Xe as it acts as a strong neutron absorber, depressing the fission density in the fuel pin. Figure 25 shows the production of $^{135}$Xe as calculated by SERPENT and APIDA, as well as the relative difference over several burnup steps.

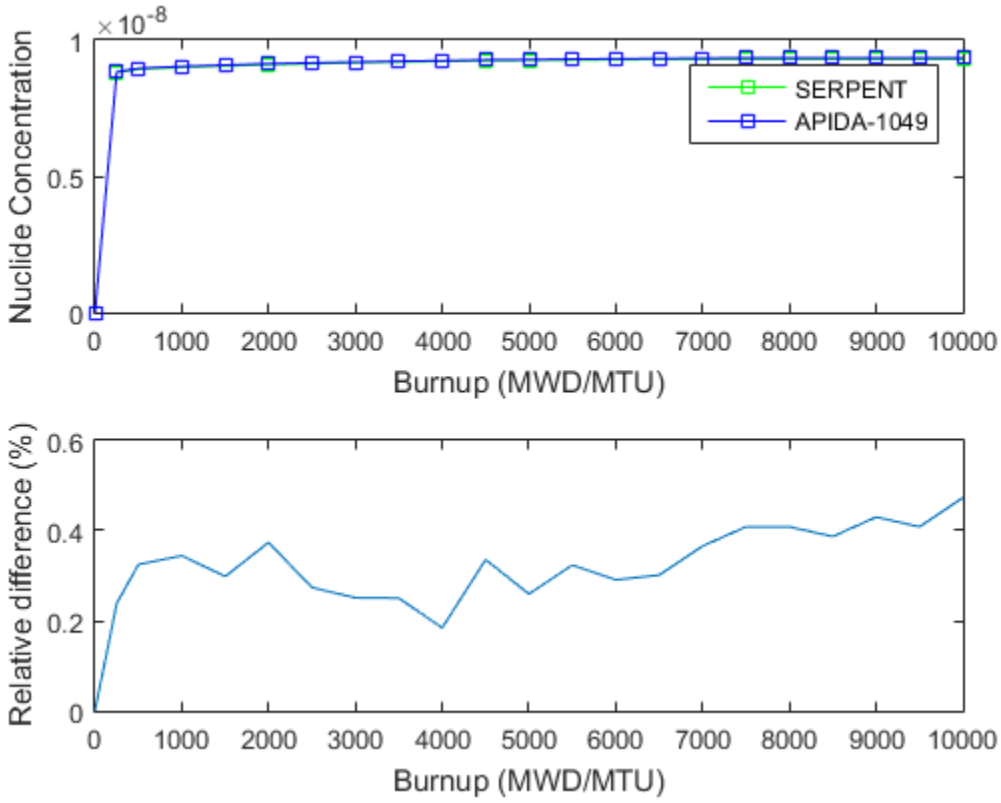


**Figure 25.** Production of $^{135}$Xe as a function of burnup resulting from SERPENT and APIDA calculations.

Another factor in criticality is the enrichment of $^{235}U$ in $UO_2$ fuel. Figure 26 shows the change in $^{235}U$ atomic density in the fuel pin cell over several burnup steps. The relative difference between SERPENT and APIDA oscillates around zero, bounded by approximately 0.05%.

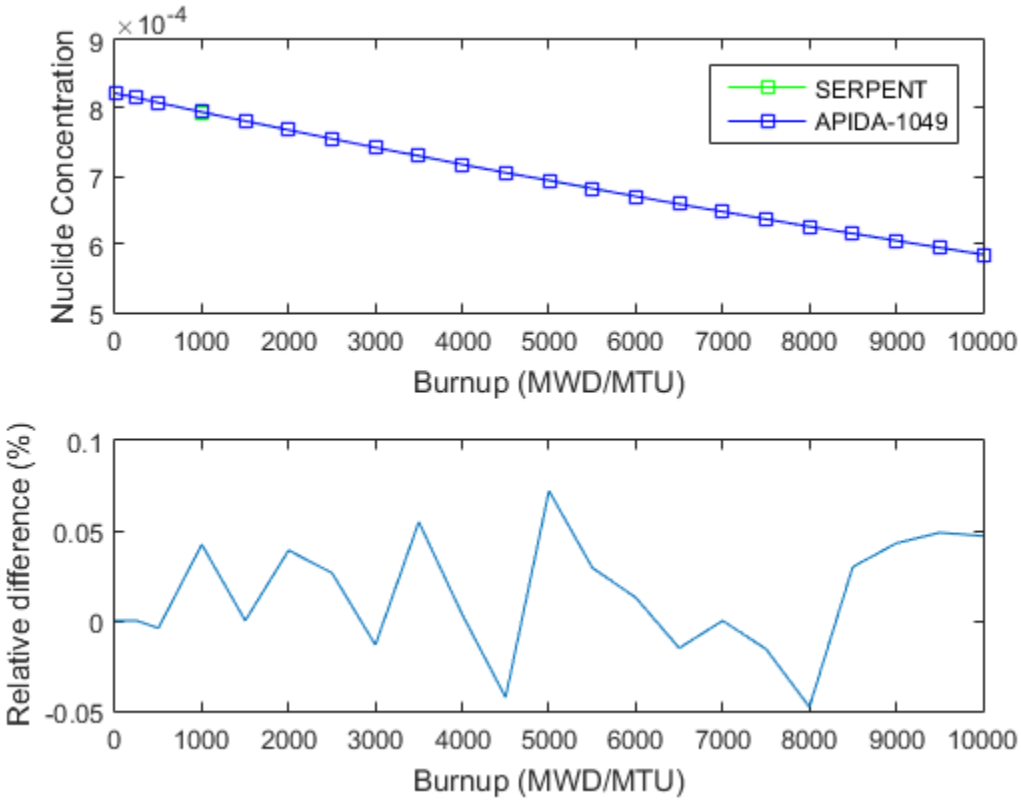


**Figure 26.** Change in $^{235}U$ as a function of burnup resulting from SERPENT and APIDA calculations.

Finally, a parameter with significance to nuclear nonproliferation and national security is the ratio of $^{240}Pu$ to $^{239}Pu$. This ratio is key in determining the viability of plutonium as a special nuclear material for weapons production. Figure 27 shows the ratio of $^{240}Pu$ to $^{239}Pu$ as a function of burnup.

