Binary Formulation of Decay Equations

Cameron R. Bates[2] for Anthony Scopatz[1]

Paul Wilson[1]

[1] The University of Wisconsin-Madison
[2] Los Alamos National Laboratory

ANS Summer Meeting, June, 2015

Outline

- Motivation
- Method Derivation
- Implementation

Motivation

There are many situations in which the Bateman equations [1] should be solved for decay channels only, rather than for generic transmutation:

- Non-reactor components of the nuclear fuel cycle (storage, disposition, transit)
- Activation analysis
- Dosage calculation

Many solvers of the Bateman equations are not targeted to this reduced use case.

Motivation

Bateman equations solvers for decay-only calculations can be slow because:

- They perform a full transmutation calculation but with zero neutron flux.
- They read in data libraries from disk each calculation.
- Perform excess floating-point calculations even for decay channels, which also add algorithmic error.

For example, ORIGEN v2.2 [2] for decay is O($\approx$ 1 sec), most of which is I/O.

This does not scale well to large systems, such a fuel cycle with 70k fuel assemblies or reactor dose calculations with 50k volume elements.

Goal

Create a decay-only solver for the Bateman equations that is much faster mathematically and algorithmically than current implementations.

Such a solver will have reduced algorithmic error stemming from fewer FLOPS.

Canonical Bateman Equations for Decay

$$N_C(t) = \frac{N_1(0)}{\lambda_C} \cdot \gamma \cdot \sum_{i=1}^{C} \lambda_i c_i e^{-\lambda_i t} \tag{1}$$

$$c_i = \prod_{j=1, i \neq j}^{C} \frac{\lambda_j}{\lambda_j - \lambda_i} \tag{2}$$

Symbol	Meaning
C	length of the decay chain
i	index for ith species, on range [1, C]
j	index for jth species, on range [1, C]
t	time [seconds]
$N_i(t)$	number density of the ith species at time t
$t_{1/2,i}$	half-life of the ith species
λ_i	decay constant of ith species, $\ln(2)/t_{1/2,i}$
γ	total branch ratio for this chain

Canonical Bateman Equations for Decay

For first (parent) species in a chain:

$$N_C(t) = N_1(0)e^{-\lambda_1 t} \tag{3}$$

For last (stable child) species in a chain:

$$N_{\lambda_C \to 0}(t) = N_1(0)\gamma \left[1.0 - \sum_{i=1}^{C-1} \left(\prod_{j=1, i \neq j}^{C-1} \frac{\lambda_j}{\lambda_j - \lambda_i} \right) e^{-\lambda_i t} \right] \tag{4}$$

Method Derivation

By grouping constants, it is possible to express the Bateman equations as a simple sum of exponentials:

$$N_C(t) = N_1(0) \sum_{i=1}^{C} k_i e^{-\lambda_i t} \tag{5}$$

In Equation 5, the coefficients k_i are defined as follows:

$$k_i = \frac{\gamma}{\lambda_C} \lambda_i c_i \tag{6}$$

Precomputing the k_i, reduces the number of FLOPS per term in the summation from $O(3C + 1)$ to $O(4)$.

Method Derivation

Modern processors are inherently binary.

Thus, naively computing 2^x is on average about 25% faster than computing e^x on most architectures.

Note: some processors come with a special instruction for computing $\exp(x)$ as fast (or faster) than $\exp 2(x)$. Intel seems to do this with SSE2, when used at compile time.

Recasting the Bateman equations away from the natural exponent to base-2 yields a 'binary' reformulation.

Method Derivation

Define a precomputed constant a_i as,

$$a_i = \frac{-1}{t_{1/2,i}} \tag{7}$$

Thus, the final form of the binary representation of the Bateman equations are seen here:

$$N_C(t) = N_1(0) \sum_{i=1}^{C} k_i \cdot 2^{a_i t} \tag{8}$$

Every term in the sum is $O(3)$ and every operation is native to the instruction set.

Method Derivation

For first (parent) species in a chain:

$$N_C(t) = N_1(0) \cdot 2^{a_1 t} \tag{9}$$

For last (stable child) species in a chain:

$$N_{\lambda_C \to 0}(t) = N_1(0) \left[1.0 + \sum_{i=1}^{C-1} \lim_{\lambda_C \to 0} (k_i) \cdot 2^{a_i t} \right] \tag{10}$$

Method Derivation

- Relies on completely precomputed k_i and a_i.
- No I/O since constants are compiled into implementation.
- Preserves all physical processes since no assumptions were made aside from the Bateman equations.
- Not possible to further reduce number of operations analytically.

