Expanding Towards Excellence: Ironing Out DKR's Wrinkles

P.P.H. Wilson and D.L. Henderson

November 1995

UWFDM-995

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

Expanding Towards Excellence: Ironing Out DKR's Wrinkles

P.P.H. Wilson and D.L. Henderson

Fusion Technology Institute
University of Wisconsin
1500 Engineering Drive
Madison, WI 53706

http://fti.neep.wisc.edu

November 1995

UWFDM-995

Expanding Towards Excellence: Ironing Out DKR's Wrinkles

Paul P.H. Wilson
Douglass L. Henderson

Fusion Technology Institute
University of Wisconsin
1500 Engineering Drive
Madison, WI 53706

November 1995

UWFDM-995

Abstract

Although much criticism has been levelled on DKR and its descendants (DKRICF and DKR-PULSAR) in the past, many of the methods which it employs still provide the most accurate solution to fusion activation problems. By eliminating the one major drawback, premature truncation of loops in the decay/transmutation trees, codes such as DKR-PULSAR could reassert their claim amongst the premier activation codes, perhaps with little competition. This paper examines just how easily, accurately, and efficiently such modification can be made with little effect on the existing methods. By demonstrating an accurate method for modelling the loops as chains and developing mathematical methods to deal with these new systems, we are able to conclude that the much maligned linear chain approach of the DKR family of codes is really the key to its future success.

1. Introduction: Separating the Good from the Bad

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 reactors 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. Many years ago, development began for one such fusion activation code, DKR[1], which has continued to perform well since.

While much criticism has been made about DKR, it employs some basic principles which are valuable and important for activation codes in general. The main source of that criticism is the inability of DKR to deal with loops in the decay/transmutation tree of isotopes. On the other hand, while DKR was developed as a steady state activation code, in recent developments towards DKRICF and DKR-PULSAR, it has pioneered the ability to calculate exact solutions for pulsed systems [2, 3, 4, 5, 6]. In addition, DKR has led the development of other aspects of activation codes including the notion of tree/chain truncation tolerances and the use of exact analytical solutions to the governing equations. In fact, DKR and its progeny produce exceptional results for all but a minority of special cases. This paper will develop and demonstrate the methods which can be used to remove the points of contention while maintaining many of the benefits which have been developed over the long history of DKR and its progeny. Since the original DKR code is no longer in use, reference to the models developed in DKR will be referred to by its successor, DKRICF.

The linear chain method employed in DKRICF and DKR-PULSAR is both a benefit and a detriment. A decay/transmutation system that is modeled with linear chains has the strong advantage that closed form solutions are well published and easily implemented. On the other hand, when the real system contains loops, such as that which arises from an (n,p) reaction followed by a β^- decay,

$$A \xrightarrow{(n,p)} B \xrightarrow{\beta^-} A,$$

in most cases (see Section 2.), DKRICF is unable to handle it correctly. Early arguments claimed that these loops would be insignificant considering the types of engineering problems for which DKRICF was intended. However, with the increase in concern for trace quantities of both radioactive and toxic materials, these loops have the potential for taking on a more significant role. In Section 2. of this paper, a comparison of exact solutions to some approximate solutions will be given and analyzed.

Should loops prove to be important, it might suggest a necessity for drastic departures from DKRICF's current solution philosophy. This, however, is not necessarily true. The decay/transmutation tree is modeled by a system of coupled, linear, first-order ordinary differential equations with constant coefficients. Most undergraduate engineering students are taught to solve small equivalents of this problem in a very algorithmic fashion. It therefore seems possible to devise a computational algorithm to perform such exact solutions. Such methods are described in Section 3. of this paper.

2. Loops: Straightening Out a Messy Situation?

As mentioned above, DKRICF has been guilty of ignoring the effect of loops in most cases. Because the choice was made to use the very fast and accurate analytical solution to the Bateman equations,

a closed form solution for a general loop case was not readily available. One case was included, a "first-order" loop from an initial isotope. A "first-order" loop is one, such as the generic one described above, which only includes two isotopes. By this naming convention, a second-order loop would include three isotopes, etc. The solution of such a system at the top of any linear chain *was* simply implemented, but precluded the inclusion of isotopes which might be daughters of that second isotope (B above) and ignored the effect of the loop on other daughters of A. All in all, the mathematical methods of DKRICF resulted in a fast and accurate solution strategy for most cases, but made the analysis of, and accounting for, loops nearly impossible.

This action was justified by the claim that the significance of solutions to these loop problems would be minimal. While this may have been true at the beginning of DKRICF's development history, it is certainly no longer clear. The interest of safety analysts in trace quantities of both radioactive hazards and non-radioactive toxicity concerns has led to the necessity of solutions with high precision and accuracy. If loops are important, then DKRICF will fail to achieve these desired levels of accuracy and precision.

For this paper, the significance of loops was analyzed to determine the best alternative for handling them. First, the philosophy, mentioned above, of ignoring loops in almost all cases. Second, the concept of "loop-straightening" whereby the essentially infinite cycles of the loop are unraveled in the model and this infinite linear chain is truncated in a similar fashion as the rest of the problem (see Appendix A). Finally, there is the exact representation which models the loops by using the exact coupled differential equations. It is easy to see that the first alternative is just a special case of the second, truncating the infinite chain after a single iteration of the loop. (A single iteration does not account for the loops whatsoever, while two iterations has a single correction, and so on.) Thus, it is only necessary to compare the third, exact method with a number of cases of the second, straightening method, each case having a different number of iterations/corrections.

