The Journal of Open Source Software

radioactivedecay: A Python package for radioactive decay calculations

Alex Malins[1]

1 Center for Computational Science & e-Systems (CCSE), Japan Atomic Energy Agency (JAEA), 178-4-4 Wakashiba, Kashiwa, Chiba, 277-0871, Japan

DOI: 10.21105/joss.03318

Software

- Review
- Repository
- Archive

Editor: Kelly Rowland

Reviewers:

- @munkm
- @shyamd

Submitted: 30 April 2021
Published: 14 June 2021

Summary

`radioactivedecay` is a Python package for radioactive decay modelling. It contains functions to fetch decay data, define inventories of radionuclides and perform decay calculations. The default nuclear decay dataset supplied with `radioactivedecay` is based on ICRP Publication 107, which covers 1252 radioisotopes of 97 elements. The code calculates an analytical solution to a matrix form of the decay chain differential equations using double or higher precision numerical operations. There are visualization functions for drawing decay chain diagrams and plotting activity decay curves.

Statement of Need

Calculations for the decay of radioactivity and the ingrowth of progeny underpin the use of radioisotopes in a wide range of research and industrial fields, spanning from nuclear engineering, medical physics, radiation protection, environmental science and archaeology to non-destructive testing, mineral prospecting, food preservation, homeland security and defence. `radioactivedecay` is an open source, cross-platform package for decay calculations and visualization. It supports decay chains with branching decays and metastable nuclear isomers. It includes a high numerical precision decay calculation mode which resolves numerical issues when using double-precision floating-point numbers for decaying chains containing radionuclides with disparate half-lives (Bakin et al., 2018).

This set of features distinguishes `radioactivedecay` from other commonly-used decay packages, such as `Radiological Toolbox` and `PyNE`. `Radiological Toolbox` is a closed-source Windows application, so it is not easily scriptable and its use of double-precision arithmetic makes it susceptible to numerical round-off errors. `PyNE` uses approximations to help mitigate numerical issues, however these may potentially affect accuracy. Moreover as of `v0.7.3`, `PyNE` does not correctly model metastable nuclear isomers within decay chains, which means, for example, it cannot simulate the production of $^{99m}\mathrm{Tc}$ from $^{99}\mathrm{Mo}$ for medical imaging applications.

Theory and Implementation

`radioactivedecay` implements the solution to the decay differential equations outlined by Amaku et al. (2010). If vector $\mathbf{N}$ contains the number of atoms of each radionuclide in a system, its elements N_i can be ordered such that no progeny (either first or subsequent generation) of radionuclide i has itself an index lower than i. Ordering in this manner is

possible because natural radioactive decay processes do not increase the mass number of the decaying radionuclide, and there are no cyclic decay chains where radionuclide i can decay to other radionuclides then reform itself (Ladshaw et al., 2020). Note metastable nuclear isomers have distinct indices from their ground states in $\mathbf{N}$.

The radioactive decay chain differential equations expressed in matrix form are:

$$\frac{\mathrm{d}\mathbf{N}}{\mathrm{d}t} = \Lambda\mathbf{N}. \tag{1}$$

Λ is a lower triangular matrix with elements:

$$\Lambda_{ij} = \begin{cases} 0 & \text{for } i < j, \\ -\lambda_j & \text{for } i = j, \\ b_{ji}\lambda_j & \text{for } i > j. \end{cases} \tag{2}$$

λ_j is the decay constant of radionuclide j, and b_{ji} is the branching fraction from radionuclide j to i. Λ is diagonalizable so its eigendecomposition can be used to rewrite Equation 1 as:

$$\frac{\mathrm{d}\mathbf{N}}{\mathrm{d}t} = C\Lambda_d C^{-1}\mathbf{N}. \tag{3}$$

Λ_d is a diagonal matrix whose elements are the negative decay constants, i.e. $\Lambda_{dii} = -\lambda_i$. Matrix C and its inverse C^{-1} are both lower triangular matrices that are calculated as:

$$C_{ij} = \begin{cases} 0 & \text{for } i < j, \\ 1 & \text{for } i = j, \\ \frac{\sum_{k=j}^{i-1} \Lambda_{ik}C_{kj}}{\Lambda_{jj}-\Lambda_{ii}} & \text{for } i > j, \end{cases} \quad \text{and} \quad C^{-1}_{ij} = \begin{cases} 0 & \text{for } i < j, \\ 1 & \text{for } i = j, \\ -\sum_{k=j}^{i-1} C_{ik}C^{-1}_{kj} & \text{for } i > j. \end{cases} \tag{4}$$

The analytical solution to Equation 3 given an initial condition of $\mathbf{N}(0)$ at $t = 0$ is:

$$\mathbf{N}(t) = Ce^{\Lambda_d t}C^{-1}\mathbf{N}(0). \tag{5}$$

$e^{\Lambda_d t}$ is a diagonal matrix with elements $e^{\Lambda_d t}_{ii} = e^{-\lambda_i t}$. `radioactivedecay` evaluates Equation 5 upon each call for a decay calculation.