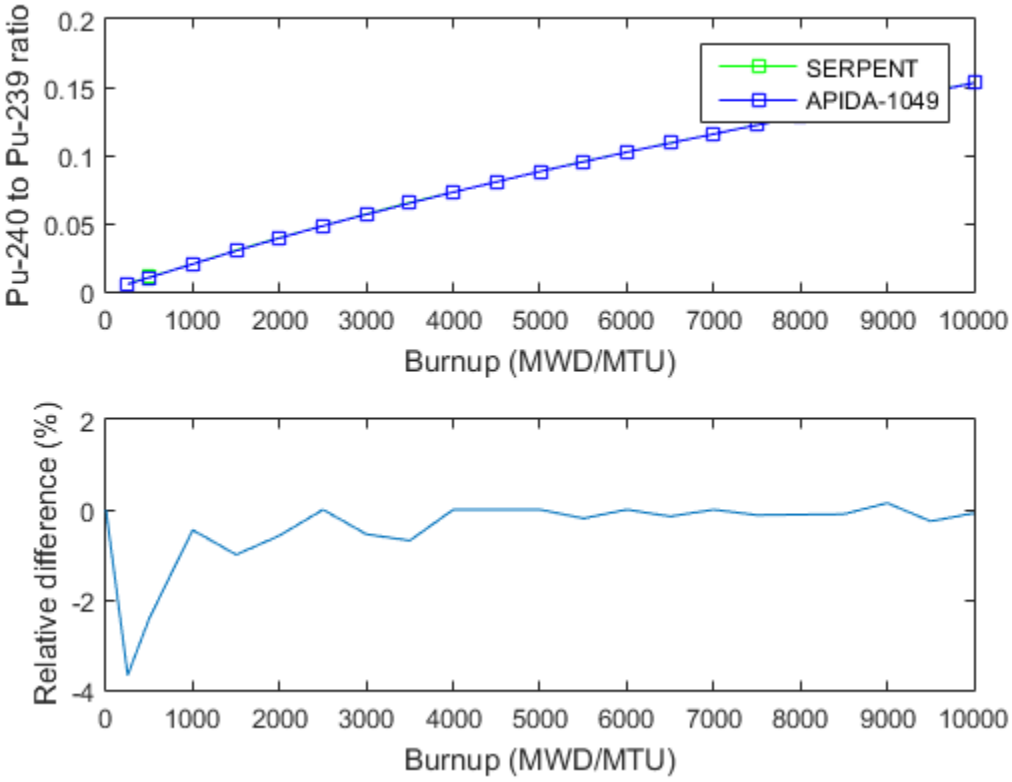


**Figure 27.** Ratio of $^{240}$Pu to $^{239}$Pu as a function of burnup resulting from SERPENT and APIDA calculations.

Initially divergent, the relative difference starts to approach zero as the fuel pin is burned more. This is a phenomenon which should be investigated further, as this ratio of interest is particularly important during the early stages of burnup when the amount of $^{239}$Pu is still dominant and makes the plutonium more amenable for weapons production. The culprit in this analysis may be the extremely low atomic density at the first time step (on the order of $1x10^{-10}$). The relative agreement between APIDA and SERPENT is encouraging, but more work should be done to benchmark APIDA with transport codes to ensure proper communication between modules.

# CHAPTER 8

# CONCLUSIONS

In this dissertation, a robust, powerful, and portable burnup tool was developed with the capability to easily couple with any transport solver in memory to efficiently perform lattice depletion calculations. The new code, APIDA, employs a novel hybrid burnup solver and presents completely new code module capable of seamless integration within independently developed transport solvers.

The APIDA code was validated with Mathematica to benchmark the burnup tool and its ability to calculate material concentrations after multiple time steps. The solutions produced by the APIDA code resulted in a relative percent error of well below 1% for all nuclides, even with the most numerically taxing problem descriptions.

The APIDA framework was built from the ground up as an object-oriented API in the C++ language and the pertinent public classes were described in the manuscript. A simple example implementation was provided to show the ease of use in terms of integration with other codes in memory.

A sensitivity analysis was conducted to investigate the effect on the core eigenvalue of number of nuclides in a burnup calculation. Using a wide range of isotopes, the APIDA code results indicate the effect of tracking more the about 600 isotopes is not significant for the EPR fuel pin cell problem described in this study. For the fuel pin with integrated burnable absorber, the number of isotopes being tracked had little to no impact in terms of eigenvalue, but further work can be done with a different concentration of gadolinium.

The APIDA code was also benchmarked with the SERPENT code to track the change in eigenvalue of a pin cell over multiple time steps. The results generated by APIDA showed excellent agreement with the SERPENT reference solution and maintained high accuracy in terms of tracking the change in nuclide concentration over time. APIDA was shown to provide accurate solutions for some of the more important parameters related to criticality and nuclear nonproliferation, namely the buildup of $^{135}$Xe and the ratio of $^{240}$Pu to $^{239}$Pu.

Overall, the capabilities of APIDA were shown to be both powerful and easy to implement. Future work for APIDA includes the implementation of a more general library reader class to handle raw ENDF files. The next step in elevating COMET as a multi-physics code is to refactor the framework to make it more conducive for interfacing with APIDA and other modules in the future. The methods in APIDA can also be expanded to handle some of the lesser occurring but still present neutron-induced reactions not considered in this study. The APIDA code should also be integrated with a massively parallel transport code to validate the thread-safe nature of the API framework in APIDA.

# APPENDIX A

# CLASSES AND FUNCTIONS IN APIDA

Tables A.1 through A.4 list and describe the public functions available to the user in the APIDA code.

**Table A1**. Functions and descriptions of the 'apida' class.

**Class:** *apida*

| **Function** | **Type** | **Description** |
|---|---|---|
| apida(); | void | Constructor to initialize the 'apida' class. |
| ~apida(); | virtual | Destructor to kill instance of class. |
| initialize_library(vector<int>, vector<double>, vector<double>); | void | Function to initialize library class and read in decay data. |
| set_initial_concentrations(vector<double>); | void | Function to set initial concentration vector. |
| set_new_concentrations(vector<double>); | void | Function to set new concentrations after a time step. |
| set_times(vector<double>); | void | Function to set time steps. |
| set_rxn_rates(vector< vector<double>>); | void | 2-d vector holding reaction rates for each isotope. |
| get_concentrations( vector< vector<double>>); | void | Function to retrieve all concentrations after each time step. |
| get_concentrations_at_step( vector<double>); | void | Function to retrieve concentrations at a specific time step. |
| run(); | void | Function to run burnup calculation. |

**Table A2**. Functions and descriptions of the 'library_builder' class.

**Class:** *library_builder*

| Function | Type | Description |
|---|---|---|
| Library_builder(); | void | Constructor to initialize class. |
| ~Library_builder(); | virtual | Destructor to kill instance of class. |
| get_lib_type(int); | int | Function to get type of library for an isotope. |
| get_hl_units(int); | int | Function to get half-life unites for an isotope. |
| get_half_life(int); | double | Function to get half-life of an isotope. |
| get_beta_1(int); | double | Function to get probability an isotope decays via beta emission. |
| get_beta_2(int); | double | Function to get probability an isotope decays via beta emission to a metastable state. |
| get_posit_1(int); | double | Function to get probability an isotope decays via positron emission. |
| get_posit_2(int); | double | Function to get probability an isotope decays via positron emission to a metastable state. |
| get_alpha(int); | double | Function to get probability an isotope decays via alpha particle emission. |
| get_isomer(int); | double | Function to get probability an isotope decays via isomeric transition. |
| get_spont_fiss(int); | double | Function to get probability an isotope decays via spontaneous fission. |
| get_delayed(int); | double | Function to get probability an isotope decays via delayed neutron emission. |
| get_decay_heat(int); | double | Function to get decay heat. |

