ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis

A Complete Package for Analysis of Induced Activation

Volume I
Technical Manual

Paul P.H. Wilson
Douglass L. Henderson

January 1998

UWFDM-1070

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis

A Complete Package for Analysis of Induced Activation

Volume I
Technical Manual

Paul P.H. Wilson
Douglass L. Henderson

Fusion Technology Institute
University of Wisconsin-Madison
1500 Engineering Dr.
Madison, WI 53706

January 1998

UWFDM-1070

Abstract

While many codes have been written to compute the induced activation and changes in composition caused by neutron irradiation, most of those which are still being updated are only slowly adding functionality and not improving the accuracy, speed and usability of their existing methods. ALARA moves forward in all four of these areas, with primary importance being placed on the accuracy and speed of solution.

By carefully analyzing the various ways to model the physical system, the methods to solve the mathematical problem and the interaction between these two issues, ALARA chooses an optimum combination to achieve high accuracy, fast computation, and enhanced versatility and ease of use.

The physical system is modeled using advanced linear chains, which include the contributions from straightened loops in the reaction scheme, while the truncation philosophy minimizes the discrepancies between the model and the real problem. The mathematical method is then adaptively chosen based on the characteristics of each linear chain to use analytically exact methods when possible and an accurate expansion technique otherwise.

Future modifications to ALARA include new functionality by implementing methods to use new data libraries, implementing methods to get new information from existing libraries, enhancing usability, and improving speed by fine tuning and parallel processing.

Contents

1. Introduction

When designing any system with a large neutron flux, an important characteristic is the amount of induced activation expected in the system's components during operation, at the end of life and at various times after the shutdown of the system. Many codes have been written to perform such calculations for a variety of systems, from accelerators to fission and fusion reactors. The special conditions of fusion reactors, such as high neutron flux/fluence and pulsed operation, have led to many variations of these codes.

The calculation of induced radioactivity in the first wall, blanket and shield materials is an important task for the design and safety of fusion reactors. The neutron products of the D-T reaction induce radioactivity by interacting with and transporting through these materials with much higher initial energies and populations than those of fission reactors of similar power. The results of these radioactivity calculations are used extensively in safety and design analyses to determine such parameters as the nature of the radioactive waste, the amount of shielding required for radiologically sensitive components, and the decay heating after shutdown. Like other engineering calculations, the accuracy of the results is important; overly conservative approximations result in costly and complicated designs while liberal approximations result in safety and technical hazards to the operators, scientists, public and equipment.

To solve this problem a code must perform two steps. First it must model the physical system in time, space and isotopic composition, creating a system of linear first order ordinary differential equations [ODE's]. Second, the solution to this system of ODE's must be found using a numerical technique. Both steps are non-trivial since the physical problem, while finite in time and space, is theoretically infinite in final isotopic composition, and the resulting ODE's have characteristics which can make their efficient and accurate solution difficult.

ALARA is a new computational tool for performing such calculations. Given a group-wise neutron flux, ALARA uses data from a variety of libraries to determine the altered material composition which is then used to calculate the activity, and β-, γ-, and α-heating. In addition, a group-wise γ-ray source flux can be computed by ALARA to be used for the calculation of doses. Finally, if provided with an adjoint importance field based on flux-to-dose conversion factors and the gamma source distributions, ALARA can directly calculate the biological dose.

1.1. Historical Attempts and Failings

The computational solutions to this problem have been well studied. Many different approaches for modeling the physical problem have been combined with at least as many mathematical solution methodologies. Each combination has its advantages and disadvantages,[1] but none have arrived at an optimum mixture of accuracy, efficiency and usability. Even ignoring the issue of usability, there are few codes which are keeping up with the demands of greater accuracy in modeling and solutions without becoming inconveniently slow.

One of the best performers in the past, in terms of speed and accuracy, has been the DKR[2–4] family of codes.[a] Unfortunately, even though it reaches the ultimate in mathematical accuracy and efficient operation, its physical modeling has left it subject to much philosophical criticism. Namely, DKR is unable to model loops in the reaction scheme. If an isotope undergoes a series of transmutations and decays which lead back to itself, DKR ignores any such contribution in all but the simplest of cases. While it has been shown that this is a somewhat valid criticism,[5] those codes which have addressed this problem in the past have many other failings. Another criticism of DKR is its inability to track and log the production of light ions, often important when analyzing the mechanical integrity of a material. On the other hand, DKR has pioneered the ability to exactly model pulsed irradiation histories and use mathematically exact solution methods while solving a multi-dimensional input problem.

Two of the most popular alternatives to DKR are FISPACT[6] and RACC.[7,8] While FISPACT is heavily used in Europe, it has a number of disadvantages. First and foremost, it is unable to accurately and exactly model the pulses which are today part of the designs of so many fusion reactor systems. This has been shown to be an important issue in the calculation of activity for some isotopes, leading to errors of up to several orders of magnitude.[9] Furthermore, it uses an ODE solver which is step-wise in time and, given the stiffness of the system, requires a slow and tedious calculation. Finally, as a 0-dimensional code, it is only able to find a solution for one given spectral distribution with each operation. While RACC has historically had the same problems with pulse modeling and mathematical method, the newest version, RACC-P, has addressed these issues and now models the pulsing

[a]The DKR family of codes has evolved much since the original authoring of DKR to the most recent version known as DKR-Pulsar. Throughout this report, DKR will refer generally to the entire family and specifically to the most recent version.

exactly and uses a matrix solution method to increase the speed. It does, however, have its own drawbacks. In certain regimes, the solution method for each matrix is subject to significant errors which can then be amplified as this matrix is used to repeatedly calculate the final answer. The data handling methods of RACC are its biggest obstacle to efficient and accurate operation. First, it employs a philosophy to truncate the reaction schemes which leads to inconsistent precision in the solution. It also recreates the reaction schemes for each point in space with a different flux spectrum, a very time consuming process which must be accelerated by solving the problem with a flux which is averaged over a number of spatial points.

While ALARA is an entirely new code product, the methods and philosophies embodied in DKR were chosen as a starting point for its development. The basic philosophies of exact modeling of pulsing, consistent truncation of reaction schemes, and mathematically exact solution methods were retained and the main criticisms addressed. The design philosophy is outlined later in this chapter. A short summary of the features is given in Chapter 2. The specific methodologies and techniques used to improve the physical modeling and mathematical solution are described in Chapters 3 and 4, respectively. A full description of how to use ALARA, including command-line options, input and output file formats, and library formats can be found in the Users' Guide (Volume II).[10]

1.2. Design Philosophy

ALARA has been designed with three basic principles in mind: accuracy, speed, and simplicity. These three qualities have been maximized in ALARA after extensive research of the models involved in such calculations.[1,5] The errors, time of execution, and learning curve have all been made "as low as reasonably achievable".[b] The methods used to model the physical system and to perform the mathematical solution are carefully combined to preserve or enhance the accuracy while accelerating the solution. Throughout all this, there is an underlying effort to ensure that ALARA will be easy to use by providing a simple, well-documented input file format, checking this input for errors, and providing a broad, flexible range of options.

[b] This phrase is the origin of the term ALARA, a well known philosophy in the nuclear industry related to the minimization of radiation exposure when working in radioactive environments.

1.2.1. Accuracy

The accuracy of the final solution is affected both by how realistically the physical system is modeled and by what mathematical methods are employed for the final solution. Unfortunately, these two requirements often conflict; as the physical model becomes more realistic the required mathematical methods become more approximate or error prone. When modeling the physical problem, two of the most important issues are how to deal with loops in the reaction scheme and how to truncate the theoretically infinite isotopic composition to a finite problem. While the effect of the latter on the mathematical method is negligible, the former has a great impact. In the past, the unwritten rule has been that realistic treatment of loops requires complicated/inefficient mathematical methods. ALARA has broken that rule by finding a physical approximation to the loops which retains problem accuracy and allows for quite simple and efficient mathematical methods. The keys to ALARA's mathematical accuracy are the ability to adaptively choose the mathematical technique and the accuracy of those techniques. Two of the three mathematical techniques which ALARA employs are mathematically exact!

1.2.2. Speed

The most significant factor affecting the speed is the chosen class of mathematical method. In particular, unless a linear transformation matrix method is used the time required to exactly model a pulsed history will be large. ALARA employs such matrix methods, solving for the linear transformation from the initial isotopic composition to the final composition for each pulse and inter-pulse dwell period, and then multiplying these matrices to obtain a complete linear transformation for the entire history. In addition to this decision, speed was considered throughout the code design process. For example, data library formats and internal data handling have been implemented with modern techniques to enhance versatility without sacrificing speed.

1.2.3. Simplicity

While accuracy and speed have long been issues in the creation of engineering codes, their simplicity is of increasing importance. In this context, simplicity is an issue for both modification/maintenance and use of the code. Since ALARA has been written in C++, it benefits from some of the philosophies of object-oriented code design. This allows the code

itself to be more readable to future programmers and also facilitates enhanced modularity. This modularity means that if new functionality is added to the code, it can be optimized internally with minimal detrimental effect on the existing code.