Certain assumptions of conservatism were made in performing this analysis, but care was taken to ensure that the results were still meaningful. Any conservative choices, in such an analysis, can easily force the results to one conclusion or the other, thus it is necessary to use the most accurate data and employ conservative assumptions to limit the search for offending cases. As a result, the analysis was limited to loops of first-order. It was assumed that since the relative production of an isotope decreases with each step down the linear chain, the contribution of a first-order loop would be the most significant. That is, if the contribution of a first-order loop is not important, then it can be assumed that the contribution of a second-order loop would be even less important. In addition, real possible loops were used with their correct neutron cross-section and decay data. The data libraries (based on USACT93[7]) were scanned to find all occurrences of $(n,p) \rightarrow \beta^-$ loops and the destruction and production rates were calculated and recorded. The one departure from this was to ignore transmutation of the radioactive isotope since it would add a negligible contribution to the total destruction of that isotope. It would have been possible to perform this analysis for an artificial domain of destruction and production rates, but if a significance was found in part of this domain, it would not be certain whether or not this part of the domain was within the realm of physically possible systems. Finally, a real flux profile was used so that destruction rates were not inflated by unreasonable fluxes in certain energy groups. A flux representative of an MFE system was used.

The results show the relative error between calculating the solution using straightened loops and calculating the solution exactly. Table 1 shows the maximum error in the calculation of isotope A, which occurs at 10^{10} s $\approx$ 317 y of continuous operation for 0 through 4 corrections. Figures 1

Table 1. Relative Errors when Using the Straightened Loop Method

(n,p) Reaction	# of corrections				
	0	1	2	3	4
$^{13}C(n,p)^{13}B$	4.05673e-05	8.22863e-10	1.09741e-14	1.11998e-16	1.11998e-16
$^{16}O(n,p)^{16}N$	0.000350779	6.15303e-08	7.19553e-12	5.61671e-16	1.12323e-16
$^{28}Si(n,p)^{28}Al$	0.0024263	2.94585e-06	2.38492e-09	1.44851e-12	1.02186e-15
$^{49}Ti(n,p)^{49}V$	0.000246933	3.04905e-08	2.50984e-12	0	1.20563e-16
$^{56}Fe(n,p)^{56}Mn$	0.00103427	5.35044e-07	1.84539e-10	4.77253e-14	0

through 5 show how this error varies with operation lifetimes from 10^4 s to 10^{10} s (2.8 h to 317 y).

It should be noted that errors less than 10^{-14} are approaching machine precision, and thus should be considered as 0. It is for this reason that the figures only show results for relative errors greater than 10^{-14}. Also, although a flux representative of an MFE system was used, the analysis is valid for both MFE and IFE systems. The IFE systems would have a higher flux (perhaps a slightly different spectrum) resulting in a shift to the left for all the plots.

It is clear that a solution using no corrections can experience significant error, as high as 0.2%, and that each correction reduces that error significantly. It is also clear that the relative error is a strongly increasing function of the operation lifetime, but the errors remained generally very small with only one or two corrections. These results do show that by successively adding corrective iterations, the error can be reduced to an acceptable level without performing the exact solution. In fact, it is rarely necessary to to use more than 3 corrections (4 iterations) to achieve an acceptable accuracy.

Furthermore, because these are relative errors, it indicates an error in the accuracy of the result and not in the precision. That is, whether the user is interested in quantities of 10^{23} or 10^6 or 10^{-6}, the *relative* error in that quantity will be the result from this analysis. It does not represent a "detection limit" of the process, but simply a limit on the accuracy of a result, the number of significant digits in the result. This accuracy is often more affected by uncertainties in input data and, thus, a relative error caused by this approximation of 10^{-7} is adequate.

In theory, it should be possible for the user to provide both a precision limit, which would determine how far down the chain to follow, and an accuracy limit, which would determine how many iterations of a particular loop to follow. Thus, a user could find solutions to any precision, say parts per billion, and have the accuracy determined separately. In practice, however, it is difficult to implement these two values separately, since it requires a pre-emptive search to determine whether or not a particular isotope may be involved in a loop further down the chain, a very costly operation.

3. Mathematical Method: Matching the Math to the Model

If loop straightening is a valid method of handling loops, then we can solve the entire problem as a set of linear chains, although some chains may have straightened loops in them. This suggests that a mathematical method exists that is a slight variation on the exact Bateman solution of the equations, as is currently implemented in DKRICF. On paper, for small problems, a scientist might simply use a Laplace transform technique, converting this system of ODE's to a system of algebraic

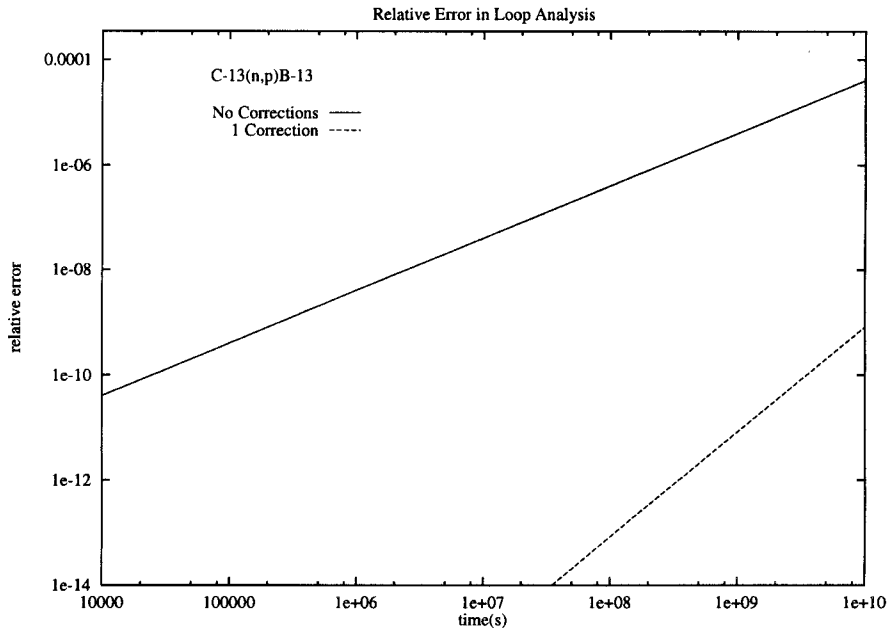

Figure 1. Relative error in calculation of ^{13}C as part of (n,p) loop.