Table A2 (continued)

| | | |
|---|---|---|
| get_recover_gx(int); | double | Function to get fraction of recoverable energy per disintegration from gamma and x-rays. |
| get_nat_abund(int); | double | Function to get atom percent abundance of naturally occurring isotopes. |
| get_water_rcg(int); | double | Function to get radioactivity concentration in water. |
| get_air_rcg(int); | double | Function to get radioactivity concentration in air. |
| get_beta_double(int); | double | Function to get probability an isotope decays via double beta emission. |
| get_neutron_decay(int); | double | Function to get probability an isotope decays via neutron emission. |
| get_beta_alpha(int); | double | Function to get probability an isotope decays via beta and alpha emission. |
| get_fiss_array(int); | double | Function to get isotope from fissionable isotope array. |
| get_yield_frac(int,int); | double | Function to get fission yield fraction from one isotope to a specific fission product. |
| interpolate_yields(vector<double>, int,); | double | Function to interpolate and get the fission yield fraction from one isotope to a specific product given multiple yield energies. |

**Table A3**. Functions and descriptions of the 'tran_mat' class.

**Class:** *tran_mat*

| Function | Type | Description |
|---|---|---|
| tran_mat(); | void | Constructor to initialize class. |
| ~tran_mat() | virtual | Destructor to kill instance of class. |
| construct_transition_matrix(Library_builder*, vector<vector<double>>,vector<int>,bool); | void | Function to construct transition matrix for burnup calculation. |
| convert_to_lambda(double,int); | double | Function to convert half-life in seconds to decay probability (1/s). |
| branching_ratio_calculator(Library_builder*, int,int,int); | double | Function to calculate branching ratio of one isotope decaying to another. |
| neutron_rxn_calc(Library_builder*,int,int,int&); | bool | Function to calculate if an isotope is produced via a neutron induced reaction. |
| fission_yield_calculator(Library_builder*, int,int,int,int); | double | Function to calculate the fission yield fraction of an isotope. |
| print_transition_matrix(vector<int>, vector<double>,vector<double>,constchar*); | void | Function to print the transition matrix in MATLAB sparse matrix format. |
| reduce_transition_matrix(vector<int>,bool, vector<double>&,vector<double>&, vector<double>&,bool&); | void | Function to reduce the transition matrix and remove isotopes causing a zero diagonal element. |

**Table A4**. Functions and descriptions of the ‘depletion’ class.

**Class:** *depletion*

| **Function** | **Type** | **Description** |
|---|---|---|
| depletion(); | void | Constructor to initialize class. |
| ~depletion(); | virtual | Destructor to kill instance of class. |
| set_library(Library_builder*); | void | Function to set pointer to library class. |
| set_initial_conc(vector<double>,vector<double>, tran_mat*); | void | Function to set initial concentration. |
| set_reaction_rates(vector<vector<double>>); | void | Function to set reaction rates. |
| set_time_interval(double); | void | Function to set time intervals. |
| set_final_conc_u(vector<double>); | void | Function to set final concentration of isotopes not reduced from the matrix. |
| set_final_conc_s(vector<double>); | void | Function to set final concentration of isotopes reduced from the matrix. |
| get_new_conc(vector<double>&,tran_mat*); | void | Function to get final concentrations. |
| get_time(); | double | Function to get time interval for a substep. |
| get_initial_conc_u(int); | double | Function to get initial concentration of a reduced isotope. |

Table A4 (continued)

| | | |
|---|---|---|
| get_initial_conc_s(int); | double | Function to get initial concentration of a not reduced isotope. |
| run(depletion*,tran_mat*,bool,bool,int,int); | void | Function to run calculation. |

# APPENDIX B

# LIST OF ISOTOPES IN SENSITIVITY ANALYSIS

**Table B1**. Isotopes used in 274 nuclide calculations.

| Isotope # | | Isotope # | | Isotope # | | Isotope # | |
|---|---|---|---|---|---|---|---|
| 1 | 10010 | 36 | 360860 | 71 | 441010 | 106 | 501160 |
| 2 | 10020 | 37 | 370850 | 72 | 441020 | 107 | 501170 |
| 3 | 10030 | 38 | 370860 | 73 | 441030 | 108 | 501180 |
| 4 | 20030 | 39 | 370870 | 74 | 441040 | 109 | 501190 |
| 5 | 40090 | 40 | 380840 | 75 | 441050 | 110 | 501200 |
| 6 | 60120 | 41 | 380860 | 76 | 441060 | 111 | 501220 |
| 7 | 70140 | 42 | 380870 | 77 | 451030 | 112 | 501230 |
| 8 | 80160 | 43 | 380880 | 78 | 451050 | 113 | 501240 |
| 9 | 80170 | 44 | 380890 | 79 | 461020 | 114 | 501250 |
| 10 | 280610 | 45 | 380900 | 80 | 461040 | 115 | 501260 |
| 11 | 280620 | 46 | 390890 | 81 | 461050 | 116 | 511210 |
| 12 | 280640 | 47 | 390900 | 82 | 461060 | 117 | 511230 |
| 13 | 290630 | 48 | 390910 | 83 | 461070 | 118 | 511240 |
| 14 | 290650 | 49 | 400900 | 84 | 461080 | 119 | 511250 |
| 15 | 310690 | 50 | 400910 | 85 | 461100 | 120 | 511260 |
| 16 | 310710 | 51 | 400920 | 86 | 471070 | 121 | 521220 |
| 17 | 320700 | 52 | 400930 | 87 | 471090 | 122 | 521230 |
| 18 | 320720 | 53 | 400940 | 88 | 471101 | 123 | 521240 |
| 19 | 320730 | 54 | 400950 | 89 | 471110 | 124 | 521250 |
| 20 | 320740 | 55 | 400960 | 90 | 481060 | 125 | 521260 |
| 21 | 320760 | 56 | 410930 | 91 | 481080 | 126 | 521271 |
| 22 | 330750 | 57 | 410940 | 92 | 481100 | 127 | 521280 |
| 23 | 340760 | 58 | 410950 | 93 | 481110 | 128 | 521291 |
| 24 | 340770 | 59 | 420920 | 94 | 481120 | 129 | 521300 |
| 25 | 340780 | 60 | 420940 | 95 | 481130 | 130 | 521320 |
| 26 | 340790 | 61 | 420950 | 96 | 481140 | 131 | 531270 |
| 27 | 340800 | 62 | 420960 | 97 | 481150 | 132 | 531290 |
| 28 | 340820 | 63 | 420970 | 98 | 481151 | 133 | 531300 |
| 29 | 350790 | 64 | 420980 | 99 | 481160 | 134 | 531310 |
| 30 | 350810 | 65 | 420990 | 100 | 491130 | 135 | 531350 |
| 31 | 360800 | 66 | 421000 | 101 | 491150 | 136 | 541260 |
| 32 | 360820 | 67 | 430990 | 102 | 501120 | 137 | 541280 |