ALARA has also been designed with the user in mind. Even though improved methods have existed for years, many codes have continued to use input formats which are reminiscent of punch card input entry. Furthermore, most tools in this field have been designed for the solution at a single spatial point, requiring many subsequent and slightly altered runs to get any kind of spatial information. ALARA allows the user to find the solution to an activation problem in a variety of different multi-dimensional geometries, using a flexible system to define the material properties and allowing a complicated pulsed/intermittent irradiation history and a variety of after-shutdown solution times. Furthermore, the input file can be fully commented, preventing the common difficulty of creating a long list of seemingly disconnected numbers for code input.

Finally, the data used by ALARA can come from one of a variety of sources. To accommodate this, a companion code, ALARA Data Conversion [ALARA_DC], has been written and is described in Appendix C.

2. ALARA Features

ALARA has a number of features which distinguish it from other activation codes.

Geometry Definition

- 3-dimensional activation calculations
- point, slab, cylindrical, spherical and toroidal geometries

Mixture Definition

- predefined material and element libraries
- arbitrary combinations of predefined materials, user-defined materials, elements and isotopes

Interval (Fine Mesh) Definition

- interval volumes defined by user OR calculated by code
- grouping of intervals within zones defined by user OR calculated by code

Truncation

- user defined calculation precision

Flux History

- multiple sequential pulsing schedules
- within each schedule
  - independent pulse width
  - independent scalar flux
  - multiple nested pulsing levels

Internal Features

- adaptive solution method based on individual chain characteristics
- calculation of "looped" chains to user defined precision

Output Options

- spatial solutions can be grouped by mixture definition, zone (coarse mesh point), and/or interval (fine mesh point)
- results given at the end of each pulsing schedule and each after-shutdown time
- various types of results are available including:
  - number density
  - volumetric activity
  - total decay heat

  - partial decay heat from alpha, beta and/or gamma
  - gamma source at each interval with user-defined group structure
  - dose calculation if gamma importance field is provided for folding with gamma source
- full output of decay chains including:
  - relative production used in chain creation/truncation decision
  - reaction type information
  - reason for truncation

3. Physical Model

After describing the nature of the physical problem, this chapter will describe the philosophies, approximations and methods used by ALARA to model this physical problem.

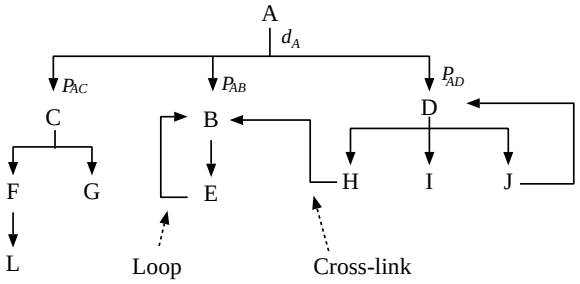

Figure 3.1. Sample reaction tree showing loops and cross-links.

When an isotope is subjected to neutron irradiation, it is likely that a neutron will interact with a nucleus of that isotope, converting it to a different isotope. Many such reactions are possible with each isotope, so that after only one round of neutron reactions, a material made of only one isotope can be partially converted into over 20 others. These isotopes, in turn, can undergo similar interactions, leading to yet more isotopes, and so on. Many of these isotopes can and will be radioactive, and through their decay, even more isotopes can enter the physical system. If this is represented graphically (Figure 3.1), it forms a tree of isotopes which can go on, in principle, infinitely. For the purpose of discussing this physical problem, the products of each generation of reactions (transmutation or decay) will be referred to as a *rank* and each individual reaction from one isotope to another will be referred to as a *branch*.

Each isotope has a unique destruction rate, d_i, while each branch of this tree has an associated production rate, P_{ij}, for the isotope to which this branch leads. For decay reactions, this production rate is independent of the neutron flux while for transmutation reactions it is a function of the spectral distribution of the flux. The raw data used to form these production rates is read from large data libraries, either as decay rate/half-life data from decay libraries or as transmutation cross-sections from transmutation libraries. The methods used to measure, evaluate and compile such data will not be discussed here.

It is possible, as mentioned in Chapter 1, for one nucleus to undergo a series of reactions, being converted from one isotope to another and so on and eventually back to the original isotope. Loops such as this are of specific importance when modeling this physical problem. The nature of such loops is somewhat random; they can begin at any rank in the tree and can undergo any number of reactions before closing the loop. If the *order* of a loop is defined as the number of isotopes between two occurrences of the same isotope in

a loop, then the order can range from 1 to greater than 10. A related but less important phenomenon is that of "cross-linking" of subtrees. This is caused when two different isotopes, which could each be at any rank, both undergo reactions to the same isotope. In this case, the tree can become quite tangled, departing from the classical tree structure known and studied in computer science.

While the problem has finite dependencies on time and space (see below), it is the modeling of this potentially infinite aspect of the physical problem which causes the most difficulty and is described in Section 3.1.

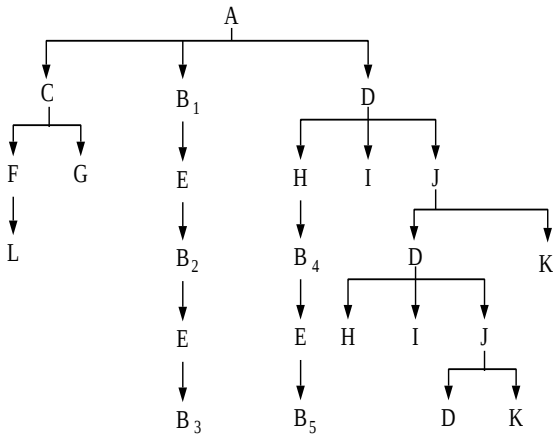

Figure 3.2. Fully straightened and unlinked reaction tree.

Current designs for experimental and power fusion reactors of all types often include the necessity for pulses, from the short frequent pulses of an inertial confinement system to the long infrequent pulses of a magnetic confinement system. This pulsing creates an important effect[9,11,12] since between each pulse, the radioactive isotopes which have been created are able to decay while the stable isotopes remain unchanged. This changes the distribution of isotopes having important implications on the reactions during the subsequent pulses. In practice, these reactors can be pulsed at different frequencies, depending on the experimental, power and/or maintenance requirements. Section 3.2 will describe the approximations and assumptions used in modeling this aspect of the physical model.

Finally, the material composition and neutron flux spectrum will vary from location to location in the device. A structural region of a problem may contain some variety of steel while a coolant region might have water and a breeding region would contain lithium. The initial isotopes, and thus the reaction trees, will therefore be very different for each region. Further, for each point of interest, the spectral distribution and magnitude of the fluxes will be different. Thus, even for identical trees from the same material composition, the production rates for each isotope will vary from point to point. Section 3.3 describes the important aspects of modeling these spatial variations.

3.1. Chain Creation

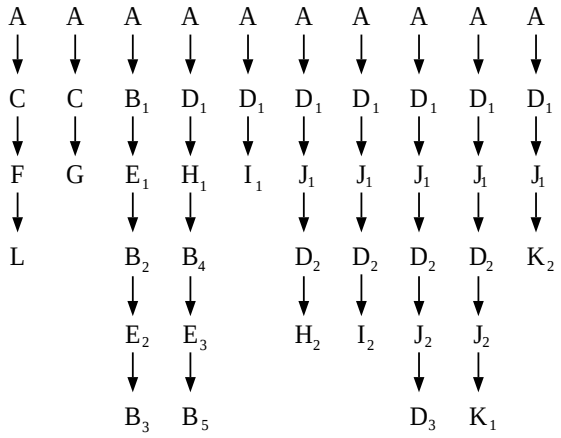

Figure 3.3. Separated linear chains representation of reaction tree.

In principle, it is possible to convert the physical model in its entirety to a mathematical model and solve the problem directly. It is far more practical, however, to convert the single large problem into a number of smaller sub-problems and then solve each one individually, combining the results where appropriate. The first step in this process is to develop a philosophy to convert the cross-linked and loop-filled tree into a true classical tree structure (see Figure 3.2). To perform this requires the introduction of what will be called *partial-contribution* isotopes, or *pc*-isotopes. A *pc*-isotope is an isotope in a tree or chain which is not necessarily unique. The basic physical model described above only allows for one occurrence of each isotope in each tree. By creating *pc*-isotopes, a somewhat larger system is created, but its solution is more simply achieved. After the full solution of the problem, the *pc*-isotopes are collapsed into unique isotopes, with the contribution from each being accounted for appropriately. While the use of *pc*-isotopes to remove cross-links is straightforward, their implementation for removing loops is more complicated and the subject of Section 3.1.1.

After all the loops and cross-links have been removed, each tree is traversed in a *depth-first search*[a], creating a number of independent linear chains (see Figure 3.3). In practice, it is necessary at this stage to truncate these chains to some finite size. As mentioned above, these trees, and therefore the resulting chains, are theoretically infinite and some philosophy must be developed to decide exactly how these chains will be truncated. This is the subject of Section 3.1.2.

[a] A depth-first search is an algorithm which moves deeper into a tree as far as it can go before backtracking and moving down a different path.

3.1.1. Loop Handling

In principle, the method used to remove the loops from the reaction trees is the same as that used to remove the cross-links. Loops, in fact, are just a special case of cross-linked chains, where the chain is linked to itself. Thus, loops are removed by a technique which will be referred to as *loop straightening*. In this method, all the isotopes which make up a loop in the tree are repeated using *pc*-isotopes. This repetition occurs an infinite number of times in theory, but in practice, the chain is truncated using the same methods which are outlined in Section 3.1.2 below.