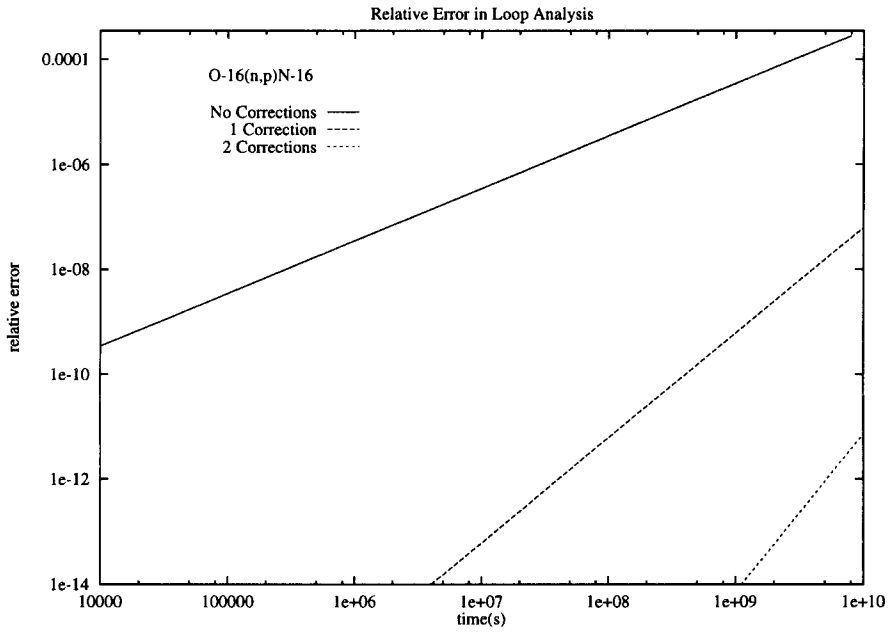

Figure 2. Relative error in calculation of ^{16}O as part of (n,p) loop.

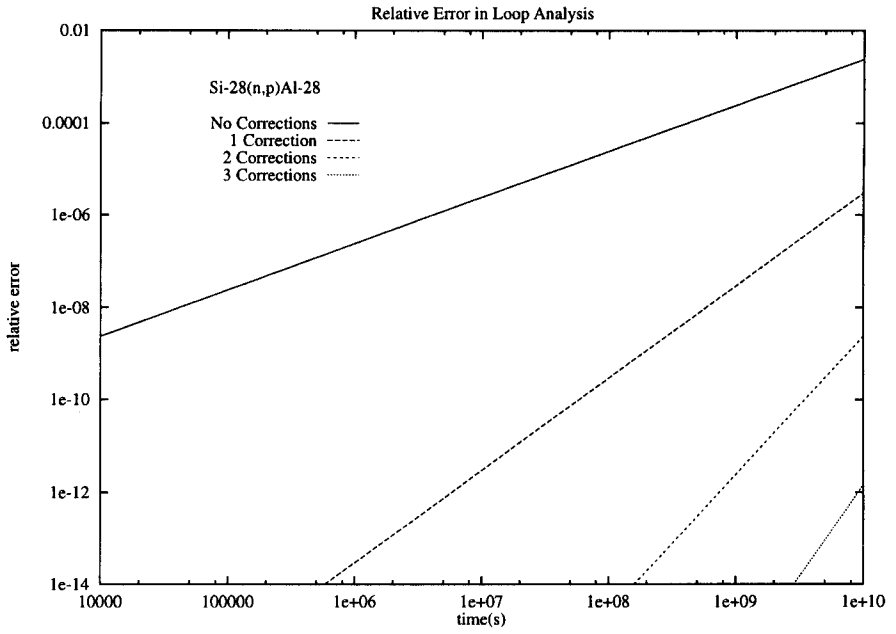

Figure 3. Relative error in calculation of ^{28}Si as part of (n,p) loop.

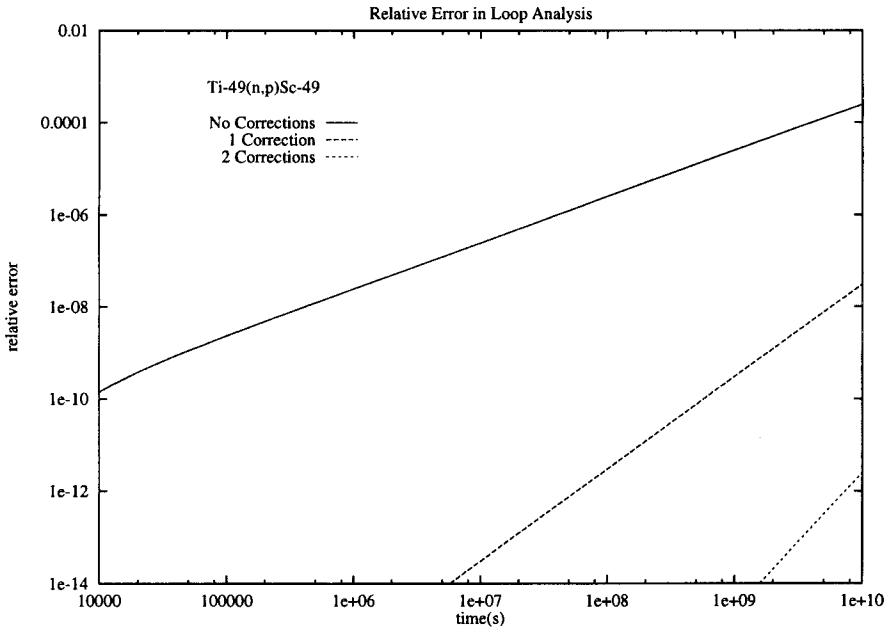

Figure 4. Relative error in calculation of ^{49}Ti as part of (n,p) loop.