Table B1 (continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 33 | 360830 | 68 | 440980 | 103 | 501130 | 138 | 541290 |
| 34 | 360840 | 69 | 440990 | 104 | 501140 | 139 | 541300 |
| 35 | 360850 | 70 | 441000 | 105 | 501150 | 140 | 541310 |

| Isotope # | | Isotope # | | Isotope # | |
|---|---|---|---|---|---|
| 141 | 541320 | 176 | 601450 | 211 | 661580 |
| 142 | 541330 | 177 | 601460 | 212 | 661600 |
| 143 | 541340 | 178 | 601470 | 213 | 661610 |
| 144 | 541350 | 179 | 601480 | 214 | 661620 |
| 145 | 541360 | 180 | 601500 | 215 | 661630 |
| 146 | 551330 | 181 | 611470 | 216 | 661640 |
| 147 | 551340 | 182 | 611480 | 217 | 671650 |
| 148 | 551350 | 183 | 611481 | 218 | 671661 |
| 149 | 551360 | 184 | 611490 | 219 | 681640 |
| 150 | 551370 | 185 | 611510 | 220 | 681660 |
| 151 | 561300 | 186 | 621470 | 221 | 681670 |
| 152 | 561320 | 187 | 621480 | 222 | 681680 |
| 153 | 561330 | 188 | 621490 | 223 | 681700 |
| 154 | 561340 | 189 | 621500 | 224 | 882230 |
| 155 | 561350 | 190 | 621510 | 225 | 882240 |
| 156 | 561360 | 191 | 621520 | 226 | 882250 |
| 157 | 561370 | 192 | 621530 | 227 | 882260 |
| 158 | 561380 | 193 | 621540 | 228 | 892260 |
| 159 | 561400 | 194 | 631510 | 229 | 892270 |
| 160 | 571380 | 195 | 631520 | 230 | 902270 |
| 161 | 571390 | 196 | 631530 | 231 | 902280 |
| 162 | 571400 | 197 | 631540 | 232 | 902290 |
| 163 | 581380 | 198 | 631550 | 233 | 902300 |
| 164 | 581390 | 199 | 631560 | 234 | 902320 |
| 165 | 581400 | 200 | 631570 | 235 | 902330 |
| 166 | 581410 | 201 | 641520 | 236 | 902340 |
| 167 | 581420 | 202 | 641530 | 237 | 912310 |
| 168 | 581430 | 203 | 641540 | 238 | 912320 |
| 169 | 581440 | 204 | 641550 | 239 | 912330 |
| 170 | 591410 | 205 | 641560 | 240 | 922320 |
| 171 | 591420 | 206 | 641570 | 241 | 922330 |
| 172 | 591430 | 207 | 641580 | 242 | 922340 |

Table B1 (continued)

| | | | | | |
|---|---|---|---|---|---|
| 173 | 601420 | 208 | 641600 | 243 | 922350 |
| 174 | 601430 | 209 | 651590 | 244 | 922360 |
| 175 | 601440 | 210 | 651600 | 245 | 922370 |

| Isotope # | |
|---|---|
| 246 | 922380 |
| 247 | 922390 |
| 248 | 922400 |
| 249 | 922410 |
| 250 | 932350 |
| 251 | 932360 |
| 252 | 932370 |
| 253 | 932380 |
| 254 | 932390 |
| 255 | 942360 |
| 256 | 942370 |
| 257 | 942380 |
| 258 | 942390 |
| 259 | 942400 |
| 260 | 942410 |
| 261 | 942420 |
| 262 | 942430 |
| 263 | 942440 |
| 264 | 952410 |
| 265 | 952420 |
| 266 | 952421 |
| 267 | 952430 |
| 268 | 952440 |
| 269 | 952441 |
| 270 | 962410 |
| 271 | 962420 |
| 272 | 962430 |
| 273 | 962440 |
| 274 | 962450 |

**Table B2**. Additional isotopes used in 298 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 360851 |
| 2 | 360870 |
| 3 | 360880 |
| 4 | 380910 |
| 5 | 380920 |
| 6 | 390911 |
| 7 | 390920 |
| 8 | 390930 |
| 9 | 400970 |
| 10 | 410970 |
| 11 | 430991 |
| 12 | 511270 |
| 13 | 511280 |
| 14 | 521311 |
| 15 | 521340 |
| 16 | 531320 |
| 17 | 531330 |
| 18 | 531340 |
| 19 | 541331 |
| 20 | 551380 |
| 21 | 561390 |
| 22 | 571410 |
| 23 | 571420 |
| 24 | 591450 |

**Table B3**. Additional isotopes used in 322 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 350830 |
| 2 | 360831 |
| 3 | 370880 |
| 4 | 370890 |
| 5 | 390940 |
| 6 | 390950 |
| 7 | 421010 |
| 8 | 421020 |
| 9 | 431010 |
| 10 | 461090 |
| 11 | 511290 |
| 12 | 511310 |
| 13 | 521270 |
| 14 | 521310 |
| 15 | 521330 |
| 16 | 521331 |
| 17 | 541311 |
| 18 | 541380 |
| 19 | 551390 |
| 20 | 561410 |
| 21 | 561420 |
| 22 | 571430 |
| 23 | 591460 |
| 24 | 601490 |
| 25 | |

**Table B4**. Additional isotopes used in 347 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 330770 |
| 2 | 340830 |
| 3 | 350840 |
| 4 | 360890 |
| 5 | 370900 |
| 6 | 380930 |
| 7 | 380940 |
| 8 | 410951 |
| 9 | 431040 |
| 10 | 431050 |
| 11 | 451031 |
| 12 | 461120 |
| 13 | 501210 |
| 14 | 501211 |
| 15 | 501270 |
| 16 | 501280 |
| 17 | 511300 |
| 18 | 511330 |
| 19 | 521290 |
| 20 | 541351 |
| 21 | 541370 |
| 22 | 581450 |
| 23 | 581460 |
| 24 | 591470 |
| 25 | 621560 |

**Table B5**. Additional isotopes used in 372 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 320770 |
| 2 | 340810 |
| 3 | 360900 |
| 4 | 370901 |
| 5 | 370910 |
| 6 | 400980 |
| 7 | 421030 |
| 8 | 431030 |
| 9 | 451070 |
| 10 | 471120 |
| 11 | 471130 |
| 12 | 481170 |
| 13 | 491151 |
| 14 | 511301 |
| 15 | 511320 |
| 16 | 511321 |
| 17 | 531360 |
| 18 | 541390 |
| 19 | 551400 |
| 20 | 571440 |
| 21 | 591480 |
| 22 | 591490 |
| 23 | 601510 |
| 24 | 601520 |
| 25 | 932400 |

**Table B6**. Additional isotopes used in 397 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 320780 |
| 2 | 330780 |
| 3 | 340840 |
| 4 | 350850 |
| 5 | 350860 |
| 6 | 350870 |
| 7 | 380950 |
| 8 | 410981 |
| 9 | 410990 |
| 10 | 421040 |
| 11 | 481131 |
| 12 | 481180 |
| 13 | 491171 |
| 14 | 501291 |
| 15 | 501300 |
| 16 | 531341 |
| 17 | 531370 |
| 18 | 551410 |
| 19 | 561430 |
| 20 | 571450 |
| 21 | 581470 |
| 22 | 581480 |
| 23 | 591440 |
| 24 | 611500 |
| 25 | 641590 |

**Table B7**. Additional isotopes used in 422 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 350880 |
| 2 | 360910 |
| 3 | 401000 |
| 4 | 410960 |
| 5 | 410991 |
| 6 | 411010 |
| 7 | 421050 |
| 8 | 441070 |
| 9 | 441080 |
| 10 | 461110 |
| 11 | 471150 |
| 12 | 491170 |
| 13 | 501231 |
| 14 | 501290 |
| 15 | 501301 |
| 16 | 521350 |
| 17 | 531321 |
| 18 | 531361 |
| 19 | 541400 |
| 20 | 551381 |
| 21 | 561440 |
| 22 | 611520 |
| 23 | 611530 |
| 24 | 621550 |
| 25 | 651610 |