Matrices C and C^{-1} are independent of time so they are pre-calculated and imported from files into `radioactivedecay`. C and C^{-1} are stored in sparse matrix data structures to minimize memory use and maximize efficiency when computing the matrix multiplications in Equation 5. For decay calculations with double-precision floating-point operations, C and C^{-1} are stored in `SciPy` (Virtanen et al., 2020) Compressed Sparse Row (CSR) matrix data structures. Conversely, they are stored in `SymPy` (Meurer et al., 2017) SparseMatrix data structures for high numerical precision calculations.

The high numerical precision decay calculation mode resolves numerical issues arising from using double-precision floating-point numbers for decay calculations for chains containing nuclides with disparate half-lives. One example is the decay chain for $^{254}\mathrm{Es}$, which contains $^{238}\mathrm{U}$ (4.468 billion year half-life) and $^{214}\mathrm{Po}$ ($t_{1/2}$ is 164.3 μs half-life). This a 20 orders of magnitude difference in half-life. Loss of numerical precision inevitably occurs when evaluating the off-diagonal elements of C and C^{-1} in Equation 4 with double-precision floating-point numbers (which hold approximately 15 decimal places of numerical precision). Note loss of precision also occurs in the converse scenario, i.e. when a decay chain contains radionuclides with similar half-lives. However this scenario does not occur in the ICRP Publication 107

decay dataset, as the relative difference between half-lives of any two radionuclides in the same decay chain is always greater than 0.1%.

The default operation of the high precision decay mode is to evaluate Equation 5 using floating-point numbers with 320 significant figures of precision. This is sufficient precision to ensure accurate results for any physically relevant decay calculation users may wish to perform. Moreover, computations in the high precision mode are fast, taking only 0.5 seconds on a notebook equipped with an Intel Core i5-8250U processor.

Decay Datasets

The default decay dataset supplied with `radioactivedecay` is based on ICRP Publication 107 (Eckerman & Endo, 2008). Endo et al. (2005) and Endo & Eckerman (2007) describe the development of the ICRP Publication 107 decay dataset. The raw data was converted into dataset files suitable for `radioactivedecay` using a Jupyter notebook. Along with `SciPy` and `SymPy` versions of the sparse matrices C and C^{-1}, the dataset files store radionuclide half-lives, decay constants, progeny, branching fractions and decay modes. Although ICRP Publication 107 is the default dataset, `radioactivedecay` allows users to import and use other decay datasets.

Main Functionality

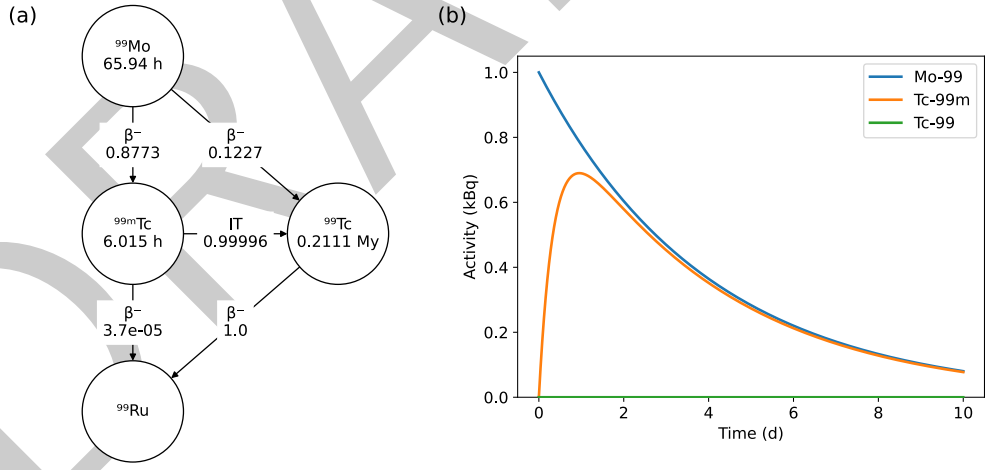

Figure 1: Examples of the plotting capabilities of `radioactivedecay`: (a) Decay chain diagram for molybdenum-99. (b) Graph showing the decay of 1 kBq of $^{99}\mathrm{Mo}$ along with the ingrowth of $^{99m}\mathrm{Tc}$ and a trace quantity of $^{99}\mathrm{Tc}$.

The main functionality of `radioactivedecay` is based around two classes: the `Radionuclide` class and the `Inventory` class. The `Radionuclide` class is used for fetching decay data about a single radionuclide, such as its half-life, the decay modes, the progeny and the branching fractions. It creates diagrams of the radionuclide's decay chain (ex. Figure 1(a)) based on `NetworkX` (Hagberg et al., 2008).

An `Inventory` can contain multiple radionuclides, each with an associated radioactivity. The `decay()` and `decay_high_precision()` methods calculate the decay of the `Inventory`, adding any ingrown radioactive progeny automatically. Plots can be made of the variation of radionuclide activities over time (ex. Figure 1(b)) based on `Matplotlib` (Hunter, 2007).