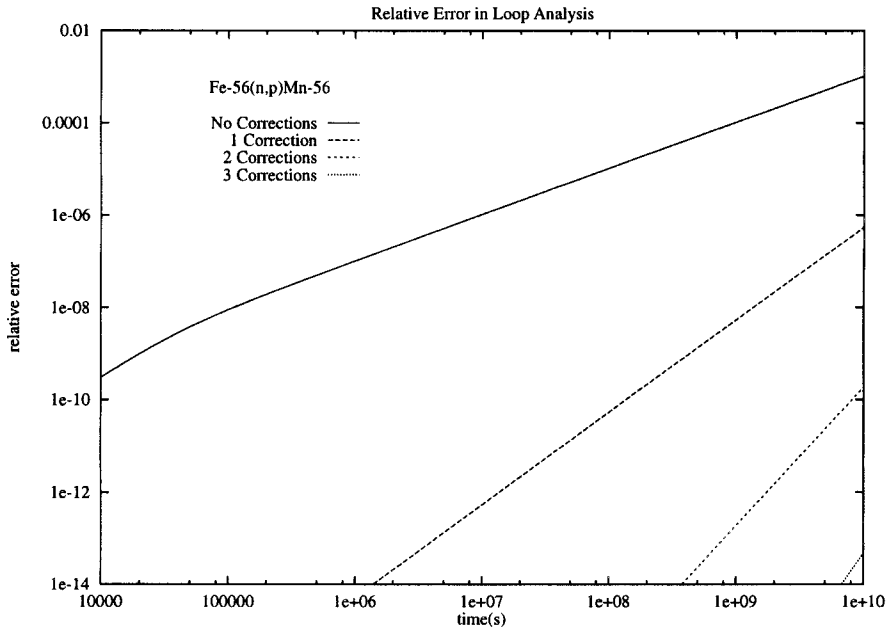

Figure 5. Relative error in calculation of ^{56}Fe as part of (n,p) loop.

equations, solving it algebraically, and then converting back to the solution. The results, in the Laplace transform space, are very simple and the conversion for any small arbitrary problem is straightforward. To extend this to large systems of equations, it is necessary to convert the method to a robust algorithm.

The first approach is to simply determine the steps used in the solution of a simple small problem and find a way to generalize them to a straightforward algorithm for a general problem. The key part of this effort is in determining the residues when performing a partial fractions separation of the solution in the Laplace transform space. In situations with no loops, the solution is trivial: the Bateman equations, but loops in the chain lead to multiplicities in the poles of the Laplace solution, and a much more complicated solution methodology. On the other hand, even for an arbitrary loop, the specific nature of the Laplace solution results in an elegant recursive method for determining the residues. Alternately, the solution can be determined as an expansion in time ($1/s$ in the Laplace domain) which also has a simple and elegant representation, and can be more appropriate for short times when the eigenvalues are very similar. Both methods will be presented here and start with the same basic equations.

Let us examine this problem in the Laplace domain. For a chain of isotopes with concentrations N_i, destruction rates d_i (which are $\lambda_i + \sigma_{(i,tot)}\phi$ where tot indicates the total cross section for all reactions), and production rates from the previous isotope of P_{i-1} (which λ_{i-1} or $\sigma_{(i-1,x)}\phi$ where x indicates reaction type x), the differential equation is simply:

$$\frac{dN_i}{dt} = -d_i N_i + P_{i-1} N_{i-1} \tag{3.1}$$

and in the Laplace domain:

$$s\tilde{N}_i = N_{i_o} - d_i \tilde{N}_i + P_{i-1} \tilde{N}_{i-1} \,. \tag{3.2}$$

Using this, the transformed concentration of each isotope in the chain can be found in terms of its predecessor as:

$$\tilde{N}_i = \frac{N_{i_0}}{s+d_i} + P_{i-1}\frac{\tilde{N}_{i-1}}{s+d_i}.h \tag{3.3}$$

If this predecessor is expanded in terms of its predecessor and so on, the transformed concentration of each isotope can be written as:

$$\begin{aligned}\tilde{N}_i = &\frac{N_{i_0}}{s+d_i} + \frac{N_{i-1_0}}{s+d_i}\frac{P_{i-1}}{s+d_{i-1}} + \frac{N_{i-2_0}}{s+d_i}\frac{P_{i-2}P_{i-1}}{(s+d_{i-2})(s+d_{i-1})} + \cdots \\ &+ \frac{N_{1_o}}{s+d_i}\prod_{j=1}^{i-1}\frac{P_j}{s+d_j} + \frac{N_{0_o}}{s+d_i}\prod_{j=0}^{i-1}\frac{P_j}{s+d_j}.\end{aligned} \tag{3.4}$$

This can be expressed as a contribution from each of the previous isotopes of the form:

$$\begin{aligned}\tilde{N}_i &= \sum_{j=0}^{i}\tilde{N}_{i,j} \\ &= \sum_{j=0}^{i} N_{j_o}\prod_{k=j}^{i-1}P_k\prod_{l=j}^{i}\frac{1}{s+d_l} \\ &= \sum_{j=0}^{i} N_{j_o}\tilde{F}_{i,j}(s)\prod_{k=j}^{i-1}P_k,\end{aligned} \tag{3.5}$$

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

$$\tilde{F}_{i,j}(s) = \prod_{l=j}^{i}\frac{1}{s+d_l}. \tag{3.6}$$

Once again, if there are no straightened loops in the chain, this is a trivial problem, and the solution becomes exactly that of the Bateman solution. However, the repeated poles caused by the straightening of a loop result in more complicated solutions.