**Table B8**. Additional isotopes used in 447 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 330790 |
| 2 | 340791 |
| 3 | 340811 |
| 4 | 340850 |
| 5 | 340860 |
| 6 | 370920 |
| 7 | 370930 |
| 8 | 390960 |
| 9 | 390961 |
| 10 | 390970 |
| 11 | 400990 |
| 12 | 410980 |
| 13 | 431020 |
| 14 | 431060 |
| 15 | 451051 |
| 16 | 491191 |
| 17 | 501251 |
| 18 | 501310 |
| 19 | 501311 |
| 20 | 501320 |
| 21 | 521360 |
| 22 | 531380 |
| 23 | 571460 |
| 24 | 611540 |
| 25 | 631580 |

**Table B9**. Additional isotopes used in 471 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 300720 |
| 2 | 320750 |
| 3 | 330810 |
| 4 | 330820 |
| 5 | 330830 |
| 6 | 350820 |
| 7 | 350841 |
| 8 | 350890 |
| 9 | 370940 |
| 10 | 401010 |
| 11 | 401020 |
| 12 | 411000 |
| 13 | 431070 |
| 14 | 451090 |
| 15 | 481171 |
| 16 | 501171 |
| 17 | 511281 |
| 18 | 521251 |
| 19 | 551420 |
| 20 | 561450 |
| 21 | 571461 |
| 22 | 591481 |
| 23 | 591510 |
| 24 | 601530 |

**Table B10**. Additional isotopes used in 496 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 320800 |
| 2 | 340870 |
| 3 | 360920 |
| 4 | 380960 |
| 5 | 390990 |
| 6 | 411020 |
| 7 | 411021 |
| 8 | 411030 |
| 9 | 411040 |
| 10 | 421060 |
| 11 | 431021 |
| 12 | 451060 |
| 13 | 461130 |
| 14 | 461140 |
| 15 | 471091 |
| 16 | 471160 |
| 17 | 481190 |
| 18 | 501271 |
| 19 | 511341 |
| 20 | 541410 |
| 21 | 551430 |
| 22 | 571470 |
| 23 | 581490 |
| 24 | 591500 |
| 25 | 621570 |

**Table B11.** Additional isotopes used in 519 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 310720 |
| 2 | 310730 |
| 3 | 330800 |
| 4 | 340831 |
| 5 | 390980 |
| 6 | 390981 |
| 7 | 441090 |
| 8 | 451080 |
| 9 | 471111 |
| 10 | 471131 |
| 11 | 471170 |
| 12 | 481200 |
| 13 | 491190 |
| 14 | 491211 |
| 15 | 511220 |
| 16 | 521370 |
| 17 | 531390 |
| 18 | 561460 |
| 19 | 581500 |
| 20 | 601540 |
| 21 | 611550 |
| 22 | 621580 |
| 23 | 631590 |
| 24 | 661660 |

**Table B12**. Additional isotopes used in 544 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280660 |
| 2 | 290670 |
| 3 | 300690 |
| 4 | 300740 |
| 5 | 310740 |
| 6 | 310750 |
| 7 | 310760 |
| 8 | 310770 |
| 9 | 310780 |
| 10 | 310790 |
| 11 | 310800 |
| 12 | 320790 |
| 13 | 320791 |
| 14 | 320810 |
| 15 | 320820 |
| 16 | 320830 |
| 17 | 320840 |
| 18 | 320860 |
| 19 | 330760 |
| 20 | 330821 |
| 21 | 330840 |
| 22 | 330850 |
| 23 | 330860 |
| 24 | 330870 |
| 25 | 340880 |

**Table B13**. Additional isotopes used in 569 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 340890 |
| 2 | 350900 |
| 3 | 350910 |
| 4 | 360930 |
| 5 | 360940 |
| 6 | 370950 |
| 7 | 370960 |
| 8 | 380970 |
| 9 | 380980 |
| 10 | 380990 |
| 11 | 390931 |
| 12 | 391000 |
| 13 | 391010 |
| 14 | 391020 |
| 15 | 401030 |
| 16 | 401040 |
| 17 | 401050 |
| 18 | 410971 |
| 19 | 411001 |
| 20 | 411041 |
| 21 | 411050 |
| 22 | 411060 |
| 23 | 421070 |
| 24 | 421080 |
| 25 | 430980 |

**Table B14**. Additional isotopes used in 594 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 431000 |
| 2 | 431080 |
| 3 | 431090 |
| 4 | 431100 |
| 5 | 441100 |
| 6 | 441110 |
| 7 | 441120 |
| 8 | 451040 |
| 9 | 451081 |
| 10 | 451100 |
| 11 | 451110 |
| 12 | 451120 |
| 13 | 451130 |
| 14 | 451140 |
| 15 | 461111 |
| 16 | 461150 |
| 17 | 461160 |
| 18 | 461170 |
| 19 | 461180 |
| 20 | 471140 |
| 21 | 471151 |
| 22 | 471180 |
| 23 | 471190 |
| 24 | 481191 |
| 25 | 481210 |

**Table B15**. Additional isotopes used in 619 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 481211 |
| 2 | 481220 |
| 3 | 481230 |
| 4 | 481240 |
| 5 | 491180 |
| 6 | 491200 |
| 7 | 491210 |
| 8 | 491220 |
| 9 | 491230 |
| 10 | 491231 |
| 11 | 491240 |
| 12 | 491241 |
| 13 | 491250 |
| 14 | 491251 |
| 15 | 491260 |
| 16 | 491270 |
| 17 | 491271 |
| 18 | 491280 |
| 19 | 491290 |
| 20 | 491291 |
| 21 | 491300 |
| 22 | 501191 |
| 23 | 501281 |
| 24 | 501330 |
| 25 | 501340 |

**Table B16**. Additional isotopes used in 644 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 511261 |
| 2 | 511340 |
| 3 | 511350 |
| 4 | 511360 |
| 5 | 521380 |
| 6 | 531280 |
| 7 | 531331 |
| 8 | 531400 |
| 9 | 531410 |
| 10 | 541420 |
| 11 | 541430 |
| 12 | 551341 |
| 13 | 551351 |
| 14 | 551361 |
| 15 | 551440 |
| 16 | 551450 |
| 17 | 561351 |
| 18 | 561371 |
| 19 | 561470 |
| 20 | 561480 |
| 21 | 571480 |
| 22 | 571490 |
| 23 | 581510 |
| 24 | 581520 |
| 25 | 591441 |

**Table B17**. Additional isotopes used in 665 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 591520 |
| 2 | 591530 |
| 3 | 591540 |
| 4 | 601550 |
| 5 | 601560 |
| 6 | 611460 |
| 7 | 611521 |
| 8 | 611541 |
| 9 | 611560 |
| 10 | 611570 |
| 11 | 621590 |
| 12 | 631600 |
| 13 | 641610 |
| 14 | 641620 |
| 15 | 651620 |
| 16 | 651630 |
| 17 | 661650 |
| 18 | 671660 |
| 19 | 671670 |
| 20 | 681690 |
| 21 | 902310 |

