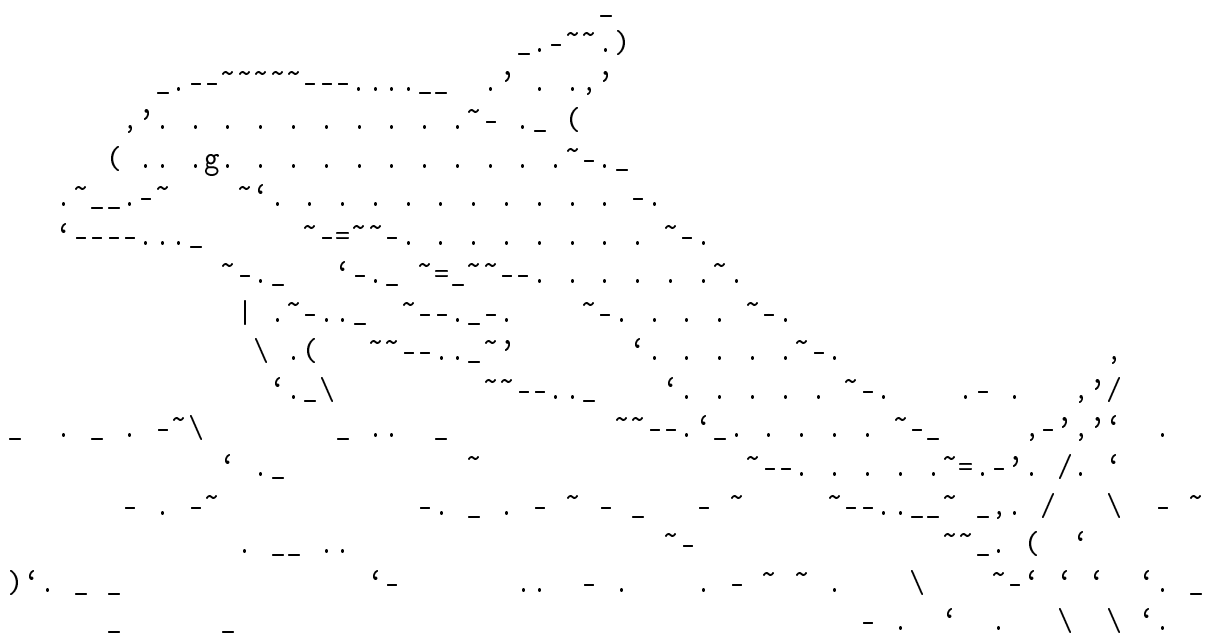
Nuclear Data for Fuel Cycle Optimization Studies

Nathan Gibson
Nick Horelik

Friday, December 16, 2011



1 Introduction

In the creation of a fast running, multi-cycle PWR reload optimization tool, immediately accessible nuclear data is of paramount importance. For core calculations, assembly-averaged homogenized cross sections are desired, and these are generally calculated from a lattice physics tool. Codes such as CASMO and DRAGON use the method of characteristics and the collision probability method to compute neutronics behavior in a reactor assembly; recently, Monte Carlo has been gaining added attention for this process with SERPENT being the most prominent Monte Carlo lattice tool. However, the deterministic tools still are the most prevalent, as even with improving computers, it is a great computational expense to yield all the desired nuclear data. For a given assembly, not only are assembly-averaged cross sections desired, but parameters such as moderator temperature coefficient and Xenon worth are desired. In designing and optimizing a full reactor core, many assembly types with varying geometries, fuel and poison materials, operating conditions, and burn-up histories must be represented.

Using a reactor database leads to some complications. First, the database must have enough information to adequately represent the range of conditions used in a core calculation. This leads to a very large and complicated database from which to pull nuclear data. With such a large database, data lookups can be slow and complicated. Furthermore, the reactor database coupling most lattice physics codes with their respective core solvers is in a proprietary format that would not be easily extendable to the fast PWR reload optimization tool of interest in this project.

For the desired optimization tool, the complications of a lattice physics calculation for each assembly and its operating history is simply not feasible. Thus, a subset of possible conditions were precomputed using CASMO to populate a small and simple reactor database. Various methods of interpolating and curve fitting on this database were considered and discussed in this report.