Implementation

A reference implementation of this method is available through the pyne library [3].

Cyclus [4] uses this implementation for nuclear fuel cycle simulation.

Even this implementation was forced to make certain assumptions in the face of floating point arithmetic for reasons of performance, compile-ability, and computability.

Implementation Assumptions

Spontaneous fission chains are not tallied as they lead to an explosion of the total number of chains while contributing to extraordinarily rare branches. For most species the relative error induced is $O(< 10^{-8})$. Relatively few and rare nuclides, such as Cm-250, have spontaneous fission branch greater than 5%. (Helps with **compilability**.)

Decay alphas are not treated as He-4 production. This can lead to errors that are less than 2% of the total mass of all chains for a nuclide. (**computability**)

To prevent other sources of floating point error, a nuclide is determined to be stable when $\lambda_i < 10^{-16}$, rather than when $\lambda_i = 0.0$. (**computability**)

Implementation Assumptions

For chains longer than length 2, any term whose half-life is less than 10^{-8} of the sum of all half-lives in the chain is dropped. This filtering prevents excessive calculation from species which do not significantly contribute to end atom fractions. If the filtering causes there to be less than two terms in the summation, then the filtering is turned off and all terms are computed. (**performance**)

If a chain has any NaN decay constants, the chain is rejected. (**computability**)

If a chain has any infinite k_i values, the chain is rejected. (**computability**)

Comparison to ORIGEN - Speed

Run times are fast:

- **ORIGEN v2.2:** O(1+ sec)
- **Binary:** O(5-100 μsec)

Factor of 10^3 to 10^6 speedup.

ORIGEN runtimes likely stem from from excessive I/O.

The binary timings depend on the number of chains in a decay and timing results include Python round tripping.

Comparison to ORIGEN - Method

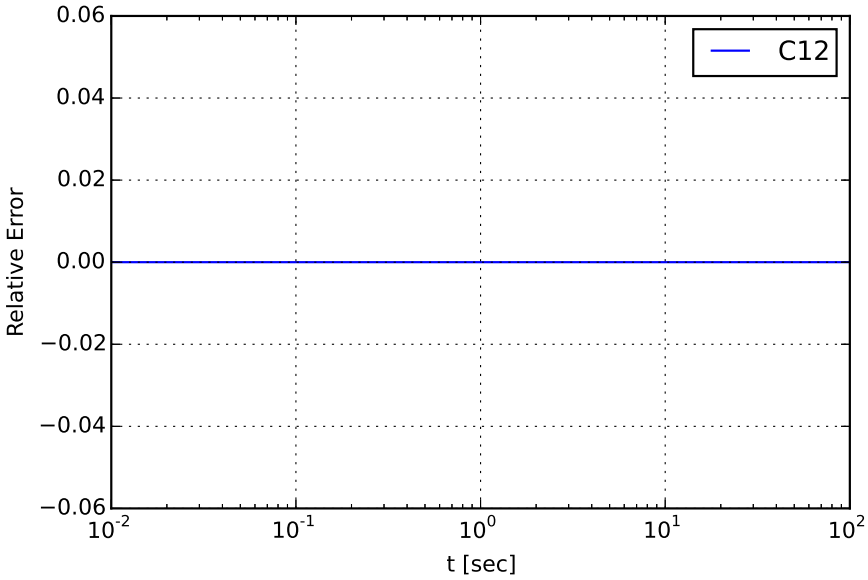
Take x as the mass of a species in a material after decay in the binary formulation, and y as the mass of the same species decayed via ORIGEN. Define the relative error as:

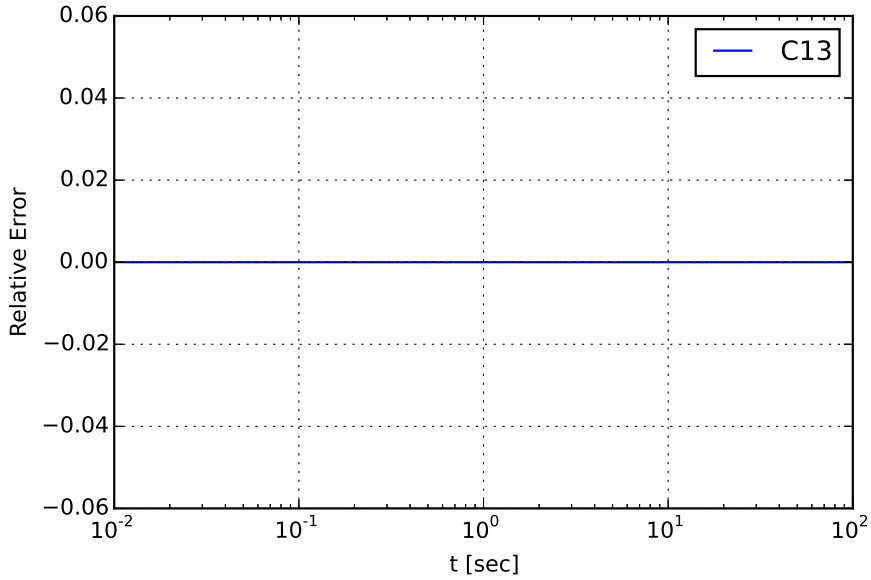
$$\epsilon = \frac{2 \cdot |x - y|}{x + y}, \epsilon \in [0, 2] \tag{11}$$

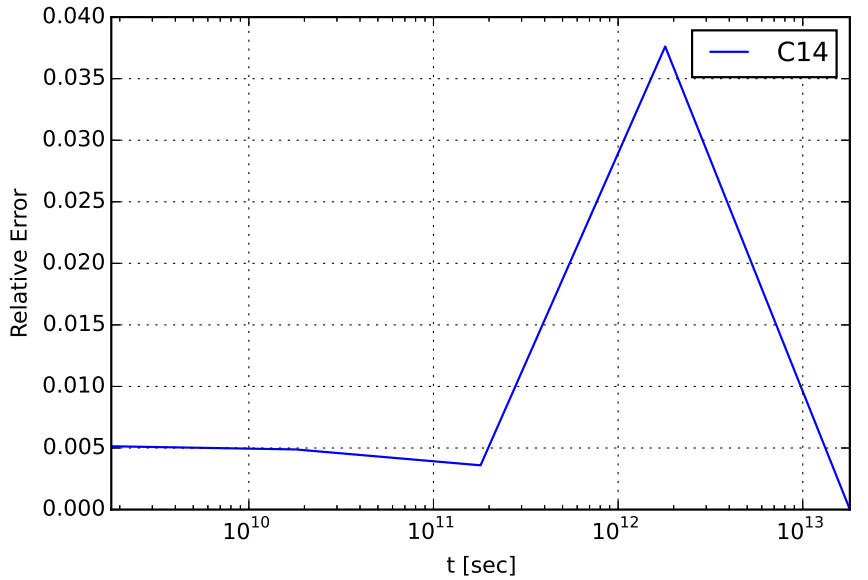
Experiment: Decay a unit mass of a parent nuclide whose half-lives in ORIGEN and ENSDF match to within the precision given in the ORIGEN library. Run this decay for 0.01, 0.1, 1.0, 10, 100 times the half-life of the nuclide. Compare the relative errors and take the maximum relative error for all children.

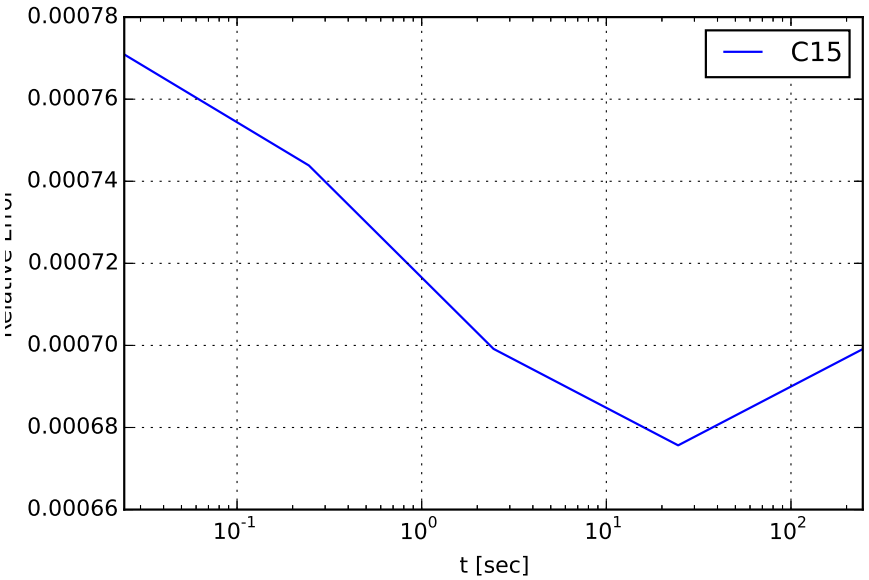
This generated 2802 nuclide/decay time cases that are on a common basis.