**Table B18**. Additional isotopes used in 687 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 300750 |
| 2 | 300760 |
| 3 | 310810 |
| 4 | 330880 |
| 5 | 350920 |
| 6 | 381000 |
| 7 | 410931 |
| 8 | 421090 |
| 9 | 441130 |
| 10 | 451061 |
| 11 | 451150 |
| 12 | 471161 |
| 13 | 471171 |
| 14 | 471181 |
| 15 | 471200 |
| 16 | 481300 |
| 17 | 491201 |
| 18 | 491221 |
| 19 | 491261 |
| 20 | 491281 |
| 21 | 511370 |
| 22 | 541341 |
| 23 | 551320 |
| 24 | 571500 |

**Table B19**. Additional isotopes used in 713 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 300770 |
| 2 | 300780 |
| 3 | 310820 |
| 4 | 320771 |
| 5 | 370970 |
| 6 | 390901 |
| 7 | 451101 |
| 8 | 451160 |
| 9 | 461190 |
| 10 | 471210 |
| 11 | 471221 |
| 12 | 481250 |
| 13 | 481260 |
| 14 | 491310 |
| 15 | 521390 |
| 16 | 521400 |
| 17 | 531301 |
| 18 | 541440 |
| 19 | 551460 |
| 20 | 581530 |
| 21 | 611580 |
| 22 | 621600 |
| 23 | 631610 |
| 24 | 651640 |
| 25 | 681660 |
| 26 | 912340 |

**Table B20**. Additional isotopes used in 738 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 300770 |
| 2 | 300780 |
| 3 | 310820 |
| 4 | 320771 |
| 5 | 370970 |
| 6 | 390901 |
| 7 | 451101 |
| 8 | 451160 |
| 9 | 461190 |
| 10 | 471210 |
| 11 | 471221 |
| 12 | 481250 |
| 13 | 481260 |
| 14 | 491310 |
| 15 | 521390 |
| 16 | 521400 |
| 17 | 531301 |
| 18 | 541440 |
| 19 | 551460 |
| 20 | 581530 |
| 21 | 611580 |
| 22 | 621600 |
| 23 | 631610 |
| 24 | 651640 |
| 25 | 681660 |
| 26 | 912340 |

**Table B21**. Additional isotopes used in 763 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 290690 |
| 2 | 310840 |
| 3 | 350821 |
| 4 | 360950 |
| 5 | 370980 |
| 6 | 381010 |
| 7 | 390891 |
| 8 | 431120 |
| 9 | 451020 |
| 10 | 451170 |
| 11 | 471201 |
| 12 | 471220 |
| 13 | 471230 |
| 14 | 481231 |
| 15 | 481280 |
| 16 | 481310 |
| 17 | 501350 |
| 18 | 531260 |
| 19 | 531420 |
| 20 | 551470 |
| 21 | 561490 |
| 22 | 631620 |
| 23 | 641640 |
| 24 | 661651 |
| 25 | 661680 |

**Table B22**. Additional isotopes used in 788 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 290710 |
| 2 | 290720 |
| 3 | 290730 |
| 4 | 320870 |
| 5 | 340771 |
| 6 | 340910 |
| 7 | 350930 |
| 8 | 391040 |
| 9 | 411080 |
| 10 | 421110 |
| 11 | 431130 |
| 12 | 441160 |
| 13 | 451180 |
| 14 | 461210 |
| 15 | 471100 |
| 16 | 471240 |
| 17 | 511241 |
| 18 | 571370 |
| 19 | 581540 |
| 20 | 591560 |
| 21 | 601580 |
| 22 | 611590 |
| 23 | 621610 |
| 24 | 631541 |
| 25 | 671680 |

**Table B23**. Additional isotopes used in 813 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280680 |
| 2 | 290680 |
| 3 | 290700 |
| 4 | 290701 |
| 5 | 290740 |
| 6 | 290750 |
| 7 | 300800 |
| 8 | 310741 |
| 9 | 310830 |
| 10 | 401060 |
| 11 | 411090 |
| 12 | 451190 |
| 13 | 461091 |
| 14 | 461220 |
| 15 | 491330 |
| 16 | 531430 |
| 17 | 541450 |
| 18 | 571520 |
| 19 | 611600 |
| 20 | 631630 |
| 21 | 641650 |
| 22 | 651580 |
| 23 | 651660 |
| 24 | 671690 |
| 25 | 932410 |

**Table B24**. Additional isotopes used in 836 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280670 |
| 2 | 280690 |
| 3 | 280700 |
| 4 | 280710 |
| 5 | 290660 |
| 6 | 320751 |
| 7 | 350940 |
| 8 | 360980 |
| 9 | 370990 |
| 10 | 381020 |
| 11 | 421120 |
| 12 | 431140 |
| 13 | 441170 |
| 14 | 471250 |
| 15 | 491181 |
| 16 | 511380 |
| 17 | 521410 |
| 18 | 561361 |
| 19 | 561500 |
| 20 | 621460 |
| 21 | 621620 |
| 22 | 641660 |
| 23 | 651670 |

**Table B25**. Additional isotopes used in 861 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280720 |
| 2 | 290760 |
| 3 | 290770 |
| 4 | 330890 |
| 5 | 340920 |
| 6 | 350800 |
| 7 | 391050 |
| 8 | 401070 |
| 9 | 431150 |
| 10 | 451200 |
| 11 | 461230 |
| 12 | 481290 |
| 13 | 501360 |
| 14 | 541460 |
| 15 | 551480 |
| 16 | 561510 |
| 17 | 571530 |
| 18 | 581550 |
| 19 | 591570 |
| 20 | 601590 |
| 21 | 611610 |
| 22 | 631521 |
| 23 | 631640 |
| 24 | 651680 |
| 25 | 661690 |

**Table B26**. Additional isotopes used in 885 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280730 |
| 2 | 280740 |
| 3 | 300810 |
| 4 | 320731 |
| 5 | 360970 |
| 6 | 411100 |
| 7 | 411110 |
| 8 | 421130 |
| 9 | 441180 |
| 10 | 451210 |
| 11 | 461240 |
| 12 | 471260 |
| 13 | 491161 |
| 14 | 491340 |
| 15 | 511221 |
| 16 | 511390 |
| 17 | 531440 |
| 18 | 591400 |
| 19 | 621630 |
| 20 | 631650 |
| 21 | 641670 |
| 22 | 681671 |
| 23 | 681720 |
| 24 | 691710 |

**Table B27**. Additional isotopes used in 909 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 270670 |
| 2 | 270680 |
| 3 | 270690 |
| 4 | 270700 |
| 5 | 280750 |
| 6 | 290780 |
| 7 | 310700 |
| 8 | 350950 |
| 9 | 360810 |
| 10 | 370861 |
| 11 | 381030 |
| 12 | 401080 |
| 13 | 431160 |
| 14 | 441190 |
| 15 | 451041 |
| 16 | 471080 |
| 17 | 471270 |
| 18 | 481320 |
| 19 | 511200 |
| 20 | 591421 |
| 21 | 631500 |
| 22 | 681710 |
| 23 | 691720 |
| 24 | 912350 |