1.1 Use with Other Modules

In the development of the nuclear data module, in addition to maximizing speed, an important objective was to yield an extremely simple interaction with other modules. Assembly type and conditions are input either by a user or by an automated optimization or depletion module. The outputs, assembly-averaged homogenized two-group cross sections and other reactor physics parameters, are then used by the 2-D neutronics module.

Each cross section type is wrapped in a function that returns the value. Also, for additional ease of use, a single function wraps each of these separate functions so only

one call to the module is needed for each set of cross sections. For instance, in a Python implementation, the call would appear as:

```
data = get_2g_params( B , ENR , TMO , TFU , BOR , WABA , IFBA , GD )
```

In this function call, `B` is the burnup in MWD/kgU; `ENR` is the U-235 enrichment in weight percent; `TMO` and `TFU` are the moderator and fuel temperatures in Kelvin; `BOR` is the soluble boron concentration in ppm; and `WABA`, `IFBA`, and `GD` are identifiers for the burnable poison type and configuration.

In a core depletion calculation, this routine may need to be called for each assembly, as each assembly depletes at its own rate. For a standard PWR with quarter core symmetry, this is nearly 50 calls to the data module for each state point. Thus, it is important that such a routine be very efficient so as to not be the bottleneck in a fast optimization tool.

1.2 Scope

The scope of the study and the resulting reactor database was necessarily limited due to time constraints and the initial development work. However, the groundwork has been put in place such that the database could be extended easily, as described in Section 5.2.

The reactor database developed in this implementation represents the Westinghouse STD 17x17 PWR fuel assembly. Assembly enrichment can be varied from 3-5%. Burnup data is supported up to 70 MWD/kgU. Burnable poisons are supported in a limited fashion; one configuration for each of IFBA, WABA, and gadolinium are considered.

For all cases, soluble boron content is taken to be 900 ppm; moderator temperature is 560K; coolant temperature is 580K; and fuel temperature is 900K. These parameters should be allowed to vary, but this is left to future work. Furthermore, history effects related to changes in operating conditions are not modeled in this module.

2 Methodology

2.1 Interpolation

Interpolation methods guarantee a high degree of accuracy near known data points, but often suffer in light of the large time and memory requirements of a fine mesh of data

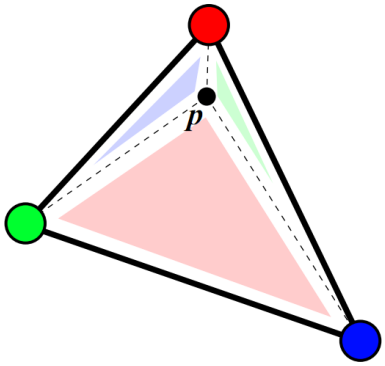


Figure 1: Schematic example of linear barycentric interpolation in 2D space

points needed for accurate predictions between points. To help mitigate these problems, numerical interpolation methods in general try to spend as much time as possible precomputing an interpolator function that can later be used to quickly calculate values at any point in the space.

While there are many types of interpolation schemes, only two simple types were explored for the nuclear data module, briefly described here.

2.1.1 Linear Interpolation (Barycentric Lagrange Interpolation)

The interpolator p can be written in the Lagrange form for data f, with weights l:

$$p(x) = \sum_{j=0}^{n} f_j l_j(x) \tag{1}$$

$$l_j = \frac{\prod_{k=0,k\neq j}^{n}(x - x_k)}{\prod_{k=0,k\neq j}^{n}(x_j - x_k)} \tag{2}$$

It is easy to understand and visualize linear interpolation in 2 dimensions, but it can also apply to N-dimensional space. Conceptually, given any N-dimensional point in the space you can estimate the value with a linear barycentric interpolation within a hypervolume created by N+1 data points. Here, as shown schematically in Figure 1, the value at the point is a linear combination of the values at the vertices of the hull weighted by the ratios of the volumes opposite the vertex.

Typically, methods that implement this type of linear interpolation pre-compute a Delaunay triangulation on the input data to obtain the grid of convex hypervolumes within which to carry out the previously described calculation. Then the problem is

rewritten in the barycentric coordinate system to manipulate the Lagrange form of the interpolator so that the weights can be determined for each hypervolume in the grid.