The primary reason for choosing such a method is to preserve the characteristics of the mathematical model which will be created. If the reaction scheme is treated directly without the removal of cross-links and loops, the mathematical model will inevitably be a somewhat large and full matrix representing a stiff system of ODE's and solving for a large number of isotopes at the same time (Equation 4.1). If, however, it is possible to convert the physical model to one of linear chains, the mathematical model can take the form of a lower bidiagonal matrix (Equation 4.3). This type of mathematical model can be subjected to many special treatments for the accurate and efficient solution of the matrix exponential, some of which are described in Chapter 4.

It is now necessary, however, to show that the use of these methods does not jeopardize the physical accuracy of the model. First, the qualitative effect that the loop-straightening process has on the model must be understood. Each iteration of the loop isotopes adds a new set of *pc*-isotopes to the chain and will be referred to as a *correction*. Just as in many such approximation processes, in the limit as the number of corrections becomes very large, the result approaches the exact solution. It is therefore necessary to choose a point to truncate the approximation while retaining confidence in the solution. Of course, the entire problem, loops or not, is theoretically infinite, so this truncation issue is not unique to loops.

Work has been done to show that the loop-straightening method is also valid quantitatively.[5] A study of the numerical effect of the loop corrections on the result showed that for many realistic first order loops, a single correction could reduce the relative error to the order of 10^{-6} and as few as 3 corrections can reduce this error to less than 10^{-10}. First order loops were used in this analysis because the contribution from each correction in such cases is logically greater than in higher order loops. It is important to recognize the

importance of this measure being a *relative* error. It has no effect on the precision of the results (discussed further in Section 3.1.2), but only on the accuracy of the results.

Relative to other loop handling models, loop straightening can improve the speed and accuracy of the mathematical calculation while maintaining the same precision in the physical model as is being used throughout.

3.1.2. Truncation Philosophy

On the surface, the concept of truncating the theoretically infinite chains created by modeling the physical system for these calculations is a simple one: truncate the chain once the isotopes have a negligible effect on the result. In practice, however, this is a delicate process which deserves some discussion.

There are two primary issues to be considered, namely,

- how can the effect of the isotopes at a certain rank in the chain be calculated, and,
- how the significance of that effect can be determined.

The easiest way to calculate the effect of a particular isotope in a chain is to simply solve the problem including that isotope. This would require solving the entire problem twice, however, drastically affecting the speed of the problem. The first enhancement, therefore, is to perform this reference calculation only once for each initial isotope using a flux which is somehow representative of all the spatial points which contain that initial isotope (see Section 3.4). If every point in space has a unique set of initial isotopes, this still leads to solving the problem twice, but as the problems become more complex, with perhaps 50 or more spatial points sharing the same mixture definition, the savings are significant.

What is the best flux to represent all the spatial points which share an initial isotope? Since higher fluxes will tend to maximize the amount of transmutation from one isotope to another, the obvious choice is some flux which is a maximum bound for the problem. Since the flux is group-wise and it is possible that one spatial point will have the highest fast flux while another point has the highest slow neutron flux, the best choice for a bounding flux should be the group-wise maximum flux of all the points which share an initial isotope. This reference flux can then be used to solve the problem for a particular chain as it is being created.

If this truncation reference calculation assumes that a unit quantity of the initial isotope exists, then the solution gives the relative production of each isotope in the chain. A user specified tolerance can then be used to determine how significant the relative production of the last isotope in that chain is, and decide whether to continue the creation of the chain. This too, however, is more subtle than first appearances suggest.

Consideration must be given as to how the user specified tolerance is to be interpreted. The best interpretation of the many possibilities is that of an *atom loss tolerance*. Due to the fact that we are truncating a theoretically infinite chain, atoms will be lost from the physical model through branches leaving the last isotope in the chain. It is best, therefore, to use this truncation tolerance to minimize this atom loss. Since it is also possible for this last isotope to have a low relative production rate, yet still have a high loss of atoms through its branches, it must be possible to calculate not just the relative production of the last isotope in a chain, but the atom loss through this last isotope. Fortunately, this is quite simple. By temporarily zeroing the destruction rates of this last isotope, the result of the reference calculation will be the total relative production of all isotopes in the subtree rooted by this isotope, and therefore represents the maximum possible atom loss through this isotope. This value can then be compared to the user specified tolerance, which is now interpreted as the maximum atom loss in any chain.

There are obviously two mechanisms of atom loss: transmutation and decay. The former is only possible when there is a neutron flux present but the latter occurs throughout the operation lifetime as well as after the shutdown of the device. This difference is important since the after-shutdown lifetimes (or cooling times) can be much longer than the operation lifetimes and it is during these times that the activity is often most important. To guard against these differences, it is necessary to compare the relative production both at shutdown and at each after-shutdown time. If the relative atom loss is less than the tolerance at shutdown, but greater than the tolerance at any of the after shutdown times, it may represent an important atom loss path during shutdown. It is here that we can distinguish between atom loss mechanisms. Since the only atom loss mechanism after shutdown is decay, only subsequent decay branches are important, even if the relative atom loss is greater than the tolerance.

The results of this philosophy are quite simple. First, any relative atom loss which exceeds the tolerance at shutdown will result in a continuation of the chain. Second, if the relative atom loss is less than the tolerance both at shutdown and at all after-shutdown times, then the chain should be totally truncated at this point. Finally, if the relative atom

loss is less than the tolerance at shutdown and greater at some after-shutdown time, then all transmutation branches in that subtree should be truncated but decay branches followed.

This truncation approach affords some rudimentary error estimates for the results. Using an analogy to experiment, the user specified tolerance provides a measure for the *precision* of the calculation. The smallest possible correction and hence the largest possible error for the results for any one isotope is this truncation tolerance. This will also be the dominant source of physical modeling error in the result. Since these same truncation rules are used indiscriminately for truncating straightened loops, the error from truncation will always be greater than the error caused by not using more corrections to the loop. Thus, following the analogy to experiment, the *accuracy* of loop solutions is affected by the number of corrections while the precision is affected by truncation tolerance. Since the number of corrections is determined indirectly by the truncation tolerance, this methodology provides the most consistency across the entire problem.

3.1.3. Alternatives and Extensions to the Truncation Philosophy

The full implementation of this philosophy does have detrimental effects on the speed of the solution. The most significant drag is caused by the full pulsing solution of the chain for each truncation calculation. An alternative is to combine the reference flux concept with that of a reference time, a representative steady-state simulation time to use only for truncation calculations returning to the exact pulsing solution when performing the final solution. When using this alternate method, it is important to understand the full implications. In particular, even if the chosen reference time approximates the operation history well, the reference calculation includes no after-shutdown history, a period in which many isotopic compositions may change. The user should always consider how sensitive the solution is to this reference time.

Another source of drag is the solution of completely negligible data at the truncation point. Considering the precision and extent of the available data, it is possible to reach a point in the chain where the relative atom loss is many orders of magnitude less than the truncation tolerance. While it is obvious that the chain should be truncated, the full solution of this *pc*-isotope will probably lead to a negligible contribution. A second user defined tolerance, known as an ignore tolerance, can be used to determine when a truncation point should be ignored completely and the chain creation procedure should continue without

performing the complete solution of this *pc*-isotope. To ignore all truncation points, an ignore tolerance of 1 should be used and to ignore none, an ignore tolerance of 0.

This truncation method provides the optimum combination of speed and accuracy. In general, using a relative atom loss is the most accurate way to physically model the system and allow the user to get a useful measure of the precision of the results.[1] Faster approximations could be made to conservatively estimate the relative lost atom production. However, while conservatisms may seem appropriate for the physical model, they can lead to the physical model being too large, causing the time required for mathematical solution to be greater. It should also be noted that this measure of the truncation error in the physical model is an upper bound since a group-wise maximum flux is being used for the calculations. In many spatial regions, the actual production in the final solution may be many of orders of magnitude less.

3.2. Pulsing Representation

Modern fusion reactor designs for both power and experimental reactors include either the ability or necessity for pulsed or intermittent operation. This has important effects on the calculation of induced activity since the radioactive isotopes which may be produced during the pulses can decay between them. When an accurate pulsed solution is calculated and compared to steady-state approximations,[9] the errors can be significant.

For most systems the pulsing scheme will follow a somewhat regular work schedule. As an example, consider an experimental reactor which is designed to operate for 10 minutes every half hour during the work day. If each operation period is neutronically identical, this could easily be modeled for a 5 year period as in Table 3.1. This kind of pulsing history can be modeled exactly by ALARA, solving the problem efficiently through the use of matrix methods.

Furthermore, these pulsing schedules may change over time. For example, after 5 years, the irradiation time may increase or the flux may change. In addition, the pulsing schedule itself may change due to such things as changing maintenance requirements or different shifts. The current version of ALARA allows any number of uniform pulsing schedules to be sequentially defined, each of which can have a unique pulse height and width and a unique set of pulsing levels.

Table 3.1. Example pulsing schedule for experimental device.