**Table B28**. Additional isotopes used in 931 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 260660 |
| 2 | 260670 |
| 3 | 270660 |
| 4 | 270710 |
| 5 | 320710 |
| 6 | 330900 |
| 7 | 410941 |
| 8 | 451220 |
| 9 | 461071 |
| 10 | 471280 |
| 11 | 481090 |
| 12 | 491140 |
| 13 | 521210 |
| 14 | 541270 |
| 15 | 581391 |
| 16 | 601410 |
| 17 | 671640 |
| 18 | 671700 |
| 19 | 671710 |
| 20 | 681650 |
| 21 | 912300 |
| 22 | 952400 |

**Table B29**. Additional isotopes used in 952 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 290681 |
| 2 | 410920 |
| 3 | 420930 |
| 4 | 430970 |
| 5 | 461030 |
| 6 | 491120 |
| 7 | 521231 |
| 8 | 641510 |
| 9 | 651690 |
| 10 | 661590 |
| 11 | 661700 |
| 12 | 661710 |
| 13 | 671701 |
| 14 | 671720 |
| 15 | 822080 |
| 16 | 822100 |
| 17 | 822120 |
| 18 | 892280 |
| 19 | 922300 |
| 20 | 942450 |
| 21 | 952450 |

**Table B30**. Additional isotopes used in 972 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 260680 |
| 2 | 300691 |
| 3 | 350801 |
| 4 | 370840 |
| 5 | 380871 |
| 6 | 451010 |
| 7 | 471060 |
| 8 | 511201 |
| 9 | 551310 |
| 10 | 561310 |
| 11 | 581370 |
| 12 | 651570 |
| 13 | 671630 |
| 14 | 671641 |
| 15 | 812080 |
| 16 | 832120 |
| 17 | 862200 |
| 18 | 882220 |
| 19 | 882280 |
| 20 | 902260 |
| 21 | 922310 |

**Table B31**. Additional isotopes used in 991 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 431170 |
| 2 | 481111 |
| 3 | 491131 |
| 4 | 531250 |
| 5 | 661570 |
| 6 | 681630 |
| 7 | 812070 |
| 8 | 822110 |
| 9 | 832100 |
| 10 | 832110 |
| 11 | 842100 |
| 12 | 842160 |
| 13 | 862180 |
| 14 | 862190 |
| 15 | 862220 |
| 16 | 872230 |
| 17 | 882270 |
| 18 | 892250 |
| 19 | 932340 |

**Table B32.** Additional isotopes used in 1013 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 280630 |
| 2 | 280650 |
| 3 | 300820 |
| 4 | 380850 |
| 5 | 430960 |
| 6 | 471071 |
| 7 | 481070 |
| 8 | 491110 |
| 9 | 551300 |
| 10 | 671620 |
| 11 | 812090 |
| 12 | 822090 |
| 13 | 822140 |
| 14 | 832130 |
| 15 | 832140 |
| 16 | 842110 |
| 17 | 842140 |
| 18 | 842150 |
| 19 | 842180 |
| 20 | 872210 |
| 21 | 872220 |
| 22 | 942371 |

**Table B33**. Additional isotopes used in 1036 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 260590 |
| 2 | 260690 |
| 3 | 270610 |
| 4 | 270620 |
| 5 | 270720 |
| 6 | 270730 |
| 7 | 280760 |
| 8 | 290790 |
| 9 | 310850 |
| 10 | 320880 |
| 11 | 340930 |
| 12 | 370830 |
| 13 | 421140 |
| 14 | 441200 |
| 15 | 471050 |
| 16 | 671631 |
| 17 | 802060 |
| 18 | 812060 |
| 19 | 812100 |
| 20 | 842120 |
| 21 | 842130 |
| 22 | 852170 |
| 23 | 852180 |

**Table B34**. Additional isotopes used in 1049 nuclide calculations.

| Isotope # | |
|---|---|
| 1 | 260600 |
| 2 | 270600 |
| 3 | 270601 |
| 4 | 290640 |
| 5 | 390880 |
| 6 | 400880 |
| 7 | 400890 |
| 8 | 410910 |
| 9 | 551280 |
| 10 | 551290 |
| 11 | 832090 |
| 12 | 862170 |
| 13 | 962400 |
| 14 | 962460 |

# APPENDIX C

# PARAMETERS FOR BENCHMARK PROBLEMS

**Table C1**. Parameters for Benchmark #1.

**Time step (seconds) =** 5.00E+17

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **U-238** | 1.0000E+10 | 8.56077093E+08 |
| **Th-234*** | 0.0000E+00 | 1.26423127E-02 |

*Non-physical, but still mathematically correct.

**Table C2**. Parameters for Benchmark #2.

**Time step (seconds) =**1.00E+12

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **Np-237** | 1.00E+12 | 9.89807657E+11 |
| **Pa-233** | 0.00E+00 | 3.40955164E+04 |
| **U-234** | 0.00E+00 | 9.51933360E+09 |

**Table C3**. Parameters for Benchmark #3.

**Time step (seconds) =** 1.00E+04

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **Pb-211** | 1.00E+10 | 4.07570892E+08 |
| **Bi-211** | 1.00E+04 | 2.56832069E+07 |
| **Tl-207** | 1.00E+01 | 6.57809821E+07 |
| **Pb-207** | 0.00E+00 | 9.50078286E+09 |

**Table C4**. Parameters for Benchmark #4.

**Time step (seconds) =** 8.64E+04

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **U-235** | 1.00E+12 | 1.76886902E+08 |
| **U-236** | 1.00E+02 | 1.52830284E+09 |
| **U-237** | 1.00E+02 | 9.22519446E+11 |
| **Np-237** | 1.00E+02 | 7.57753646E+10 |

**Table C5**. Parameters for Benchmark #5.

**Time step (seconds) =** 8.64E+04

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **U-238** | 1.00E+10 | 1.76896303E+06 |
| **U-239** | 1.00E+03 | 1.52835126E+07 |
| **Np-239** | 0.00E+00 | 4.95776250E+09 |
| **Pu-239** | 0.00E+00 | 7.45536885E+05 |
| **Pu-240** | 0.00E+00 | 1.54790667E+04 |
| **Pu-241** | 0.00E+00 | 3.48985976E+08 |
| **Pu-242** | 0.00E+00 | 5.94369007E+08 |
| **Pu-243** | 0.00E+00 | 1.68096243E+09 |
| **Am241** | 0.00E+00 | 6.98479021E+08 |
| **Am-243** | 0.00E+00 | 2.63960541E+00 |
| **Am-244** | 0.00E+00 | 6.27192533E+08 |
| **Cm-244** | 0.00E+00 | 9.42659550E+08 |

**Table C6**. Parameters for Benchmark #6.

**Time step (seconds) =** 8.64E+04

| | **Initial Concentration** | **Final Concentration** |
|---|---|---|
| **U-238** | 1.00E+12 | 1.67017008E+07 |
| **U-239** | 0.00E+00 | 4.36483823E+06 |
| **Np-239** | 0.00E+00 | 6.72015319E+11 |
| **Pu-239** | 0.00E+00 | 1.53111500E+11 |
| **Pu-240** | 0.00E+00 | 8.39445242E+10 |
| **I-135** | 0.00E+00 | 8.54352141E+07 |
| **Xe-135** | 0.00E+00 | 2.99196873E+08 |
| **Cs-135** | 0.00E+00 | 7.96828259E+08 |

# REFERENCES


[1] AREVA - U.S. EPR Final Safety Analysis Report - AREVA Design Control Document Rev. 5 - Tier 2 Chapter 04 - Reactor - Section 4.3 Nuclear Design. 2013.