The full details of the math are not discussed here, but more on this topic can be found in [1], which shows that with this method this interpolator can be updated and evaluated $\mathcal{O}(N)$ in time, where N is the number of data points.

2.1.2 Radial Basis Functions

As in linear interpolation, an interpolator function is precomputed for the dataset. This time the interpolator s for the dataset x is constructed as a combination of radially symmetric basis functions φ that act on the norm $r = ||x - x_i||$, with weights λ:

$$s(x) = \sum_{i=1}^{N} \lambda_i \varphi(||x - x_i||). \tag{3}$$

This provides a much smoother fit between data points. For this module, the triharmonic spline was chosen as basis function.

$$\varphi(r) = r^3$$

A more detailed discussion regarding radial basis function interpolation can be found in [2].

2.2 Curve Fitting

Fitting functions to entries in the reactor database provides a very fast alternative to interpolation but comes with the sacrifice of some accuracy.

The general process of curve fitting is an optimization process. One must select the optimal shape function to represent the data and pick its coefficients to minimize errors. Most commonly, this is done through a least squares process. In least squares fitting, a norm is defined as the sum of the squares of the differences between the data points and the evaluation of the shape function at the same points. When this norm is minimized,

[1] Berrut, Jean-Paul and Lloyd N. Trefethen. Barycentric Lagrange Interpolation. SIAM Review Vol. 46, No. 3, pp. 501-517.

[2] Baxter, BJC. The Interpolation Theory of Radial Basis Functions. Dissertation, Trinity College, Cambridge, UK.

the best fit to the data is obtained. The most common numerical scheme to solve this problem is to use QR factorization on an over-constrained linear system. However, this study opted to use the extablished built-in implementations in MATLAB and Python's SciPy rather than developing a new curve fitting tool.

For much of the initial scoping described in this report, optimal shape functions for the curves were determined by the so-called "eyeball norm" – that is, decisions of what functions to use were made by hand by qualitatively determining if the function appeared to follow the data. However, in the final implementation, this process was somewhat formalized by seeking to minimize a norm. However, it should be noted that additional logic is needed than simply minimizing a norm – for instance, a higher degree polynomial will always yield a more optimal norm than a lower degree polynomial, but it may result in non-physical behavior between data points and comes at slightly higher computational cost.

2.2.1 Directly Fitting Nuclear Data

The most intuitive way to fit curves to nuclear data is to fit each desired output as a function of the inputs. For instance, one could fit Σ_{a1}, Σ_{a2}, $\nu\Sigma_{f1}$, and $\nu\Sigma_{f2}$ as a function of burnup and enrichment.

The most general method of fitting a curve to data as a function of multiple variables is to set up a shape function with several independent coefficients and minimize the resulting norm by adjusting the coefficients. A simple linear model could assume $\nu\Sigma_{f2}$ is a linear function of both burnup B and enrichment R. The test curve would then be:

$$\nu\Sigma_{f2} = c_1 B + c_2 R + c_3. \tag{4}$$

More complicated functions could be attempted, potentially including nonlinear and complicated terms. For instance, an unlikely but potential test curve could be:

$$\nu\Sigma_{f2} = c_1 B^2 + c_2 B + c_3\sqrt{R} + c_4\frac{\ln B}{R} + c_5. \tag{5}$$

It is easy to see that this process could become unwieldy very quickly. It not only becomes difficult to determine coefficients, but it also is hard to determine the proper shape function. Thus, a somewhat different approach was taken in this study.

First, one independent variable is chosen to vary, and all other independent variables are held fixed. A one-dimensional fit can then be determined and easily evaluated (in the "eyeball norm" or a more quantitative norm). When an appropriate shape function

is determined, it can be then be applied to cases with different values of the initially fixed variables. The coefficients of the fits can then be fit themselves to yield a multi-dimensional fit.