Comparison to ORIGEN - Carbon Isotopes

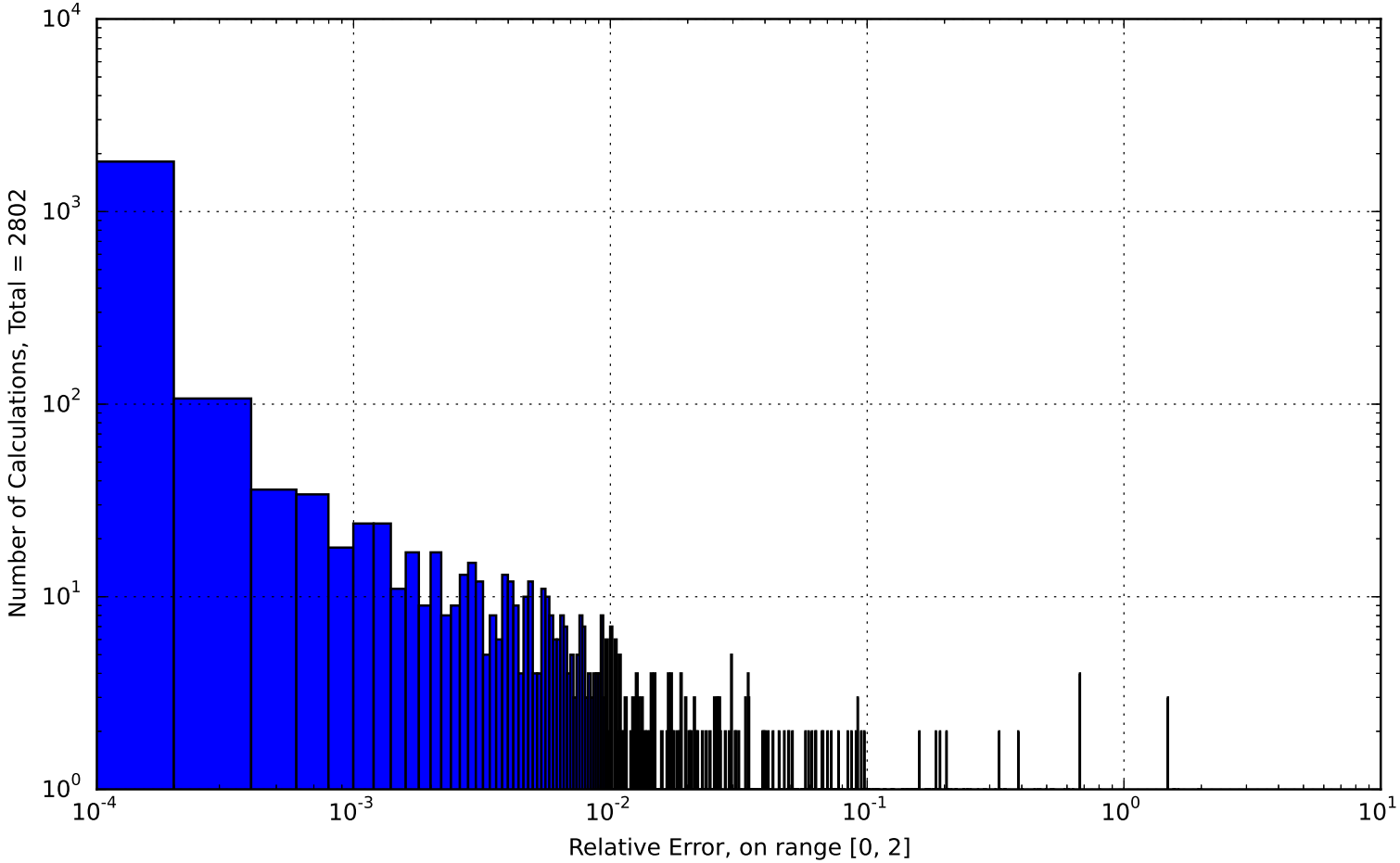








Comparison to ORIGEN - All Cases



Note: log-log scale to accentuate differences.

Comparison to ORIGEN - Outliers

Outliers with high relative error in one of their children almost always stems from discrepancies in branch ratios between ORIGEN and ENSDF.

High relative errors are also sometimes caused by discrepancies in the half-lives of the children.

In all cases, the mass of the child times its relative error is small. This mass-weighted relative error, w, indicates that the total impact on the decay calculation is small.