Description	Time	# of Pulses
Pulse length	10 min	
Operation dwell	20 min	16 (half-hour segments)
Nightly dwell	16 h 20 min	5 (work days)
Weekend dwell	64 h 20 min	49 (weeks without maintenance)
Annual Maintenance	3 weeks 64 h 20 min	5 (years)

3.3. Spatial Variations

As mentioned above, most systems will have gross variations in material composition from one spatial point to another. The existence of any material will also mean that the flux can experience similar variations. It is most convenient for the user if this can be modeled entirely within a single run of the code, rather than requiring that the code be run many times, once for each point in space. This requires an algorithm which will not cause the solution to become slow and inefficient.

ALARA accomplishes these goals. The space can be divided in up to three dimensions and a number of different geometries into *zones* whose boundaries are the transition from one material composition to another. Each zone contains a defined isotopic composition and so a *mixture* can be thought of as the possibly disjoint set of all zones which have the same initial isotopic composition. Each zone can be further subdivided into *intervals*, each allowed to have a different flux spectrum. It is important that the data handling of these spatial variations be implemented efficiently, as described briefly in the next section.

3.4. Implementation of Physical Modeling Techniques

To implement these various methods and techniques efficiently, it is first necessary to optimally cross-reference the physical model. In particular, many different mixture definitions could, in practice, consist of overlapping sets of initial isotopes. If the solution is found by looping through these mixtures, the chain information for the same root isotopes would either have to be recalculated or stored, having a severe impact on either the speed or memory resource, respectively. The solution is to create a global list of unique isotopes which

are cross-referenced to the relevant mixture definitions for the number density information, which are, in turn, cross-referenced to the appropriate intervals for the local flux information. With this list in place, the solution can be found by looping through each of the globally unique root isotopes. The reference truncation flux must be found across all the intervals containing the same root isotope, by accessing the cross-referenced lists of intervals from the cross-referenced list of mixtures. Finally, as alluded to above, as each chain is solved and its relevant solution information stored, the chain information itself can be discarded reducing any storage requirements.

Also, because the solution for each interval contains information about a large potential number of isotopes in the final composition, an efficient data structure was needed to store this data. The solution was a linked list in which each list item contained an identifier for the isotope, the final number densities at shutdown and the various after-shutdown times, and decay information relevant for the calculation of activities and decay heats.

4. Mathematical Technique and Theory

The mathematical problem which results from the physical problem described in Chapter 3 is, at first glance, a very simple one. If the original decay scheme (without removing cross-links and straightening loops – Figure 3.1) is converted directly to its mathematical equivalent, the result is a compact but potentially stiff system of linear first order ordinary differential equations [ODE's],

$$\begin{aligned}\dot{\vec{N}}(t) &= \boldsymbol{A}\vec{N}(t) \\ &= \begin{bmatrix} -d_1 & P_{2\to1} & P_{3\to1} & \cdots & \cdots & P_{l\to1} \\ P_{1\to2} & -d_2 & P_{3\to2} & \cdots & \cdots & P_{l\to2} \\ P_{1\to3} & P_{2\to3} & -d_3 & \cdots & \cdots & P_{l\to3} \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & -d_{l-1} & P_{l\to l-1} \\ P_{1\to l} & P_{2\to l} & P_{3\to l} & \cdots & P_{l-1\to l} & -d_l \end{bmatrix} \cdot \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ \vdots \\ N_l \end{bmatrix},\end{aligned} \tag{4.1}$$

where:

$$\begin{aligned}\vec{N} &\equiv \text{number densities, } N_i, \text{ of all isotopes} \\ d_i &\equiv \text{destruction rate of isotope } i \\ P_{i\to j} &\equiv \text{production rate of isotope } j \text{ from isotope } i.\end{aligned}$$

After loop straightening and cross-link removal has been performed, the result is a somewhat larger, simpler set of ODE's:

$$\begin{aligned}\dot{\vec{N}}(t) &= \boldsymbol{B}\vec{N}(t) \\ &= \begin{bmatrix} -d_1 & 0 & 0 & \cdots & \cdots & 0 \\ P_{1\to2} & -d_2 & 0 & \cdots & \cdots & 0 \\ P_{1\to3} & P_{2\to3} & d_3 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & -d_{m-1} & 0 \\ P_{1\to m} & P_{2\to m} & P_{3\to m} & \cdots & P_{m-1\to m} & -d_m \end{bmatrix} \cdot \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ \vdots \\ N_m \end{bmatrix}.\end{aligned} \tag{4.2}$$

This lower triangular matrix is quite sparse with a maximum of two entries in each row since each isotope has only one production path and one total destruction rate.

Finally, if this physical model is further broken into the previously described linear chains, many small sets of ODE's are created with special simplifying characteristics,

$$\begin{aligned}\dot{\vec{N}}(t) &= \boldsymbol{C}\vec{N}(t) \\ &= \begin{bmatrix} -d_1 & 0 & 0 & \cdots & \cdots & 0 \\ P_{1\to 2} & -d_2 & 0 & \cdots & \cdots & 0 \\ 0 & P_{2\to 3} & -d_3 & \cdots & \cdots & 0 \\ 0 & 0 & P_{3\to 4} & -d_4 & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & 0 & \cdots & P_{k-1\to k} & -d_k \end{bmatrix} \cdot \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ \vdots \\ \vdots \\ N_k \end{bmatrix}.\end{aligned} \tag{4.3}$$

The bidiagonal nature of these matrices is an important factor in subsequent derivations and calculations.

In all cases, the generic solution takes the form

$$\vec{N}(t) = \boldsymbol{T}\vec{N}_o(t), \tag{4.4}$$

where $\boldsymbol{T}$ is the exponential of the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, or $\boldsymbol{C}$, depending on which method is used (e.g. $\boldsymbol{T} = e^{\boldsymbol{A}t}$).

It is interesting to compare the sizes of these three matrices as it gives some initial insight into the efficiency of the solution. To compare the size of the original scheme, l, with that of the straightened tree, m, is not easy. Since the physical conversion is to convert cross-links and loops into *pc*-isotopes, it is clear that $m > l$; however, the severity of this inequality is difficult to determine. An approximate comparison between k and m, however, can be made. Since $\boldsymbol{B}$ represents a true tree structure, it can be analyzed by assuming that it is a full n-ary tree, that is, assuming that each *pc*-isotope in the tree has the same number of branches. Since k is the depth of such a tree, there will be n^{k-1} chains representing the $m = \frac{n^k-1}{n-1} \approx O(n^{k-1})$ nodes of the tree. The mathematical operations performed on these matrices are generally at least of order of the square of the matrix dimension, and often the cube. Thus, for matrix $\boldsymbol{B}$, the mathematical costs will be at least $O(n^{2k-2})$ and possibly $O(n^{3k-3})$ or higher. On the other hand, for the n^{k-1} matrices $\boldsymbol{C}$, the mathematical costs will be $O(k^2n^{k-1})$ or $O(k^3n^{k-1})$. This very rough analysis shows that for mathematical

operations of order x, as long as $n^{1-\frac{1}{x}} > k^{\frac{1}{k-1}}$, the linear chain method will be more efficient. While it may be difficult to visualize this relationship, it can be used to define some limits. For second order operations, for $n \geq 4$, the linear chain method is always more efficient. For third order operations, only $n \geq 3$ is required. On the other hand, for $n = 2$, the chain depth, k, must be greater than 2, and even $k = 3$ requires 5^{th} order operations for the linear chains to be more efficient. Finally, for a real problem in which $n \geq 4$, the linear chain method is more efficient for all orders of solution.

It is important to note that not only is this analysis not rigorous, but the order of the mathematical solution is itself dependent on the method which is used. Thus, the analysis gives a first glance into the comparison of efficiency, but is hardly complete.

By transfering this system to the Laplace domain and considering each equation individually, it is possible to write the solution in a more compact form (N.B. P_i implies $P_{i-1\to i}$):

$$\tilde{N}_i = \frac{N_{i_0}}{s+d_i} + P_i\frac{\tilde{N}_{i-1}}{s+d_i} \tag{4.5}$$

$$\begin{aligned} &= \frac{N_{i_0}}{s+d_i} + \frac{N_{i-1_0}}{s+d_{i-1}}\frac{P_i}{s+d_i} + \frac{N_{i-2_0}}{s+d_{i-2}}\frac{P_{i-1}P_i}{(s+d_{i-1})(s+d_i)} + \cdots \\ &\quad + \frac{N_{2_o}}{s+d_2}\prod_{j=3}^{i}\frac{P_j}{s+d_j} + \frac{N_{1_o}}{s+d_1}\prod_{j=2}^{i}\frac{P_j}{s+d_j}, \end{aligned} \tag{4.6}$$

which can be written as

$$\begin{aligned} \tilde{N}_i &= \sum_{j=1}^{i}\tilde{N}_{ij} \\ &= \sum_{j=1}^{i} N_{j_o}\prod_{k=j+1}^{i}P_k\prod_{l=j}^{i}\frac{1}{s+d_l} \\ &= \sum_{j=1}^{i} N_{j_o}\tilde{F}_{ij}(s)\prod_{k=j+1}^{i}P_k. \end{aligned} \tag{4.7}$$

In this representation, the matrix $\boldsymbol{T}$ is filled by setting

$$\begin{aligned} T_{ij} &= N_{ij}/N_{j_o} \\ &= \mathcal{L}^{-1}\left[\tilde{F}_{ij}(s)\right] \prod_{k=j+1}^{i} P_k, \end{aligned} \tag{4.8}$$

and it becomes an exercise of solving for the inverse transform of the term

$$\tilde{F}_{ij}(s) = \prod_{l=j}^{i} \frac{1}{s + d_l}. \tag{4.9}$$

Normally, two such matrices, $\boldsymbol{T}$ and $\boldsymbol{D}$, are required to represent the pulse and dwell times, respectively. In the first case, all the destruction and production rates include the terms for neutron transmutation which are dependent on the flux spectrum and therefore it is only during this period in which loops can occur in the isotope tree. In the dwell period, the destruction and production rates are only those of decay, and therefore, many of the values will be zero.