To clarify this process, consider an example. Again, assume $\nu\Sigma_{f2}$ is a function of burnup B and enrichment R. Fits of $\nu\Sigma_{f2}$ versus burnup can be obtained for each value of E. Consider for simplicity a linear fit.

$$\nu\Sigma_{f2}(R_i) = a_1(R_i)B + a_2(R_i) \tag{6}$$

Then, the coefficients a_1 and a_2 can be fit to enrichment, potentially with different fits. Take, for instance, a linear fit of a_1 and a quadratic fit of a_2:

$$\begin{aligned} a_1 &= b_1R + b_2 \\ a_2 &= c_1R^2 + c_2R + c_3. \end{aligned} \tag{7}$$

The result is then a multi-dimensional fit for $\nu\Sigma_{f2}$:

$$\nu\Sigma_{f2}(B,R) = (b_1R + b_2)B + (c_1R^2 + c_2R + c_3). \tag{8}$$

2.2.2 Grouped Parameter Fitting – Linear Reactivity Model

It is recognized that grouping quantities together in a curve fit can yield more desirable shapes. This is a practice commonly seen in thermal hydraulics correlations, which often use non-dimensional combinations of quantities for best results. Such a concept was applied to cross sections in this study using the Linear Reactivity Model (LRM).

The Linear Reactivity Model predicts that the reactivity of a given fuel bundle with no burnable poison decreases linearly with burnup. Furthermore, it predicts that varying enrichment in a given bundle type does not change the slope of the reactivity trajectory; rather, it simply shifts the trajectory up or down.

For these curve fits, the LRM was extended. The k-eigenvalue was split into parts k_1 and k_2, as defined by CASMO:

$$k = \underbrace{\frac{\nu\Sigma_{f1}}{\Sigma_R + \Sigma_{a1}}}_{k_1} + \underbrace{\frac{\nu\Sigma_{f2}}{\Sigma_{a2}}\frac{\Sigma_{R1}}{\Sigma_{R1} + \Sigma_{a1}}}_{k_2} \tag{9}$$

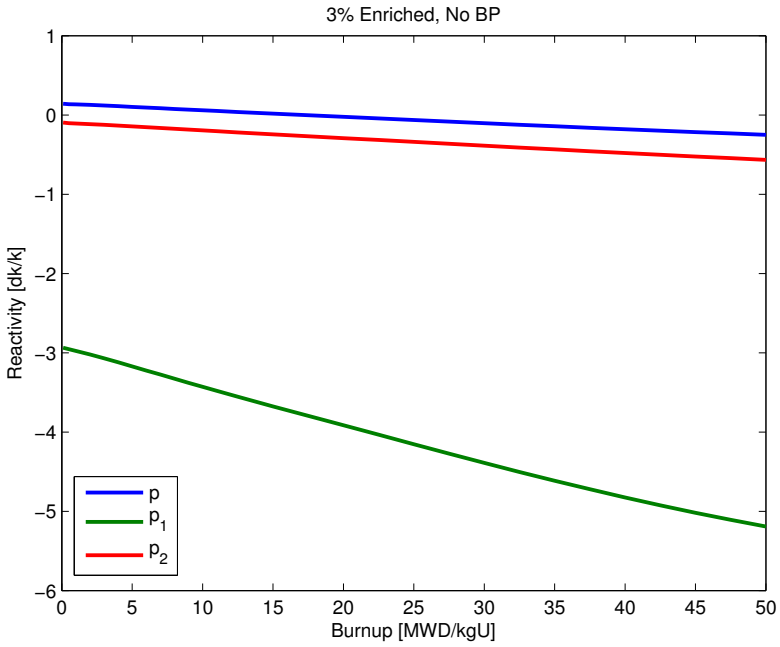


Figure 2: Reactivity and pseudo-reactivity trajectories for a test case.

With these group contributions to the eigenvalue, pseudo-reactivities for each group were defined:

$$\begin{aligned} \rho_1 &= \frac{k_1 - 1}{k_1} \\ \rho_2 &= \frac{k_2 - 1}{k_2} \end{aligned} \tag{10}$$

Both the reactivity and the pseudo-reactivities were plotted for a test case, shown in Figure 2. As expected, the true reactivity showed a linear trajectory. Somewhat surprisingly, both the pseudo-reactivities also exhibited such behavior. Furthermore, the slopes were essentially constant with varying enrichments and the intercept values varied linearly with enrichment (Figure 3). For better agreement, quadratic fits were used for the slopes and intercepts, but the added terms were small enough to not alter the underlying LRM assumption.