[2] AREVA - U.S. EPR Final Safety Analysis Report - AREVA Design Control Document Rev. 5 - Tier 2 Chapter 01 - Introduction and General Description of the Plant - Section 1.2 General Plant Description. 2013.

[3] Ball, S.J., Adams, R.K., MATEXP: A General Purpose Digital Computer Program for Solving Ordinary Differential Equations by the Matrix Exponential Method, ORNL/TM-1933, Union Carbide Corporation (Nuclear Division), Oak Ridge National Laboratory, Oak Ridge, Tenn., August 1967.

[4] Bell, M.J. "ORIGEN: the ORNL isotope generation and depletion code." (1973).

[5] Bateman, H. Proc. Cambridge Phil. Soc., no. 15, p. 423, 1910.

[6] Cetnar, J., "General solution of Bateman equations for nuclear transmutations," Annals of Nuclear Energy, vol. 33, no. 7, pp. 640–645, 2006.

[7] Chadwick, M. B. and others, "ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology," Nuclear Data Sheets, vol.107, pp. 2931–3060, 2006.

[8] Connolly, K., Rahnema, F. "Heterogeneous Coarse Mesh Radiation Transport Method for Neutronic Analysis of Prismatic Reactors," Ann. Nucl. Energy, 56, 87-101 (2013).

[9] Duderstadt, J.J., Hamilton, L.J., *Nuclear reactor analysis.* Wiley, New York. 1976.

[10] Gallopoulos, E., Saad, Y., "Efficient Solution of Parabolic Equations by Krylov Approximation Methods," SIAM J. Sci. Stat. Comput., 13, 5, 1236 (1992).

[11] Gauld, I., others, "Isotopic Depletion and Decay Methods and Analysis Capabilities in Scale." Nuclear Technology, Vol . 174, 169-195, (2009).

[12] Gauld, I., Wiarda, D., "ORIGEN Data Libraries." Scale manual. Oak Ridge National Laboratory. 2009.

[13] Nave, C. Hyperphysics – Nuclear Binding Energy. Copyright 1998. Accessed 2015. (http://hyperphysics.phy-astr.gsu.edu/hbase/nucene/nucbin.html)

[14] Kotlyar, D., Shwageraus, E., (2013), On the use of predictor–corrector method for coupled Monte Carlo burnup codes. Annals Of Nuclear Energy. August 1, 2013;58:228-237.

[15] Lepannen, J., "PSG2/Serpent—A Continuous-Energy Monte Carlo Reactor Physics Burnup Calculation Code," VTT Technical Research Centre of Finland. 2008.

[16] Manalo, K. (2008), Development, Optimization, and Testing of a 3-D Zone Based Burnup/Depletion Solver for Deterministic Transport. Nuclear and Radiological Engineering. Gainesville, FL, University of Florida. MS.

[17] Manalo, K. (2013), Detailed Analysis of Phase Space Effects in Fuel Burnup/Depletion for PWR Assembly & Full Core Models Using Large-Scale Parallel Computation. Nuclear and Radiological Engineering. Atlanta, GA, Georgia Institute of Technology. PhD.

[18] Moler, C., Van Loan, C., "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later," SIAM Rev., 45 (2003).

[19] Mosher, S, Rahnema, F., "The Incident Flux Response Expansion Method for Heterogeneous Coarse Mesh Transport Problems," Transport Theory and Statistical Physics, 35, No. 1, 55-86 (2006).

[20] Pusa, M., Leppanen, J. (2010), "Computing the Matrix Exponential in Burnup Calculations." Nuclear Science And Engineering, 164(2), 140-150.

[21] Pusa, M., "Numerical methods for nuclear fuel burnup calculations." Ph.D. Thesis. Aalto University. Finland. May 2013.

[22] Pusa, M., "Rational Approximations to the Matrix Exponential in Burnup Calculations," Nuclear Science and Engineering, vol. 169, no. 2, pp. 155–167, 2011.

[23] SCALE, a modular code system for performing standardized computer analyses for licensing evaluation / prepared for Spent Fuel Project Office, Office of Nuclear Material Safety and Safeguards, U.S. Nuclear Regulatory Commission. (2000). Washington, DC.

[24] Shultis, J., Faw, R., *Fundamentals of Nuclear Science and Engineering*. CRC Press 2002.

[25] Simeonov, T., November 2003. "Release Notes – Helios System Version 1.8," Studsvik Scandpower Report, SSP-03/221.

[26] Skutnik, S., Havlůj, F., Lago, D., Gauld, I., (2013), Development of an Object-Oriented ORIGEN for Advanced Nuclear Fuel Modeling Applications. M&C 2013:

International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering, Sun Valley, ID (USA), 5-9 May 2013.

[27] Rothblum, U. “Resolvent Expansions of Matrices and Applications.” Linear Algebra and its Applications. 38:33-49, (1981).

[28] Weideman, J., Trefethen, L. N., “Parabolic and Hyperbolic Contours for Computing the Bromwich Integral,” Math. Comp., 76, 259, 1341. 2007.

[29] X-5 Monte Carlo Team, “MCNP—A General Monte Carlo N-Particle Transport Code, Version 5,” Los Alamos National Laboratory. 2005.

[30] Zhang, D., Rahnema, F., “An Efficient Hybrid Stochastic/Deterministic Coarse Mesh Neutron Transport Method,” Ann. Nucl. Energy, 41, 1–11 (2012).

[31] Zhang, D., Rahnema, F., “A Whole-Core Coarse Mesh Neutron Transport Method in 2-D Cylindrical (R, Theta) Geometry,” Nucl. Eng. Des., 265, 997-1004 (2013).

[32] Zhang, D., Rahnema, F., “High Order Perturbation Theory for Incident Flux Response Expansion Methods,” Nucl. Sci. Eng., 176, No. 1, 69-80 (2014).

[33] Zhang, D., Rahnema, F., “A Fission Collision Separation Method for Efficient Incident Flux Response Expansion Coefficient Generation,” Ann. Nucl. Energy, 73, 264–269 (2014).

# VITA

**Daniel Edgardo Lago**

Daniel Lago was born in Caguas, Puerto Rico in 1989 and moved to the continental United States shortly thereafter. Daniel graduated from Cypress Creek High School in Orlando, FL in May of 2007. The University of Florida in Gainesville, FL was Daniel's next destination, the institution where he earned his B.S. in Nuclear and Radiological Engineering in May of 2011. Daniel then elected to attend graduate school at the Georgia Institute of Technology, earning his M.S. in Nuclear and Radiological Engineering in December of 2013.

Daniel currently lives in Atlanta with his lovely wife Lily and his loyal dog Dusty. The latter bears a striking resemblance to Daniel, especially when the doctoral candidate eschews shaving for several months. When he is not working, Daniel enjoys running, brewing his own beer, and following his favorite sports team, the Jacksonville Jaguars.