4.1. Adaptive Mathematical Methods

Upon examining available methods to solve such lower bidiagonal systems, it can be seen that there are certain easily determined characteristics of the matrix which can be used to adaptively choose a method for each linear chain which will optimize the speed and accuracy of the solution.

The eigenvalues of these matrices are simply the diagonal elements, which in turn are the destruction rates of each isotope in the linear chain being modelled. True degeneracies in these values will occur only when the chain being modelled by this matrix has straightened loops and a simple test can determine whether this is the case. If no loop exists, this bidiagonal system is mathematically identical to the system solved by Bateman[13] many years ago, and many accurate and efficient methods exist to find the solution. This method is always used to calculate the transfer matrix for the dwell period since loops cannot exist. On the other hand, if loops do exist, the solution is somewhat more complicated and is facilitated by this transformation to the Laplace space. The next adaptive decision determines which method will be used to perform the Laplace inversion and is based on an analysis of the

spectral radius of the matrix, $\boldsymbol{C}$: for sufficiently small radii, a series method can be used, while in other cases, a direct and analytic inversion is preferred.

Because the mathematical method is chosen independently and adaptively for each chain, the efficiency and accuracy of the entire solution is optimized. In particular, if a small loop occurs in one small portion of a scheme, it is not necessary to perform the slower Laplace based calculations on the entire problem when the Bateman solution is available for most of the problem. On the other hand, the solution is not limited by the non-loop Bateman solution when loops do exist.

4.2. The Analytical Bateman Solution

If all the destruction rates, d_i, are distinct, Equation 4.9 can be easily inverted,

$$f_{ij}(t) = \sum_{l=j}^{i} e^{-d_l t} \prod_{\substack{m=j \\ m\neq l}}^{i} \frac{1}{d_m - d_l}. \tag{4.10}$$

This leads to a compact representation of the solution to the Bateman equations:

$$N_i(t) = N_{i_o} e^{-d_i t} + \sum_{j=1}^{i-1} N_{j_o} \left[\sum_{k=j}^{i-1} \frac{P_{k+1}(e^{-d_k t} - e^{-d_i t})}{d_i - d_k} \prod_{\substack{l=j \\ l\neq k}}^{i-1} \frac{P_{l+1}}{d_l - d_k} \right]. \tag{4.11}$$

Finally, we can write the transfer matrix elements as:

$$\begin{aligned} T_{ii} &= e^{-d_i t} \\ T_{\substack{ij \\ i\neq j}} &= \sum_{k=j}^{i-1} \frac{P_{k+1}(e^{-d_k t} - e^{-d_i t})}{d_i - d_k} \prod_{\substack{l=j \\ l\neq k}}^{i-1} \frac{P_{l+1}}{d_l - d_k}. \end{aligned} \tag{4.12}$$

4.3. Laplace Inversion Method

When the destruction rates (eigenvalues) are not distinct, other methods of solving Equation 4.9 are required. In general, however, because of the bidiagonal nature of the matrix representation (that is, because of the linear chain physical representation), this is a simple

problem which, for a small system, can easily be solved on paper by hand. In particular, this Laplace space representation can be directly inverted to the time representation.

For repeated poles, one uses the residue theorem to determine the coefficients for each term in a partial fractions expansion. Each of those terms would result in an exponential, perhaps multiplied by a polynomial in time, t, when converted back to the time domain. Those residues are calculated using one of two simple rules.

If the pole is not repeated, then the residue, R_k, for the pole, $-d_k$, is calculated as

$$R_k = \lim_{s \to d_k} (s + d_k) \tilde{F}_{ij}(s) \tag{4.13}$$

which becomes $R_k e^{-d_k t}$ in the time domain. If all poles have a singular multiplicity, the solution reduces exactly to the Bateman solution, and can be represented in many ways. This is obviously the solution with no loops.

If the pole is repeated m times, under a partial fraction expansion, this becomes m terms in that expansion:

$$\frac{R_{km}}{(s + d_k)^m} + \frac{R_{km-1}}{(s + d_k)^{m-1}} + \cdots + \frac{R_{k1}}{(s + d_k)} \tag{4.14}$$

which becomes

$$e^{-d_k t} \left(R_{km} \frac{t^{m-1}}{(m-1)!} + R_{k,m-1} \frac{t^{m-2}}{(m-2)!} + \cdots + R_{k2} \frac{t}{1!} + R_{k1} \right) \tag{4.15}$$

in the time domain. In this case, the residues are found using:

$$R_{kn} = \frac{1}{(m-n)!} \lim_{s \to d_k} \frac{d^{m-n}}{ds^{m-n}} \left[(s + d_k)^m \tilde{F}_{ij}(s) \right]. \tag{4.16}$$

This latter rule requires the ability to evaluate derivatives of a generic function:

$$\tilde{G}^k_{ij}(s) = (s + d_k)^m \tilde{F}_{i,j}(s) \tag{4.17}$$

at values of $s = -d_k$ for all i. By examining the successive derivatives of $\tilde{G}(s)$ it can be shown (see Appendix A) that these derivatives can be recursively defined as:

$$\left[\tilde{G}^k_{ij}(s)\right]^{(n)} = \sum_{j=1}^{n}(-1)^j \frac{(n-1)!}{(n-j)!}\left[\tilde{G}^k_{ij}(s)\right]^{(n-j)} \sum_{i=1}^{I}\frac{1}{(s+d_i)^j} \tag{4.18}$$

and this, in turn, can be converted to a computational numerical algorithm, allowing the entire problem to be solved.

We will call this method the Laplace Inversion Method.

4.4. Laplace Expansion Method

Alternately, an expansion in $1/s$ may be desirable in some cases. There are cases in which the above method might amplify roundoff errors due to division by small numbers which result from the subtraction of two similar numbers (such as when two of the poles are very near each other), but such divisions can be eliminated by writing the solution as a difference of exponentials, expanding that difference, factoring out the offending term from the numerator and canceling. While this seems like a monumental task to perform on an arbitrary problem, thanks to the bidiagonal nature of the system, it is again quite simple to implement. If we start again with our function:

$$\tilde{F}_{ij}(s) = \prod_{l=j}^{i}\frac{1}{s+d_l} \tag{4.19}$$

and making no assumptions about the multiplicity of the poles, we expand this as a series in $1/s$, the result is:

$$\begin{aligned}
\tilde{F}_{ij}(s) &= \frac{1}{s^{i-j+1}}\prod_{l=j}^{i}\frac{1}{1+\frac{d_l}{s}} \\
&= \frac{1}{s^{i-j+1}}\prod_{l=j}^{i}\left(1-\frac{d_l}{s}+\frac{d_l^2}{s^2}-\frac{d_l^3}{s^3}+\cdots\right) \\
&= \frac{1}{s^{i-j+1}}\left[1-\frac{\sum_{l=j}^{i}d_l}{s}+\frac{\sum_{l=j}^{i}d_l\sum_{k=l}^{i}d_k}{s^2}-\frac{\sum_{l=j}^{i}d_l\sum_{k=l}^{i}d_k\sum_{m=k}^{i}d_m}{s^3}+\cdots\right].
\end{aligned} \tag{4.20}$$

If $n = i - j$, in the time domain, this becomes:

$$f_{ij}(t) = t^n \left[\frac{1}{n!} - \frac{t}{(n+1)!} \sum_{l=j}^{i} d_l + \frac{t^2}{(n+2)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \right. \\ \left. - \frac{t^3}{(n+3)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \sum_{m=k}^{i} d_m + \cdots \right] \tag{4.21}$$

and thus:

$$\begin{aligned} T_{ii} &= e^{-d_i t} \\ T_{\substack{ij \\ i \neq j}} &= f_{ij}(t) \prod_{k=j}^{i-1} P_k \,. \end{aligned} \tag{4.22}$$

It is clear that this solution will only be computationally viable when the product, $\max\{d_i\} \cdot t$ is small. For arbitrary problems, this is only guaranteed when there are small times, but can easily be tested for a particular problem. Other representations can be formed, each providing different insight into the method (see Appendix B).

We will call this method the Laplace Expansion Method.

4.5. Mathematical Implementation with Pulsing History

As with the physical modeling, the implementation of the mathematical solution requires some special implementation to enhance its efficiency.