Outliers are therefore an issue with data and precision rather than an indictment of the binary formulation.

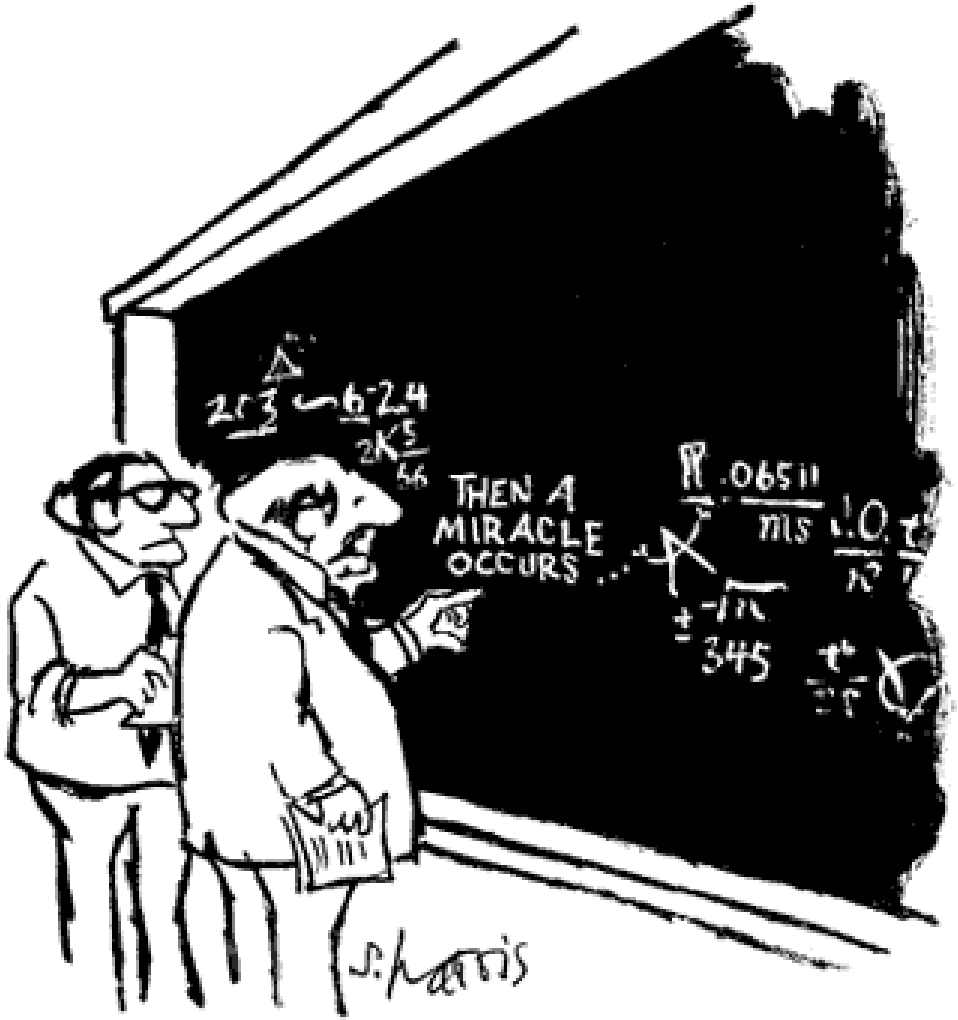
Comparison to ORIGEN - In-114 Outlier

Differences in relative error come from differing branch ratios.

t [sec]	ϵ, rel. err.	child	child mass	w, weighted err.
0.719	0.029356	Sn-114	0.006873	0.000202
7.190	1.483651	Cd-114	0.002259	0.003352
71.900	1.483744	Cd-114	0.016870	0.025031
719.000	1.483783	Cd-114	0.033710	0.050018
7190.000	1.483744	Cd-114	0.033740	0.050062

Mass weighted errors remain small here even when relative error is large.

QUESTIONS?



"I think you should be more explicit here in step two."

References I

Harry Bateman.
The solution of a system of differential equations occurring in the theory of radioactive transformations.
In *Proc. Cambridge Philos. Soc*, volume 15, pages 423–427, 1910.

AG Croff.
Origen2: a revised and updated version of the oak ridge isotope generation and depletion code.
Technical report, Oak Ridge National Lab., TN (USA), 1980.

the PyNE Development Team.
PyNE: The Nuclear Engineering Toolkit, 2014.

the Cyclus Development Team.
Cyclus, 2015.
http://fuelcycle.org/.