A test case with IFBA was also considered. It was found that the difference in reactivity and pseudo-reactivities between the IFBA and bare cases followed very closely to an exponential. This is a very reasonable finding; to first order, because IFBA is a very thin layer of absorbing material, little to no self-shielding effects are expected and rate of absorption in poison is expected to be proportional to the amount of poison present.

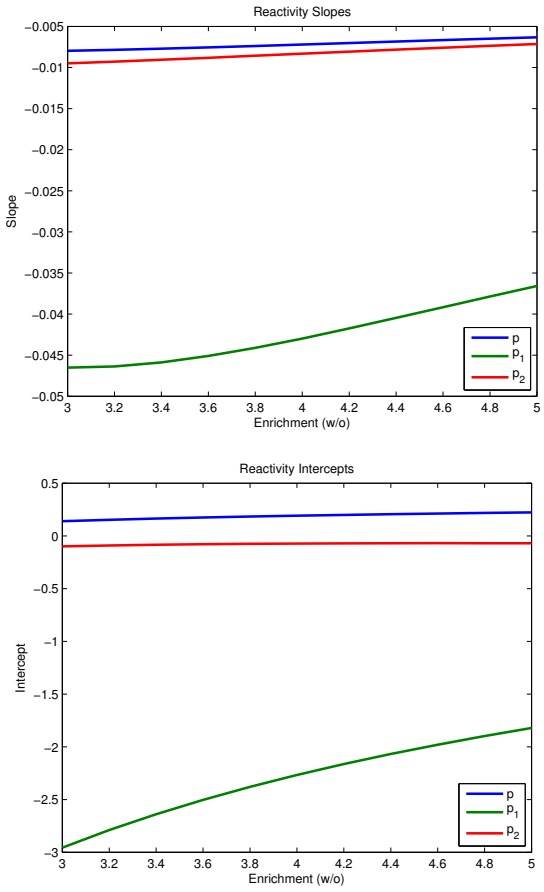


Figure 3: Slopes (left) and intercepts (right) for reactivity fits.

With fits for the pseudo-reactivities, one can consider the component cross sections. Solving for the absorption cross sections:

$$\begin{aligned} k_1 &= \frac{1}{1-\rho_1} \\ k_2 &= \frac{1}{1-\rho_2} \end{aligned} \tag{11}$$

$$\begin{aligned} \Sigma_{a1} &= \frac{\nu\Sigma_{f1}}{k_1} - \Sigma_{R1} \\ \Sigma_{a2} &= \frac{\nu\Sigma_{f2}}{k_2}\frac{\Sigma_{R1}}{\Sigma_{R1}+\Sigma_{a1}} \end{aligned} \tag{12}$$

The resulting system of Equation 12 contains only two equations but several more unknowns. This suggests that directly fitting some of the parameters is still needed, with the methods discussed in Section 2.2.1. Unfortunately, after this process the derived fits for the absorption cross sections were very poor. Errors in k_1 and $\nu\Sigma_{f1}$ propagate significantly into Σ_{a1}, largely because k_1 is small, as shown in the following error propagation study:

$$\Sigma_{a1} = \frac{\nu\Sigma_{f1}}{k_1} - \Sigma_{R1} \tag{13}$$

$$\sigma^2(\Sigma_{a1}) = \left(\frac{1}{k_1}\right)^2 \sigma^2(\nu\Sigma_{f1}) + \left(\frac{\nu\Sigma_{f1}}{k_1^2}\right)^2 \sigma^2(k_1) + \sigma^2(\Sigma_R) \tag{14}$$

$$\frac{\sigma^2(\Sigma_{a1})}{\Sigma_{a1}^2} = \underbrace{\left(\frac{\nu\Sigma_{f1}}{\Sigma_{a1}k_1}\right)^2}_{large} \frac{\sigma^2(\nu\Sigma_{f1})}{\nu\Sigma_{f1}^2} + \underbrace{\left(\frac{\nu\Sigma_{f1}}{\Sigma_{a1}k_1}\right)^2}_{large} \frac{\sigma^2(k_1)}{k_1^2} + \left(\frac{\Sigma_{R1}}{\Sigma_{a1}}\right)^2 \frac{\sigma^2(\Sigma_R)}{\Sigma_R^2}. \tag{15}$$

Thus, this physics-based grouping of parameters was not a successful process for fitting nuclear data. Rather, directly fitting the absorption cross sections is determined to be the better option.