The first such enhancement is to store the solution matrices from one chain to another. The value of this implementation can be seen when considering the solution of many subsequent chains of large rank, n. If each chain differs in only the last branch, all the $n \times n$ transfer matrices would have to be completely recalculated ($n^2/2$ calculations) for each of these subsequent chains. On the other hand, if the transfer matrix solution for each interval is stored after its use in the solution of one chain, only the last row (n calculations) is necessary. This savings carries to all situations throughout the scheme. Since the chains are formed by a depth-first search, it is likely that for each chain, a significant portion of the data has already been determined for a previous calculation.

This is also true for the decay matrices between pulses and after shutdown, but in this case, the matrices are not only saved between chains, but between intervals for the same chain. Since the decay matrices are independent of flux, they need only be calculated once for each chain and then used in all the intervals for that chain. This has large potential savings since there may be many intervals which share the decay matrices, each recalculation of which would cost n calculations even with the already implemented savings. Furthermore, since the decay matrices are likely to be much sparser than the pulse transfer matrices, a special implementation of the Bateman solution routine to intelligently fill these matrices has been implemented.

Once these methods have been implemented to efficiently calculate the individual transfer matrices for each pulse, each inter-pulse decay period and each after-shutdown decay period, it is straightforward to calculate the total transfer matrix for the entire problem.

First, a single pulse transfer matrix, $\boldsymbol{T}_0$, and a single dwell matrix for the dwell time of the first level of pulsing, $\boldsymbol{D}_1$, are calculated. The product of these, $\boldsymbol{D}_1\boldsymbol{T}_0$, is then raised to a power representing the number of pulses in that level, n_1. This becomes the transfer matrix for the next level, $\boldsymbol{T}_1 = \boldsymbol{T}_0(\boldsymbol{D}_1\boldsymbol{T}_0)^{n_1}$. Now a single dwell matrix for the dwell time of the second level is calculated, $\boldsymbol{D}_2$. This is repeated using the general formula[11]

$$\boldsymbol{T}_i = \boldsymbol{T}_{i-1}(\boldsymbol{D}_i\boldsymbol{T}_{i-1})^{n_i}. \tag{4.23}$$

It is important to use an efficient algorithm[14] for this matrix exponentiation process since it will be performed so often during the operation of the code.

For N levels of pulsing, the matrix, $\boldsymbol{T}_N$, will be the transfer matrix for the entire problem history up to shutdown. Multiplying this matrix by the initial number density, $\vec{N}_o$, results in a final, at-shutdown number density. This final number density vector can then be multiplied by a single dwell matrix for each after-shutdown time to determine the appropriate isotopic compositions.

Acknowledgements

Much of the development used to create ALARA, and in particular, the handling of loops, was inspired by vigorous discussion with the late Prof. Emeritus Charles Maynard. He always insisted that loops were usually not important, and when they were, the solution could be easily found. I am happy to have proven him correct.

In addition, Jim Sisolak, Jeff Crowell and Prof. Jake Blanchard have all, at times, been good listeners to help me focus my ideas and develop my methods.

Partial support for this work was provided by the Fusion Technology Institute of the University of Wisconsin-Madison under DOE Grants DE-FG02-94ER54244 and DE-AS08-88DP10754 and Sandia National Laboratory contract Al-7232.

References

[1] P.P.H. Wilson and D.L. Henderson, "Qualitative Analysis of Physical and Mathematical Approximations Necessary for Induced Radioactivity Calculations of Fusion Devices", *Fusion Eng. and Design*, **36**, 415 (1997).

[2] T.Y. Sung and W.F. Vogelsang, "DKR: A Radioactivity Calculation Code for Fusion Reactors," UWFDM-170, Fusion Technology Institute, University of Wisconsin-Madison, Madison, Wisconsin, September 1976.

[3] D.L. Henderson and O. Yasar, "DKRICF: A Radioactivity and Dose Rate Calculation Code Package: Vols. I & II," UWFDM-714, Fusion Technology Institute, University of Wisconsin-Madison, Madison, Wisconsin, November 1986. This code package is available from the Radiation Shielding Information Center (RSIC) at Oak Ridge National Laboratory as Computer Code Collection entry CCC-323-DKR.

[4] DKR-PULSAR is a new version of the DKR-ICF code which implements methods from Reference 9 for the exact treatment of pulsed history irradiation. It is being developed by D.L. Henderson and H. Khater at the University of Wisconsin-Madison.

[5] P.P.H. Wilson and D.L. Henderson, "Expanding Towards Excellence: Ironing Out DKR's Wrinkles", UWFDM-995, University of Wisconsin Fusion Technology Institute, Madison, Wisconsin, November 1995.

[6] R.A. Forrest and J-Ch. Sublet, "FISPACT3 - User Manual," AEA/FUS 227, April 1993.

[7] J. Jung, "RACC: Theory and Use of the Radioactivity Code RACC," Argonne National Laboratory Report: ANL/FPP/TM-122, May 1979.

[8] H. Attaya, "Input Instructions for RACC-P," Argonne National Laboratory Report: ANL/FPP/TM-270, September 1994.

[9] J.E. Sisolak, S.E. Spangler and D.L. Henderson, "Pulsed/Intermittent Activation in Fusion Energy Systems," *Fus. Tech.*, **21**, 2145 (May 1992).

[10] P.P.H. Wilson and D.L. Henderson, "ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis, Volume II, Users' Guide," UWFDM-1071, University of Wisconsin Fusion Technology Institute, Madison, Wisconsin, January 1998.

[11] S.E. Spangler, J.E. Sisolak and D.L. Henderson, "Calculational Models for the Treatment of Pulsed/Intermittent Activation Within Fusion Energy Devices," *Fusion Eng. and Design*, **22**, 349 (July 1993).

[12] S.E. Spangler, Master's Thesis, "A Numerical Method for Calculating Nuclide Densities in Pulse Activation Studies," University of Wisconsin-Madison, 1991.

[13] H. Bateman, *Proc. Cambridge Phil. Soc.* **15**, 423 (1910).

[14] E.S. Lee, "Computer Engineering: Computer Algorithms, Data Structures, and Languages," Prepared Notes, University of Toronto, 1989.

A. Derivation of Recursive Derivative Definition

$$G(s) = \prod_{i=1}^{N} \frac{1}{s+d_i} \tag{A.1}$$

$$\begin{aligned} G'(s) &= \sum_{j=1}^{N} \frac{-1}{s+d_j} \prod_{i=1}^{N} \frac{1}{s+d_i} \\ &= -G(s) \sum_{j=1}^{N} \frac{1}{s+d_j} \end{aligned} \tag{A.2}$$

$$G''(s) = -G'(s) \sum_{j=1}^{N} \frac{1}{s+d_j} + G(s) \sum_{j=1}^{N} (s+d_j)^{-2} \tag{A.3}$$

$$\begin{aligned} G'''(s) &= -G''(s) \sum_{j=1}^{N} \frac{1}{s+d_j} + G'(s) \sum_{j=1}^{N} (s+d_j)^{-2} \\ &\quad + G'(s) \sum_{j=1}^{N} (s+d_j)^{-2} - 2G(s) \sum_{j=1}^{N} (s+d_j)^{-3} \\ &= -G''(s) \sum_{j=1}^{N} \frac{1}{s+d_j} + 2G'(s) \sum_{j=1}^{N} (s+d_j)^{-2} \\ &\quad - 2G(s) \sum_{j=1}^{N} (s+d_j)^{-3} \end{aligned} \tag{A.4}$$

$$\begin{aligned} G''''(s) &= -G'''(s) \sum_{j=1}^{N} \frac{1}{s+d_j} + G'''(s) \sum_{j=1}^{N} (s+d_j)^{-2} \\ &\quad + 2G''(s) \sum_{j=1}^{N} (s+d_j)^{-2} - 4G'(s) \sum_{j=1}^{N} (s+d_j)^{-3} \\ &\quad - 2G'(s) \sum_{j=1}^{N} (s+d_j)^{-3} + 6G(s) \sum_{j=1}^{N} (s+d_j)^{-4} \\ &= -G''''(s) \sum_{j=1}^{N} \frac{1}{s+d_j} + 3G''(s) \sum_{j=1}^{N} (s+d_j)^{-2} \\ &\quad - 6G'(s) \sum_{j=1}^{N} (s+d_j)^{-3} + 6G(s) \sum_{j=1}^{N} (s+d_j)^{-4} \end{aligned} \tag{A.5}$$

Thus, for n=4,

$$\begin{aligned} G^{(n)}(s) &= -\frac{(n-1)!}{(n-1)!}G^{(n-1)}(s)\sum_{j=1}^{N}\frac{1}{s+d_j} \\ &\quad + \frac{(n-1)!}{(n-2)!}G^{(n-2)}(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad - \frac{(n-1)!}{(n-3)!}G^{(n-3)}(s)\sum_{j=1}^{N}(s+d_j)^{-3} \\ &\quad + \frac{(n-1)!}{(n-4)!}G^{(n-4)}(s)\sum_{j=1}^{N}(s+d_j)^{-4} \\ &= \sum_{i=1} n(-1)^i\frac{(n-1)!}{(n-i)!}G^{(n-i)}(s)\sum_{j=1}^{N}(s+d_j)^{-i} \end{aligned} \tag{A.6}$$

A.1. Induction Proof

$$G^{(n)}(s) = \sum_{i=1}^{n}(-1)^i\frac{(n-1)!}{(n-i)!}G^{(n-i)}(s)\sum_{j=1}^{N}(s+d_j)^{-i} \tag{A.7}$$

given

$$G^{(0)}(s) = G(s) = \prod_{j=1}^{N}(s+d_j)^{-1} \tag{A.8}$$

First, we solve for n=1:

$$\begin{aligned} G'(s) &= (-1)\frac{0!}{0!}G(s)\sum_{j=1}^{N}(s+d_j)^{-1} \\ &= -G(s)\sum_{j=1}^{N}(s+d_j)^{-1} \end{aligned} \tag{A.9}$$

which matches Equation A.2.