3.1. Deriving Recursive Derivatives

One approach is to continue following the steps of a manual calculation. For repeated poles, one would use 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 power of 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, for the pole, $-d_k$, is calculated as

$$R = \lim_{s\to d_k}(s+d_k)\tilde{F}_{i,j}(s) \tag{3.7}$$

which becomes $Re^{-d_k t}$ 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:

$$\frac{R_m}{(s+d_k)^m} + \frac{R_{m-1}}{(s+d_k)^{m-1}} + \cdots + \frac{R_1}{(s+d_k)} \tag{3.8}$$

which becomes

$$e^{-d_k t}\left(R_m \frac{t^{m-1}}{(m-1)!} + R_{m-1}\frac{t^{m-2}}{(m-2)!} + \cdots + R_2\frac{t}{1!} + R_1\right) \tag{3.9}$$

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

$$R_n = \frac{1}{(m-n)!}\lim_{s\to d_k}\frac{d^{m-n}}{ds^{m-n}}\left[(s+d_k)^m \tilde{F}_{i,j}(s)\right]. \tag{3.10}$$

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

$$\tilde{G}^k_{i,j}(s) = (s+d_k)^m \tilde{F}_{i,j}(s) \tag{3.11}$$

at values of $s \neq -d_k$ for all i. By examining the successive derivatives of $G(s)$ it can be shown (see Appendix B.) that it can be recursively defined as:

$$\left[\tilde{G}^k_{i,j}(s)\right]^{(n)} = \sum_{j=1}^{n}(-1)^j\frac{(n-1)!}{(n-j)!}\left[\tilde{G}^k_{i,j}(s)\right]^{(n-j)}\sum_{i=1}^{I}\frac{1}{(s+d_i)^j} \tag{3.12}$$

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.

3.2. Expanding, Factoring, and Simplifying

Alternately, an expansion in $1/s$ may be desirable in some cases. There are problems in which the above method might introduce roundoff errors due to division by small 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 cancelling. While this seems like a monumental task to perform on an arbitrary problem, it is again quite simple to implement. If we start again with our function:

$$\tilde{F}_{i,j}(s) = \prod_{l=j}^{i}\frac{1}{s+d_l} \tag{3.13}$$

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

$$\begin{aligned}
\tilde{F}_{i,j}(s) &= \frac{1}{s^{i-j+1}}\prod_{l=j}^{i}\frac{1}{1+\frac{d_l}{s}} \\
&= \frac{1}{s^{i-j+1}}\prod_{l=j}^{i}\left(1 - \frac{d_l}{s} + \frac{d_l^2}{s^2} - \frac{d_l^3}{s^3} + \cdots\right) \\
&= \frac{1}{s^{i-j+1}}\left[1 - \frac{\sum_{l=j}^{i} d_l}{s} + \frac{\sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k}{s^2} - \frac{\sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \sum_{m=k}^{i} d_m}{s^3} + \cdots\right].
\end{aligned} \tag{3.14}$$

If $n = i - j + 1$, in the time domain, this becomes:

$$f(t) = t^{n-1}\left[1 - \frac{t}{n!}\sum_{l=j}^{i} d_l + \frac{t^2}{(n+1)!}\sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k - \frac{t^3}{(n+2)!}\sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \sum_{m=k}^{i} d_m + \cdots\right]. \tag{3.15}$$

It is clear that this solution will only be computationally viable when the product, $n \cdot \max\{d_i\} \cdot t$ is small, which is only guaranteed for arbitrary problems when there are small times, perhaps requiring scaling and squaring of the result for larger times. Other representations can be formed, each providing different insight into the method (see Appendix D.).

We will call this method the Laplace Expansion Method.

Both of these methods can be used to evaluate the solution to the differential equations which describe a linear chain with possible straightened loops. The results from Section 2. were generated using the Laplace Inversion Method and the C++ algorithm is included as an appendix (Appendix C.).

4. Future Developments: Continuing DKR's Family Tree

This report conclusively shows that straightened loop approximations are certainly capable of accurately modeling the loops which may occur in a decay/transmutation tree while maintaining the linear chain form of the data. Further, we have demonstrated that mathematical methods exist to solve these special linear chains either with an analytical approach or with an accurate expansion technique.

With these issues resolved, effort can be returned to the final development of an activation code to include these and other developments. Currently under development at the University of Wisconsin, a new code named ALARA [Analytic and Laplace Adaptive Radioactivity Analysis] will borrow the most successful elements of its parent code, DKRICF, and incorporate these and other refinements. At the writing of this report, ALARA was in the final stages of testing for robustness in a variety of problems.

Acknowledgement

We acknowledge many hours of fruitful discussion and debate with the late Prof. Emeritus Charles Maynard of the Nuclear Engineering and Engineering Physics Department of the University of Wisconsin-Madison. Partial support for this work was provided by the Fusion Technology Institute of the University of Wisconsin-Madison under DOE Grant DE-AS08-88DP10754 and the University of Wisconsin-Madison Reactor Safety Analysis Group.

References

[1] 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.

[2] 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.

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

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

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

[6] P. Wilson, J. Sisolak, and D.L. Henderson, "Novel Approach to the General Solution of Pulsed History Activation Problems," Fus. Techn. Vol. 26, p. 1092 (November 1994).