Table 1: Input and output specification for the data module

Inputs
Burnup (0-70 MWD/kgU)
Enrichment (3-5 w/o)
IFBA (boolean)
WABA (boolean)
Gadolinium (boolean)

3 Implementation

The ultimate goal of producing a library for nuclear data retrieval was accomplished using only direct curve fitting of 2-group parameters with burnup and enrichment, implemented as a library of ANSI C functions. An interface with Python was also developed using SWIG.

This final product is the second iteration of a pure Python implementation which included both curve fitting and the two previously described interpolation methods, as implemented in the open source Scientific Tools for Python module SciPy. This section discusses the details of the initial implementation and timing studies that led to the choice of the curve fitting methods, as well as provides documentation for the C library produced in the second iteration.

This work was done with Python 2.7, and GCC 4.5.2.

3.1 Inputs and Outputs

As discussed previously, the scope of this data module was limited in order to achieve the speed required to carry out intensive fuel cycle optimization studies. As such the inputs from the user are limited, as shown in Table 1. For each of the types of burnable poisons , only one pattern was chosen.

The outputs contain all of the 2 group parameters typically available in the CASMO output, as shown in Table 2.

The output values are accessed via the output keys in Table 2 in the NUCLEAR_DATA C structure via the access function:

```
NUCLEAR_DATA get_2g_parms(B,ENR, IFBA,WABA,GAD);
```

Table 2: Data module outputs

Outputs			
SM2 _NO_XE	BOR2_XE	K2	NUFISS2
ABS1	DIFF1	KAPPA	REMOV1
ABS2	DIFF2	M2 NO XE	SM2 XE
BOR1_NO_XE	K_INF_NO_XE	M2_XE	XE_YIELD
BOR1_XE	K_INF_XE	NU	XE2_MAC
BOR2_NO_XE	K1	NUFISS1	XE2_MIC

Table 3: Initial speed tests for various methodologies

Method	Speed (lookups/s)
Curve Fitting	12645
Linear Interp.	1368
RBF	576

3.2 Python Implementation

The initial iteration of the data module was implemented in a Python module named nucleardata.py, which contained the initial curve fits, the interpolators, and the corresponding access functions. This prototype allowed for easy benchmarking and speed testing of each of the methodologies using the SciPy interpolation functions. In this iteration the curve fits were done manually for only 11 of the cross sections listed in Table 2, and evaluated with the "eyeball norm". The fits were done over the whole range of the data except for the initial burnup point before equilibrium Xenon, producing on average 1% relative error at the data points.

With this initial module the relative speed of each method was evaluated, as shown in Table 3.

After examining the details of each method, these results can be seen to be roughly tied to the number of operations needed for each evaluation. For instance, the manual fits were no more than third order polynomials at maximum in burnup, and the fits of the coefficients with enrichment were no more than quadratic. Thus for each evaluation at worst we have for each of the 4 coefficients with enrichment: 2 multiplies, 1 exponent, and 2 additions; and for burnup: 3 multiplies, 2 exponents, and 3 additions; a total of 28 operations. On the other hand, the RBF and linear interpolation methods involve weighted sums over the entire range of 4180 input data points, along with the calculation of a norm for each point in the case of RBF. It is speculated that the fact that SciPy is built on NumPy is the only reason the differences in speed weren't even greater, given the wide disparity in the complexity of the tasks. (NumPy speeds up numerical tasks in Python by interfacing with array and matrix operations written in C, much like SWIG.)