Now, we solve for n=2:

$$\begin{aligned} G''(s) &= (-1)\frac{1!}{1!}G'(s)\sum_{j=1}^{N}(s+d_j)^{-1} + \frac{1!}{0!}G(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &= -G'(s)\sum_{j=1}^{N}(s+d_j)^{-1} + G(s)\sum_{j=1}^{N}(s+d_j)^{-2} \end{aligned} \tag{A.10}$$

which matches Equation A.3.

Now, given $G^{(k)}(s)$, we take the derivative, $G^{(k+1)}(s)$, and see if it matches the correct form:

$$G^{(k+1)}(s) = \sum_{i=1}^{k}(-1)^i\frac{(k-1)!}{(k-i)!}\left[G^{(k-i+1)}\sum_{j=1}^{N}(s+d_j)^{-i} - iG^{(k-i)}\sum_{j=1}^{N}(s+d_j)^{-(i+1)}\right] \tag{A.11}$$

letting $l = k+1$:

$$\begin{aligned} G^{(l)}(s) = \quad & \sum_{i=1}^{l-1}(-1)^i\frac{(l-2)!}{(l-i-1)!}G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \\ & - \sum_{i=1}^{l-1}(-1)^i\frac{(l-2)!}{(l-i-1)!}iG^{(l-i-1)}\sum_{j=1}^{N}(s+d_j)^{-(i+1)} \end{aligned} \tag{A.12}$$

Now, letting $m = i+1$ in the second sum:

$$\begin{aligned} G^{(l)}(s) = \quad & \sum_{i=1}^{l-1}(-1)^i\frac{(l-2)!}{(l-i-1)!}G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \\ & + \sum_{m=2}^{l}(-1)^m\frac{(l-2)!}{(l-m)!}(m-1)G^{(l-m)}\sum_{j=1}^{N}(s+d_j)^{-m} \end{aligned} \tag{A.13}$$

and recombining the sums:

$$
\begin{aligned}
G^{(l)}(s) = &-\frac{(l-2)!}{(l-2)!}G^{(l-1)}(s)\sum_{j=1}^{N}(s+d_j) \\
&+\sum_{i=2}^{l-1}(-1)^i\left[\frac{(l-2)!}{(l-i-1)!}+(i-1)\frac{(l-2)!}{(l-i)!}\right]G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \qquad \text{(A.14)} \\
&+(-1)^l(l-2)!(l-1)G(s)\sum_{j=1}^{N}(s+d_j)^{-l} \\
= &-G^{(l-1)}(s)\sum_{j=1}^{N}(s+d_j)^{-1} \\
&+\sum_{i=2}^{l-1}(-1)^i\left[(l-i)\frac{(l-2)!}{(l-i-1)!(l-i)}+(i-1)\frac{(l-2)!}{(l-i)!}\right]G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \\
&+(-1)^l(l-1)!G(s)\sum_{j=1}^{N}(s+d_j)^{-l} \qquad \text{(A.15)} \\
= &-G^{(l-1)}(s)\sum_{j=1}^{N}(s+d_j)^{-1} \\
&+\sum_{i=2}^{l-1}(-1)^i\frac{(l-1)!}{(l-i)!}G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \qquad \text{(A.16)} \\
&+(-1)^l(l-1)!G(s)\sum_{j=1}^{N}(s+d_j)^{-l} \\
= &\sum_{i=1}^{l}(-1)^i\frac{(l-1)!}{(l-i)!}G^{(l-i)}\sum_{j=1}^{N}(s+d_j)^{-i} \qquad \text{(A.17)}
\end{aligned}
$$

QED.

B. Other Forms of $1/s$ Expansion

The $1/s$ expansion from Section 4.4 can take on many slightly different forms providing different methods for determining the coefficients. First, it is instructive to relate the expansion as shown in Equation 4.21 to a simple difference of exponentials. Starting with the Bateman solution (Equation 4.10) for a single matrix element,

$$T_{31} = \frac{P_2(e^{-d_1 t} - e^{-d_3 t})}{d_3 - d_1}\frac{P_3}{d_2 - d_1} + \frac{P_3(e^{-d_2 t} - e^{-d_3 t})}{d_3 - d_2}\frac{P_2}{d_1 - d_2} \tag{4.12}$$

and using the standard expansion for the exponential, we get

$$\begin{aligned}
&= P_2P_3\left[\frac{1 - d_1t + \frac{(d_1t)^2}{2} - \frac{(d_1t)^3}{6} - 1 + d_3t - \frac{(d_3t)^2}{2} + \frac{(d_3t)^3}{6} + \dots}{(d_3 - d_1)(d_2 - d_1)}\right.\\
&\qquad\left. + \frac{1 - d_2t + \frac{(d_2t)^2}{2} - \frac{(d_2t)^3}{6} - 1 + d_3t - \frac{(d_3t)^2}{2} + \frac{(d_3t)^3}{6} + \dots}{(d_3 - d_2)(d_1 - d_2)}\right]\\
&= P_2P_3\left[\frac{(d_3 - d_1)[t - (d_3 + d_1)\frac{t^2}{2} + (d_3^2 + d_3d_1 + d_1^2)\frac{t^3}{6} + \dots]}{(d_3 - d_1)(d_2 - d_1)}\right.\\
&\qquad\left. + \frac{(d_3 - d_2)[t - (d_3 + d_2)\frac{t^2}{2} + (d_3^2 + d_3d_2 + d_2^2)\frac{t^3}{6} + \dots]}{(d_3 - d_2)(d_1 - d_2)}\right]\\
&= P_2P_3\left[\frac{(d_2 - d_1)\frac{t^2}{2} + [d_3(d_1 - d_2) + (d_1^2 - d_2^2)]\frac{t^3}{6} + \dots}{d_2 - d_1}\right]\\
&= P_2P_3\left[\frac{t^2}{2} - (d_3 + d_2 + d_1)\frac{t^3}{6} + \dots\right]\\
&= P_2P_3\, t^2\left[\frac{1}{2} - \frac{t}{6}(d_3 + d_2 + d_1) + \dots\right]
\end{aligned} \tag{B.1}$$

which has the form of Equation 4.21.

Whether in the Laplace transform domain or the time domain, there is a necessity to calculate coefficients of the form:

$$\{c_i\} = \left\{\sum_{j=1}^{N} d_j\ ,\ \sum_{j=1}^{N} d_j \sum_{k=j}^{N} d_k\ ,\ \sum_{j=1}^{N} d_j \sum_{k=j}^{N} d_k \sum_{l=k}^{N} d_l\ ,\ \dots\right\}. \tag{B.2}$$

A different form for these coefficients becomes apparent when $N = 2$ or $N = 3$. The coefficients, $\{c_i\}$, are:

$$\{c_i\} = \left\{d_1 + d_2\ ,\ d_1(d_1 + d_2) + d_2^2\ ,\ d_1\left[d_1(d_1 + d_2) + d_2^2\right] + d_2^3\ , \dots\right\} \tag{B.3}$$

or

$$\begin{aligned}\{c_i\} = \big\{&d_1 + d_2 + d_3,\ d_1(d_1 + d_2 + d_3) + d_2(d_2 + d_3) + d_3^2,\\ &d_1\left[d_1(d_1 + d_2 + d_3) + d_2(d_2 + d_3) + d_3^2\right] + d_2\left[d_2(d_2 + d_3) + d_3^2\right] + d_3^3\ , \dots\big\}.\end{aligned} \tag{B.4}$$

This shows the following pattern, assuming $\{\lambda_{0,j}\} = 1;\ j = [1, N]$:

$$\lambda_{ij} = \sum_{k=j}^{N} d_k \lambda_{i-1,k} \tag{B.5}$$

$$c_i = \lambda_{i1}. \tag{B.6}$$

This last form leads to an efficient way to calculate these coefficients using matrix multiplications. If we form a matrix, M, with elements $m_{ij} = d_j; j \geq i$:

$$\boldsymbol{M} = \begin{bmatrix} d_1 & d_2 & d_3 & \dots & d_N \\ 0 & d_2 & d_3 & \dots & d_N \\ 0 & 0 & d_3 & \dots & d_N \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_N \end{bmatrix}, \tag{B.7}$$

it is clear that $\lambda_1 = [\boldsymbol{M} \cdot \vec{1}]$ and that $\lambda_i = [\boldsymbol{M}^i \cdot \vec{1}]$. Therefore,

$$c_i = \lambda_{i1} = \left[M^i \cdot \vec{1}\right]_1. \tag{B.8}$$

Since the direct calculation of

$$\sum_{j_1=1}^{N} d_{j_1} \sum_{j_2=j_1}^{N} d_{j_2} \sum_{j_3=j_2}^{N} d_{j_3} \cdots \sum_{j_{n-1}=j_{n-2}}^{N} d_{j_{n-1}} \sum_{j_n=j_{n-1}}^{N} d_{j_n} = \prod_{l=n}^{1} \sum_{j_l=j_{l-1}}^{N} d_{j_l} \tag{B.9}$$

tends to require $O(N^n)$ calculations, the matrix method above will be highly advantageous since it requires only $O(nN^3)$ calculations.