[7] F.M. Mann and D.E. Lessor, "REAC*3 Nuclear Data Libraries," Proceedings of an International Conference on Nuclear Data entitled: Nuclear Data for Science and Technology, held at the Forschungszentrum Juelich, Fed. Rep. of Germany, 13-17 May 1991.

Appendices

A. A How-To Guide: Straightening Decay Schemes

The concept of straightening loops is quite simple. All reactions with an isotope, either neutron interactions or radioactive decays, are followed as independent branches in a tree. If we have a decay/transmutation scheme such as in Figure A.1, we can see loops such as $B \rightarrow E \rightarrow B$ as well as cross-linked branches, such as the link $H \rightarrow B$. When this scheme is straightened to produce a completely un-linked, loopless scheme (Figure A.2), "psuedo-unique" isotopes are introduced, such as B_2, B_3, B_4, B_5. After the complete solution for the scheme has been achieved, the production of each of these psuedo-unique isotopes is summed with the production of B_1 to find the total production of isotope B.

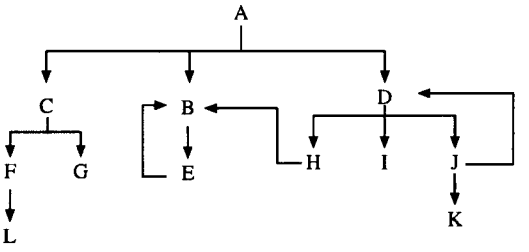

Figure A.1. Sample reaction tree (explicit loop representation).

Performing this loop and link removal in a code is actually a very straightforward and natural operation. As the cross-section and decay data reveal reactions, each reaction is followed as if an entirely new isotope is being added to the tree/chain. There is no need to traverse the tree, search for a previous occurrence and perform the linking, as some codes have done. Each branch is then truncated using the same rules as the rest of the tree, resulting in a uniform precision for the results for every isotope in the tree. Once the calculation is complete, the tree/chains must be traversed only once, summing the contributions from each pseudo-unique isotope. When this is combined with the mathematical methods outlined in this report, the result is an accurate and efficient method to solve decay/transmutation trees.

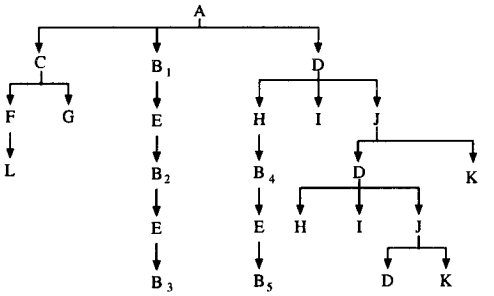

Figure A.2. Straightened loops representation.

B. Derivation of Recursive Derivative Definition

$$G(s) = \prod_{i=1}^{N} \frac{1}{s + d_i} \tag{B.16}$$

$$\begin{aligned} G'(s) &= \sum_{j=1}^{N} \frac{-1}{s + d_j} \prod_{i=1}^{N} \frac{1}{s + d_i} \\ &= -G(s) \sum_{j=1}^{N} \frac{1}{s + d_j} \end{aligned} \tag{B.17}$$

$$G''(s) = -G'(s)\sum_{j=1}^{N}\frac{1}{s+d_j} + G(s)\sum_{j=1}^{N}(s+d_j)^{-2} \tag{B.18}$$

$$\begin{aligned} G'''(s) &= -G''(s)\sum_{j=1}^{N}\frac{1}{s+d_j} + G'(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad + G'(s)\sum_{j=1}^{N}(s+d_j)^{-2} - 2G(s)\sum_{j=1}^{N}(s+d_j)^{-3} \\ &= -G''(s)\sum_{j=1}^{N}\frac{1}{s+d_j} + 2G'(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad - 2G(s)\sum_{j=1}^{N}(s+d_j)^{-3} \end{aligned} \tag{B.19}$$

$$\begin{aligned} G''''(s) &= -G'''(s)\sum_{j=1}^{N}\frac{1}{s+d_j} + G''(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad + 2G''(s)\sum_{j=1}^{N}(s+d_j)^{-2} - 4G'(s)\sum_{j=1}^{N}(s+d_j)^{-3} \\ &\quad - 2G'(s)\sum_{j=1}^{N}(s+d_j)^{-3} + 6G(s)\sum_{j=1}^{N}(s+d_j)^{-4} \\ &= -G'''(s)\sum_{j=1}^{N}\frac{1}{s+d_j} + 3G''(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad - 6G'(s)\sum_{j=1}^{N}(s+d_j)^{-3} + 6G(s)\sum_{j=1}^{N}(s+d_j)^{-4} \end{aligned} \tag{B.20}$$

Thus, for n=4,

$$\begin{aligned} G^{(n)}(s) &= -\frac{(n-1)!}{(n-1)!}G^{(n-1)}(s)\sum_{j=1}^{N}\frac{1}{s+d_j} \\ &\quad + \frac{(n-1)!}{(n-2)!}G^{(n-2)}(s)\sum_{j=1}^{N}(s+d_j)^{-2} \\ &\quad - \frac{(n-1)!}{(n-3)!}G^{(n-3)}(s)\sum_{j=1}^{N}(s+d_j)^{-3} \\ &\quad + \frac{(n-1)!}{(n-4)!}G^{(n-4)}(s)\sum_{j=1}^{N}(s+d_j)^{-4} \\ &= \sum_{i=1} n(-1)^i\frac{(n-1)!}{(n-i)!}G^{(n-i)}(s)\sum_{j=1}^{N}(s+d_j)^{-i} \end{aligned} \tag{B.21}$$

B.1. Induction Proof

$$G^{(n)}(s) = \sum_{i=1}^{n}(-1)^i \frac{(n-1)!}{(n-i)!} G^{(n-i)}(s) \sum_{j=1} N(s+d_j)^{-i} \tag{B.22}$$

given

$$G^{(0)}(s) = G(s) = \prod_{j=1}^{N}(s+d_j)^{-1} \tag{B.23}$$

First, we solve for n=1:

$$\begin{aligned} G'(s) &= (-1)\frac{0!}{0!} G(s) \sum_{j=1}^{N}(s+d_j)^{-1} \\ &= -G(s) \sum_{j=1}^{N}(s+d_j)^{-1} \end{aligned} \tag{B.24}$$

which matches Equation B.17.