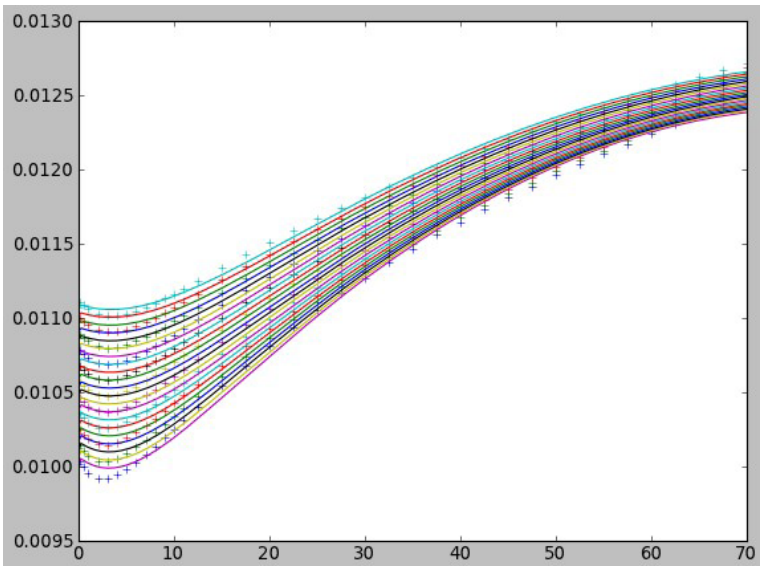


Figure 4: IFBA ABS2 fits

In light of these results, it became clear that the curve fitting methodology needed to be implemented in a compiled language in order to achieve the kinds of speeds necessary for exhaustive fuel cycle optimization studies. Indeed, initial tests with porting the simple curve fits for the 11 quantities to pure C showed a roughly 200X speedup.

3.3 ANSI C Implementation

Unlike the initial Python version, the C implementation of the nuclear data library was largely autogenerated with a series of scripts. This allowed for every 2-group parameter listed in Table 2 to be fit, and explicit error constraints evaluated. Also, in this implementation the differences between the no burnable poisons (NOBP) and each of the WABA, IFBA, and GAD cases run were also fit. For accuracy, these differences were fit piecewise, with one fit for the initial half of the burnup range and another for the later half. Furthermore, for all cases the first 2 burnup points were excluded from the fits to account for the fast Xenon transient. In this range, data is linearly interpolated. See Figure 4 for an example of a fit of ABS2 in the IFBA case.

The average relative error at the points was below 0.05% for all fits, as specified in the automated script that found the coefficients. This resulted in many higher order fits than in the previous iteration of the data module, and in some rare cases some non-physical wiggles (see Figure 5).

With these higher order fits, the piecewise split of the burnup range, and the larger number of parameters to fit, the number of operations increases dramatically for each

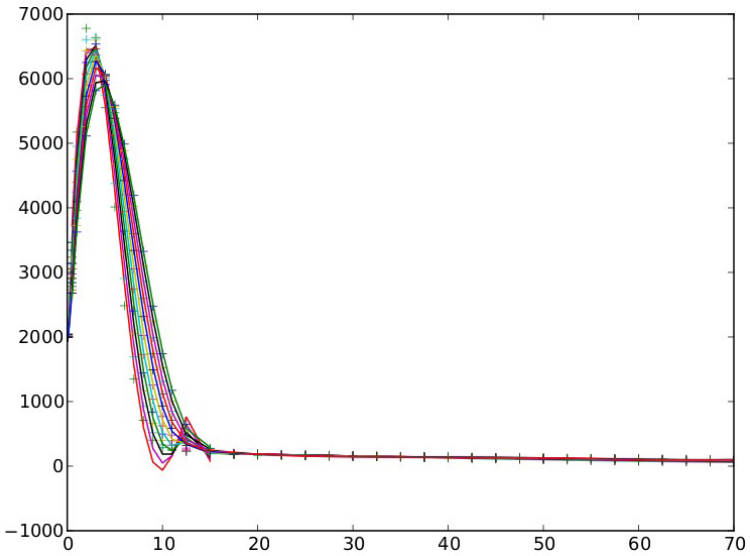


Figure 5: Fit of the difference of SM2_XE from NOBP in the GAD case, showing non-physical wiggles

Table 4: Lookup rates for the ANSI C implementation of the data module, in lookups/s

	SWIG	Pure C
No BPs	41432	45659
IFBA	18631	19628
WABA	18831	19991
GAD	17556	18266
IFBA+WABA+GAD	8720	9130

call to the data module access function. Despite this, we still see a huge speedup over the sparse pure Python implementation, as shown in Table 4. The SWIG implementation, notably, is not terribly slower than the pure C implementation.