Once these coefficients have been calculated, they are then used to calculate the time response using Equation 4.21:

$$f(t) = t^n \left[\frac{1}{n!} - \frac{t}{(n+1)!} \sum_{l=j}^{i} d_l + \frac{t^2}{(n+2)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k - \frac{t^3}{(n+3)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \sum_{m=k}^{i} d_m + \cdots \right]. \tag{4.21}$$

C. ALARA_DC: Data Conversion for Code Interfacing

In designing ALARA for maximum usability, consideration was taken for the way that it would be implemented by the end user. Most important was to consider the origins of the various inputs which would be needed to complete each calculation. Every ALARA problem requires four distinct types of input:

1. cross-section, decay and gamma libraries
2. geometry/mixture definitions
3. group-wise flux input, and
4. pulsing and history information.

By facilitating the conversion of these various inputs from other standard formats to that required by ALARA, the ease of use for the end user is enhanced. Such conversions are available for input types 1-3, while input type 4, pulsing and history information is particular to the pulsed history activation calculation. ALARA Data Conversion [ALARA_DC] is being written with these conversion needs in mind.

It was originally written to convert the cross-section, decay and gamma data from various international standard text formats to the proprietary data format required by ALARA (see Appendix D for binary data format). However, because the geometry definitions are required in various formats for neutron transport calculations which generate the group-wise fluxes in various formats, ALARA_DC is being extended to convert these data as well.

While most transport calculations require geometry definitions at least as specific as those needed by ALARA, the mixture definitions often lack certain trace elemental quantities which are unimportant for transport calculations but may be important for activation calculations. For example, a transport calculation through a steel block may define the mixture as iron for the purpose of the transport calculation while the alloying concentrations of nickel, chromium, and other elements are very important in determining the activation characteristics of the material. Therefore, ALARA_DC will be designed to extract the geometry definitions from the transport calculation and allow the user to modify/upgrade the mixture definition. In different cases, this geometry information may be extracted from either transport calculation input (deterministic calculations) or output (Monte Carlo calculations with combinatorial geometries).

The flux data can also take a number of different formats. Some of the available codes share standard binary and/or text based data formats for these results while others have their own formats. ALARA_DC will be extended to first convert the standard shared formats and then the more popular unique formats. In some cases, this conversion will be simultaneous to the geometry conversion, and in others, the fluxes will be interactively extracted from the available output.

D. Binary Reaction Library Format

Because the reaction schemes/chains are created by a depth first search using the data from the transmutation and decay libraries, these libraries need to be accessed extensively and randomly. In the past, such random access was not possible because of the limits on mass storage devices. Currently, in a text format, such random access would still be very tedious. To ensure that this random access does not create a drag on ALARA, it is necessary to either store the entire library in memory or use a binary file format. Because the libraries are often quite large (many MB) a simple binary format was designed. This section will describe the formats for the binary files and their indexes, which are generated in a text format and then appended in binary format to the end of the binary library. The transmutation library and decay library formats are no longer supported directly by ALARA. Instead, one transmutation library and one decay library must be combined into a single mixed or merged library format (see Section D.4 for difference).

The format of the binary file will be described by listing, in order, the data written to the file using the format: *(data type)*Description[**size**].

D.1. Transmutation Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Flag indicating existence of Group Boundary info[**1**]
- *(float)*Group Boundary Data[**Number of Groups + 1** if above flag]
- *(int)*Flag indicating existence of Integral Flux Data[**1**]
- *(float)*Integral Flux Data[**Number of Groups** if above flag].
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]
  - Reaction info once for each reaction
    - *(int)*Daughter KZA[**1**]

        * *(char)*Emitted Particles[**6**]
        * *(float)*Cross-section Data[**Number of Groups**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Special Code for Group Boundary Data[**1**]
- *(long)*File Index of Group Boundary Data[**1**]
- *(int)*Special Code for Integral Flux Data[**1**]
- *(long)*File Index of Integral Flux Data[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each reaction
    * *(int)*Daughter KZA[**1**]
    * *(char)*Emitted Particles[**6**]
    * *(long)*File Index of This Reaction[**1**]

D.2. Decay Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Decay Paths[**1**]
  - *(float)*Half Life[**1**]
  - *(float)*Average Beta Energy[**1**]
  - *(float)*Average Gamma Energy[**1**]

  - *(float)*Average Alpha Energy[**1**]
  - Reaction info once for each decay path
    * *(int)*Daughter KZA[**1**]
    * *(char)*Daughter Flag[**1**]
    * *(float)*Branching Ratio[**1**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Decay Paths[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each decay path
    * *(int)*Daughter KZA[**1**]
    * *(char)*Daughter Flag[**1**]
    * *(long)*File Index of This Decay Path[**1**]

D.3. Mixed Reaction Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Flag indicating existence of Group Boundary info[**1**]
- *(float)*Group Boundary Data[**Number of Groups + 1** if above flag]
- *(int)*Flag indicating existence of Integral Flux Data[**1**]
- *(float)*Integral Flux Data[**Number of Groups** if above flag].
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]

  - Reaction info once for each transmutation reaction
    * *(int)*Daughter KZA[**1**]
    * *(int)*Length of Emitted Information[**1**]
    * *(char)*Emitted Particles[**Length of Emitted Info**]
    * *(float)*Cross-section Data[**Number of Groups+1**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Special Code for Group Boundary Data[**1**]
- *(long)*File Index of Group Boundary Data[**1**]
- *(int)*Special Code for Integral Flux Data[**1**]
- *(long)*File Index of Integral Flux Data[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each reaction
    * *(int)*Daughter KZA[**1**]
    * *(int)*Length of Emitted Information[**1**]
    * *(char)*Emitted Particles[**Length of Emitted Info**]
    * *(long)*File Index of This Reaction[**1**]

Note: decay rate for each daughter is stored in extra last element of cross-section array.

D.4. Merged Reaction Library

The merged reaction library format is identical to the mixed format except any reaction channels which lead to the same daughter have been merged into a single reaction channel, with the information about emitted particles being concatenated into a list. See comments in Section 3.3 of the Users' Guide[10] about the varying usage of these two formats.

D.5. Gamma Source Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Spectra[**1**]
  - *(int)*Number of Discrete Gammas in each Spectra[**Number of Spectra**]
  - *(int)*Number of Interpolation Regions in each Spectra[**Number of Spectra**]
  - *(int)*Number of Interpolation Points in each Spectra[**Number of Spectra**]
  - Reaction info once for each spectrum
    - *(float)*Discrete Gamma Energies[**Number of Discrete Gammas(i)**]
    - *(float)*Discrete Gamma Intensities[**Number of Discrete Gammas(i)**]
    - *(float)*Interpolation Region Boundaries[**Number of Interpolation Regions(i)**]
    - *(float)*Interpolation Region Types[**Number of Interpolation Regions(i)**]
    - *(float)*Interpolation Point X-values[**Number of Interpolation Points(i)**]
    - *(float)*Interpolation Point Y-values[**Number of Interpolation Points(i)**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Spectra[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each Spectra
    - *(int)*Number of Discrete Gammas[**1**]
    - *(int)*Number of Interpolation Regions[**1**]
    - *(int)*Number of Interpolation Points[**1**]

The repetition of much of this data in the index as well as the files allows the simple reading and extracting of the index without jumping back and forth in the binary file.

E. Material and Element Library Formats

For the convenience of the user, ALARA uses both a material and element library. The element library simply contains the natural isotopic breakdown of all the elements. The material library contains the elemental breakdown (using natural elemental compositions) of well-known materials. The usage of these libraries is described in more detail in Section 4.4 of the Users' Guide.[10]

E.1. Material Library

- Title indicating which material library this is
- Material info once for each material
  - *(char)*Material name [no white space allowed in name]
  - *(float)*Material density
  - *(int)*Number of elements in material
  - Element information once for each element
    - *(char)* Elemental symbol
    - *(float)* Weight fraction of this element
    - *(int)* Atomic number of element

E.2. Element Library

- Title indicating which element library this is
- Element info once for each element
  - elemental symbol
  - nominal elemental mass [g/mol]
  - atomic number
  - nominal elemental density [g/cm^3]
  - number of constituent isotopes
  - isotope information once for each naturally occurring isotope
    - mass number of isotope
    - atomic abundance of isotope

F. Flux File Formats

Every ALARA run requires a neutron flux file. Furthermore, ALARA has the ability to calculate a dose at a given point in space if a gamma flux importance file is given. The formats of these two files are similar. The first line is the title of the flux file. Both flux files are in a normal ASCII text format. Both files require one complete entry for each interval. A complete entry consists of an integer indicating which interval this flux data is for, followed by one flux value for each group. In the neutron flux file, the very first number (on the second line) must be an integer defining the number of groups.