Now, we solve for n=2:

$$\begin{aligned} G''(s) &= (-1)\frac{1!}{1!} G'(s) \sum_{j=1}^{N}(s+d_j)^{-1} + \frac{1!}{0!} G(s) \sum_{j=1}^{N}(s+d_j)^{-2} \\ &= -G'(s) \sum_{j=1}^{N}(s+d_j)^{-1} + G(s) \sum_{j=1}^{N}(s+d_j)^{-2} \end{aligned} \tag{B.25}$$

which matches Equation B.18.

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

$$G^{(k+1)}(s) = \sum_{i=1}^{k}(-1)^i \frac{(k-1)!}{(k-i)!} \left[G^{(k-i+1)} \sum_{j=1}^{N}(s+d_j)^{-i} - iG^{(k-i)} \sum_{j=1}^{N}(s+d_j)^{-(i+1)} \right] \tag{B.26}$$

letting, $l = k+1$:

$$\begin{aligned} G^{(l)}(s) = & \sum_{i=1}^{l-1}(-1)^i \frac{(l-2)!}{(l-i-1)!} G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \\ & - \sum_{i=1}^{l-1}(-1)^i \frac{(l-2)!}{(l-i-1)!} iG^{(l-i-1)} \sum_{j=1}^{N}(s+d_j)^{-(i+1)} \end{aligned} \tag{B.27}$$

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

$$\begin{aligned} G^{(l)}(s) = & \sum_{i=1}^{l-1}(-1)^i \frac{(l-2)!}{(l-i-1)!} G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \\ & + \sum_{m=2}^{l}(-1)^m \frac{(l-2)!}{(l-m)!}(m-1) G^{(l-m)} \sum_{j=1}^{N}(s+d_j)^{-m} \end{aligned} \tag{B.28}$$

and recombining the sums:

$$\begin{aligned} G^{(l)}(s) = & -\frac{(l-2)!}{(l-2)!} G^{(l-1)}(s) \sum_{j=1}^{N}(s+d_j) \\ & + \sum_{i=2}^{l-1}(-1)^i \left[\frac{(l-2)!}{(l-i-1)!} + (i-1)\frac{(l-2)!}{(l-i)!} \right] G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \\ & + (-1)^l (l-2)!(l-1) G(s) \sum_{j=1}^{N}(s+d_j)^{-l} \end{aligned} \tag{B.29}$$

$$\begin{aligned} = & -G^{(l-1)}(s) \sum_{j=1}^{N}(s+d_j)^{-1} \\ & + \sum_{i=2}^{l-1}(-1)^i \left[(l-i)\frac{(l-2)!}{(l-i-1)!(l-i)} + (i-1)\frac{(l-2)!}{(l-i)!} \right] G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \\ & + (-1)^l (l-1)! G(s) \sum_{j=1}^{N}(s+d_j)^{-l} \end{aligned} \tag{B.30}$$

$$\begin{aligned} = & -G^{(l-1)}(s) \sum_{j=1}^{N}(s+d_j)^{-1} \\ & + \sum_{i=2}^{l-1}(-1)^i \frac{(l-1)!}{(l-i)!} G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \\ & + (-1)^l (l-1)! G(s) \sum_{j=1}^{N}(s+d_j)^{-l} \end{aligned} \tag{B.31}$$

$$= \sum_{i=1}^{l}(-1)^i \frac{(l-1)!}{(l-i)!} G^{(l-i)} \sum_{j=1}^{N}(s+d_j)^{-i} \tag{B.32}$$

QED.