While these timings are encouraging, it should be noted that there might still be considerable improvement to be had here. Specifically, if not all of the two group parameters provided by CASMO and listed in Table 2 are desired, it would be a simple task to delete or comment-out the few lines of code in the access function that calculates unwanted quantities. Also, most of these fits were found to achieve a high degree of accuracy in terms of relative error. However, for some quantities the differences between the NOBP and burnable poisons cases are so negligible that it would be sufficient to fit a line or a single value, rather than a higher order polynomial. Thus, it is estimated that with some selective refining of the libraries presented here, these data lookup rate might be nearly doubled.

Table 5: Assembly types for benchmark case

Material	Enrichment (w/o)	Burnup (MWD/kgU)	Burnable Poison
1	4.7	1	IFBA
2	4.5	50	Bare
3	3.3	20	Bare

```
 1  2  1  2  2  1  2  1  2  3
 2  1  2  1  1  2  1  2  1  3
 1  2  1  2  2  1  2  1  3  0
 2  1  2  1  1  2  1  2  3  0
 1  2  1  2  2  1  2  3  0  0
 2  1  2  1  1  2  3  0  0  0
 1  2  1  2  2  3  0  0  0  0
 2  1  2  1  3  0  0  0  0  0
 1  2  3  3  0  0  0  0  0  0
 3  3  0  0  0  0  0  0  0  0
```

Figure 6: Assembly layout for benchmark problem

3.4 Code Package

Attached with this report is a tarball containing all the scripts and CASMO input files necessary to reproduce these results, along with the final C libraries; in each folder a README file explains the nature of the folder's contents. Plots of all the fits can be found in 4 PDFs, and the final product libraries are duplicated in the C and SWIG directories as `nucleardata.c`, `nucleardata.h`, `nucleardata_IFBA.c`, `nucleardata_IFBA.h`, etc...

4 Benchmarking

To loosely quantify the effect of the approximations of using fits, the cross sections from the direct fit and interpolation schemes were inserted into a 2-D neutronics code (part of Jeremy Robert's SERMENT package). Only one case was considered. Assembly types are summarized in Table 5; geometry in Figure 6; and results in Table 6. Reference results were taken to be those generated with CASMO.

Clearly, the fitting schemes lead to significantly more error than linear interpolation. Note that this is only a single case being considered, and burnup points outside of the ones used in the interpolation schemes will lead to slightly larger error. However, this demonstrates the order of magnitude of the error adequately.

Table 6: Results for benchmark

Data Scheme	Eigenvalue Error	Maximum Fission Rate RE
Direct Fitting	175 pcm	5.47%
Linear Interp.	3.8 pcm	0.074%
RBF	3.2 pcm	0.071%

5 Conclusions and Future Work

5.1 Conclusions on Methodologies

Fitting nuclear data to various curves is shown to be a very quick approximation for a formal reactor database. However, it comes at the sacrifice of a fair degree of accuracy, as fits within 1% yielded a maximum fission rate relative error of 5% and an eigenvalue error of 175 pcm. It could easily be argued that an efficiently implemented interpolation scheme for a minimized number of statepoints would not be much less efficient and could yield much better results. However, the curve fitting methodology has the advantage that no proprietary data need be stored. While linear interpolation requires the exact state points be stored, fits wrap all this data with only a few coefficients.

5.2 Suggestions for Implementation and Expansion of the Database

A straightforward extension of the curve fit methodology would be to add temperature and soluble boron concentration as independent variables. This would allow the optimization tool to draw on a much larger subset of possible conditions in the core. Also, multiple IFBA thicknesses, WABA operating histories, and gadolinium enrichments are desired. However, it should be noted that it is not necessary to fit continuous functions to discrete variables – i.e. when IFBA thicknesses are set at discrete values by the manufacturer, there is no need to support a continuously varying thickness, as this adds extra complication with little reward.

The current fits could be made to be more selective, as the routines currently return parameters that are likely not needed by the rest of the optimization suite. Also, better logic could be implemented to determine the best fit for each set of nuclear data – especially with data that stays within a very small interval, highly precise fits are not needed. Also, to improve accuracy, sensitivity studies could be performed to determine which fits need to be made tighter.

Finally, other reactor physics parameters such as the moderator temperature coefficient and history effects should be added to this module for future versions.