Validation

Decay calculations with `radioactivedecay v0.3.2` were cross-checked against `Radiological Toolbox v3.0.0` (Hertel et al., 2015) and `PyNE v0.7.3` (Scopatz et al., 2012) (see Jupyter notebooks in the project repository). `Radiological Toolbox` employs the ICRP Publication 107 decay data. Fifty radionuclides were randomly selected and a decay calculation was performed for 1 Bq of each for a random decay time within a factor of 10^{-3} to 10^{3} of the half-life. Differences between decayed activities reported by each code were within 1% of each other in 64% of cases. Discrepancies greater than 1% were attributed to rounding differences, erroneous results from `Radiological Toolbox`, or numerical issues relating to decay chains containing radionuclides with disparate half-lives.

A dataset was prepared for `radioactivedecay` with the same Evaluated Nuclear Structure Data File (ENSDF, 2019) decay data as used by `PyNE v0.7.3`. Bugs in `PyNE v0.7.3` cause incorrect decay calculation results for chains containing metastable nuclear isomers, $^{183}\mathrm{Pt}$, $^{172}\mathrm{Ir}$ or $^{152}\mathrm{Lu}$. Radionuclides affected by this were excluded from the comparison. The decay of 1 Bq of every radionuclide was calculated for multiple decay times varying from 0 to 10^{6} times the radionuclide's half-life. The absolute difference between the decayed activities reported by each code was less than 10^{-13} Bq. Relative differences depended on the magnitude of the activity. Relative errors of greater than 0.1% only occurred when the calculated activity was less than 2.5×10^{-11} Bq, i.e. 10 orders of magnitude smaller than the initial activity of the parent radionuclide. The discrepancies between the two codes were attributed to methodological differences for computing decay chains with radionuclides with large disparities between half-lives, and numerical issues arising from double-precision floating-point operations.

Limitations

`radioactivedecay` does not model neutronics, so cannot evaluate radioactivity produced from activations or induced fission. It does not support external sources of radioactivity input or removal from an inventory over time. Caution is required if decaying backwards in time, as this can cause floating-point overflows when computing the exponential terms in Equation 5.

There are also some limitations associated with the ICRP Publication 107 decay dataset. It does not contain data for the radioactivity produced from spontaneous fission decay pathways and the minor decay pathways of some radionuclides. More details on limitations are available in the documentation, Endo et al. (2005), and Endo & Eckerman (2007).

Acknowledgements

We thank Mitsuhiro Itakura, Kazuyuki Sakuma & colleagues in JAEA's Center for Computational Science & Systems for their support for this project, Kenny McKee & Daniel Jewell for helpful suggestions, and Björn Dahlgren, Anthony Scopatz & Jonathan Morrell for their work on radioactive decay calculation software.

References

Amaku, M., Pascholati, P. R., & Vanin, V. R. (2010). Decay chain differential equations: Solution through matrix algebra. *Computer Physics Communications*, *181*(1), 21–23. https://doi.org/10.1088/0952-4746/26/3/N02

Bakin, R. I., Kiselev, A. A., Shvedov, A. M., & Shikin, A. V. (2018). Computational Errors in the Calculation of Long Radioactive Decay Chains. *Atomic Energy*, *123*(6), 406–411. https://doi.org/10.1007/s10512-018-0360-2

Eckerman, K. F., & Endo, A. (2008). ICRP 107: Nuclear Decay Data for Dosimetric Calculations. *Annals of the ICRP*, *38*(3), 119. https://doi.org/10.1016/j.icrp.2008.10.003

Endo, A., & Eckerman, K. F. (2007). *JAEA-Data/Code 2007-021: Nuclear Decay Data for Dosimetry Calculation - Data for Radionuclides with Half-lives Less than 10 Minutes*. https://doi.org/10.11484/jaea-data-code-2007-021

Endo, A., Yamaguchi, Y., & Eckerman, K. F. (2005). *JAERI 1347: Nuclear Decay Data for Dosimetry Calculation; Revised data of ICRP Publication 38*. https://doi.org/10.11484/jaeri-1347

ENSDF. (2019). *From ENSDF database as of October 4, 2019. Version available at:* https://www.nndc.bnl.gov/ensarchivals/

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 11–15. http://conference.scipy.org/proceedings/SciPy2008/paper_2/

Hertel, N. E., Eckerman, K. F., & Sun, C. (2015). Radiological Tookbox 3.0.0. *Transactions of the American Nuclear Society*, *113*, 977–980. https://www.ans.org/pubs/transactions/article-38022/

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Ladshaw, A., Wiechert, A. I., Kim, Y., Tsouris, C., & Yiacoumi, S. (2020). Algorithms and algebraic solutions of decay chain differential equations for stable and unstable nuclide fractionation. *Computer Physics Communications*, *246*, 106907. https://doi.org/10.1016/j.cpc.2019.106907

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Scopatz, A. M., Romano, P. K., Wilson, P. P. H., & Huff, K. D. (2012). PyNE: Python for nuclear engineering. *Transactions of the American Nuclear Society*, *107*, 985–987. https://www.ans.org/pubs/transactions/article-14978

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2