C. C++ Algorithm for Laplace Inversion Method

```
/* this module stores matrices is two dimensional arrays */

/* function to copy matrix b to matrix a */
```

```
void copy(double **a,double **b, int M)
{
  for (int i=0;i<M;i++)
    for (int j=0;j<M;j++)
      a[i][j] = b[i][j];
}

/* c= a*b */
void mult(double **c,double **a, double **b, int M)
{
  double **temp = new double*[M];

  for (int i=0;i<M;i++)
    {
      temp[i] = new double[M];
      for (int j=0;j<M;j++)
        {
          temp[i][j] = 0;
          for (int k=0;k<M;k++)
            temp[i][j] += a[i][k] * b[k][j];
        }
    }

  copy(c,temp,M);
  for (i=0;i<M;i++)
    delete temp[i];
  delete temp;
}

//result = a^N
/* Reference: E.S. Lee "Computer Engineering: Computer Algorithms,
              Data Structures, and Languages," Prepared Notes,
              University of Toronto, 1989. */
void matPow_a(double **result, double **a, long int N, int M)
{
  long int n = N;
  int m,l,firstmult=0;
  double **ans,**pow, **temp;
  ans = new double*[M];
  pow = new double*[M];

  /* initialize arrays */
  for (m=0;m<M;m++)
    {
      ans[m] = new double[M];
```

```
      pow[m] = new double[M];
      for (l=0;l<M;l++)
        {
          pow[m][l] = a[m][l];
          ans[m][m] = 0;
        }
      ans[m][m] = 1;
    }

  while (n!=0)
    {
      if (n%2 == 1)
        if (firstmult)
          mult(ans,ans,pow,M);
        else
          {
            copy(ans,pow,M);
            firstmult=1;
          }
      if (n==1)
        break;

      mult(pow,pow,pow,M);

      n = n/2;
    }

  copy(result,ans,M);

  for (m=0;m<M;m++)
    {
      delete pow[m];
      delete ans[m];
    }
  delete pow;
  delete ans;
}


/* compute i! */
double fact(int i)
{
  /* catch possibility of requesting 0! */
  if (i==0)
    return 1;
```

```
  /* initialize result */
  double result=i--;

  /* compute results */
  for (;i>1;i--)
    result *= i;

  /* return result */
  return result;
}


/* recursive routine to find derivative */
double dGn(double s, double s1, double s2, int n, int m1, int m2)
{
  double result=0;

  /* if 0th derivative */
  if (n == 0)
    /* return product of poles evaluated as s */
    return  1/(pow((s1-s),m1)*pow((s2-s),m2));
  /* otherwise */
  else
    /* for each term between this and 1 */
    for (int i=n;i>0;i--)
      /* the derivative is found by summing a linear combination of
       * all lower derivatives */
      result += -2*(i%2 -.5) * (fact(n-1)/fact(n-i)) * dGn(s,s1,s2,n-i,m1,m2)
                * (m1*pow(1/(s1-s),i)+m2*pow(1/(s2-s),i));

  return result;

}

/* routine to find inverse of 3 poles, d, with multiplicites, m */
double invert(double t, double* d, int* m, double* coeff)
{

  double result=0, result2=0, result3=0;
  double residue;
  int poleNum,termNum;

  /* for each pole */
  for (poleNum=0;poleNum<3;poleNum++)
```

```
    {
      /* initialize the coefficient */
      coeff[poleNum] = 0;

        /* if the multiplicity is greater than 0 */
      if (m[poleNum] != 0)
        /* for each term in the partial fractions */
        for (termNum=m[poleNum];termNum>0;termNum--)
          {
            /* the coefficient is incremented by the correct power of t^n/n!
             * times the derivative of the Laplace space function */
            coeff[poleNum] += pow(t,termNum-1)/fact(termNum-1) *
              dGn(d[poleNum],d[(poleNum+1)%3],d[(poleNum+2)%3],
                  m[poleNum]-termNum,m[(poleNum+1)%3],m[(poleNum+2)%3])
                / fact(m[poleNum]-termNum);
          }
    }

}
```

D. Other Forms of $1/s$ Expansion

The $1/s$ expansion from Section 3.2. can take on many slightly different forms providing different methods for determining the coefficients. Whether in the Laplace transform domain or the time domain, there is a necessity to calculate coefficients of the form:

$$\{c_i\} = \left\{ \sum_{j=1}^{N} d_j \ , \ \sum_{j=1}^{N} d_j \sum_{k=j}^{N} d_k \ , \ \sum_{j=1}^{N} d_j \sum_{k=j}^{N} d_k \sum_{l=k}^{N} d_l \ , \ \ldots \right\}. \tag{D.33}$$

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

$$\{c_i\} = \left\{ d_1 + d_2 \ , \ d_1(d_1 + d_2) + d_2^2 \ , \ d_1 \left[d_1(d_1 + d_2) + d_2^2 \right] + d_2^3 \ , \ldots \right\} \tag{D.34}$$

or

$$\begin{aligned} \{c_i\} = \Big\{ & d_1 + d_2 + d_3, \ d_1(d_1 + d_2 + d_3) + d_2(d_2 + d_3) + d_3^2, \\ & d_1 \left[d_1(d_1 + d_2 + d_3) + d_2(d_2 + d_3) + d_3^2 \right] \\ & + \ d_2 \left[d_2(d_2 + d_3) + d_3^2 \right] + d_3^3 \ , \ldots \Big\} \end{aligned} \tag{D.35}$$

This shows the following pattern, assuming $\{c_{0,j}\} = \{d_j\}$:

$$c_i = \sum_{j=1}^{N} d_j (c_{i-1,j}) \, . \tag{D.36}$$

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

$$M = \begin{bmatrix} d_1 & 0 & 0 & \dots & 0 \\ d_1 & d_2 & 0 & \dots & 0 \\ d_1 & d_2 & d_3 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ d_1 & d_2 & d_3 & \dots & d_N \end{bmatrix}, \tag{D.37}$$

it is clear that $\{c_{0,j}\} = \{m_{N,j}\} = [M]_{N,j}$ and that $\{c_{i,j}\} = [M^i]_{N,j}$. Therefore,

$$c_i = \sum_{j=1}^{N} c_{i,j} = \left[M^i \cdot \vec{1}\right]_N. \tag{D.38}$$

Since the direct calculation of

$$\sum_{j_1=1}^{N} d_{j_1} \sum_{j_2=j_1}^{N} d_{j_2} \sum_{j_3=j_2}^{N} d_{j_3} \cdots \sum_{j_{n-1}=j_{n-2}}^{N} d_{j_{n-1}} \sum_{j_n=j_{n-1}}^{N} d_{j_n} = \prod_{l=n}^{1} \sum_{j_l=j_{l-1}}^{N} d_{j_l} \tag{D.39}$$

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 3.15:

$$\begin{aligned} f(t) = t^{n-1} \Bigg[1 - \frac{t}{n!} \sum_{l=j}^{i} d_l + \frac{t^2}{(n+1)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \\ - \frac{t^3}{(n+2)!} \sum_{l=j}^{i} d_l \sum_{k=l}^{i} d_k \sum_{m=k}^{i} d_m + \cdots \Bigg]. \end{aligned} \tag{3.15}$$