Nuclear data processing capabilities in OpenMC

Paul Romano[1,a] and Sterling Harper[2]

[1] Argonne National Laboratory, 9700 S. Cass Avenue, Lemont, IL 60439, USA
[2] Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

Abstract. This work describes newly developed features of the OpenMC code for nuclear data processing. OpenMC, in addition to being a transport code, includes a rich, extensible Python API that enables programmatic pre- and post-processing. A new `openmc.data` package in the Python API enables users to parse ENDF and ACE files and convert them to an HDF5 format. With this capability, the OpenMC transport solver now relies on HDF5 nuclear data files rather than ACE files produced from NJOY. Moving to a native format will give much greater flexibility for researchers to explore new methods and algorithms that rely on storing data that is not present in the ACE format. Additionally, the module may serve as an independent implementation of the proposed Generalized Nuclear Data (GND) format in the future.

1. Introduction

The Monte Carlo method is often applied to particle transport simulations and is considered the most accurate method of solution because of the lack of approximations relative to deterministic methods. In order to attain such high accuracy, Monte Carlo methods also require a higher level of detail in the nuclear data representation, such as explicit cross sections for all possible reactions, a list of reaction products, and their yields/angle-energy distributions. This data typically originates from nuclear data evaluations as given in the ENDF-6 format [1], for example, the ENDF/B-VII.1 library [2]. ENDF evaluations unfortunately cannot be used directly—some processing of the data must be done in order for the Monte Carlo code to be able to sample reaction and product distributions. Often, NJOY [3] is used to process the ENDF-format data into the ACE format [4], which is used by several Monte Carlo codes including MCNP [5] and Serpent [6].

Until recently, OpenMC [7], an open-source Monte Carlo neutral particle transport code, also relied on ACE-format data. However, there was a desire to transition to a format having a clear hierarchical structure that matches the basic data abstraction within the core codebase of OpenMC and that is well documented and openly available. The present work describes the result of such an effort: the `openmc.data` package within OpenMC's Python API.

2. Design and class hierarchy

The purpose of the `openmc.data` package is to perform serialization and deserialization of nuclear data using various file formats, that is, to translate ACE- and ENDF-format data into an in-memory hierarchy of Python objects. The package also defines a new HDF5 [8] format that the Python objects can be serialized to and deserialized from. HDF5 was chosen as it provides a means to produce cross-platform binaries, a flexible type system, support for parallel I/O, and bindings in many languages. The HDF5 format is now the standard nuclear data format for OpenMC's transport solver. Figure 1 shows the flow of data in and out of classes in the Python API.

When deserializing data in files into Python objects, the same classes are used whether the files are ENDF or ACE formatted. Special care is needed because these file formats are substantially different and do not contain the same content. For instance, ENDF files contain resonance parameter data whereas ACE files contain only tabulated cross sections that were reconstructed from the resonance parameters and Doppler broadened.

At the top of the class hierarchy is the `IncidentNeutron` class that represents a single ENDF material or a single ACE continuous-energy neutron table. Each `IncidentNeutron` has a list of `Reactions`, and each `Reaction` has a list of `Products`. Each of these classes has a `from_ace()` and `from_endf()` factory method, and the factory methods can call one another recursively through the class hierarchy; that is, the `IncidentNeutron.from_ace()` method calls `Reaction.from_ace()` as appropriate. Figure 2 shows a Unified Modeling Language (UML) diagram for the top of the class hierarchy within `openmc.data`.

In the ENDF and ACE formats, reaction products are not always defined explicitly; as a result, one cannot have `from_ace()` and `from_endf()` methods for `Product`. Instead, functions that generate all reaction products for a given reaction are called from the `Reaction` factory methods. This situation occurs both for fission products (where the yield of fission products is defined separately) and for photon production (where products may be assigned to redundant reactions).

2.1. Energy-angle distributions

Each reaction `Product` can have one or more energy-angle distributions, which are represented by the abstract

[a] e-mail: `promano@anl.gov`

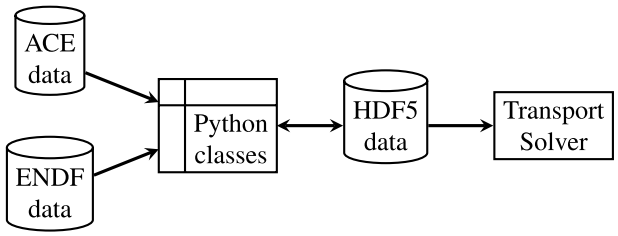

Figure 1. Flow of data from ACE/ENDF to transport with `openmc.data`.

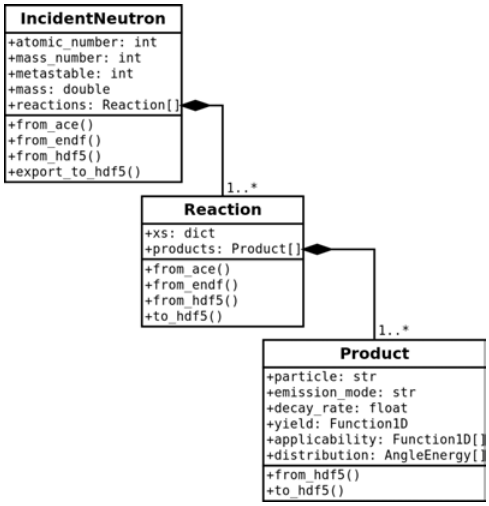

Figure 2. Top of class hierarchy within `openmc.data`.

class `AngleEnergy`. As in the ENDF format, these distributions can be either uncorrelated or correlated. Several different correlated angle-energy distribution classes exist, corresponding either to distributions in ENDF File 6 or to various ACE laws. Some distributions exist only for ACE data (e.g., `LevelInelastic`) whereas others exist only for ENDF data (e.g., `MadlandNix`). Figure 3 shows a UML diagram for the energy-angle distribution class hierarchy.

2.2. Functions and distributions

At the lowest level of the class hierarchy are functions and distributions of a single variable. The `Function1D` abstract class has several concrete implementations of which the most commonly used is `Tabulated1D`, which directly corresponds to an ENDF TAB1 record. Polynomials are also used in ACE and ENDF, so a `Polynomial` class handles these cases. Note that since NumPy provides a polynomial class, the implementation in `openmc.data` simply adds `from_hdf5()` and `to_hdf5()` methods. The `Sum` class represents the sum of several existing functions—this is used for redundant reactions and in a few other places. Figure 4 shows a UML diagram for the function class hierarchy.

Note that `openmc.data` distinguishes between a function of one variable and a univariate random variable. For random variables, classes from a separate `openmc.stats` package are used; wherever `Univariate` appears in Fig. 3, a univariate random variable is expected. These classes can also be used to define source distributions in space, angle, and energy for a simulation in OpenMC.

2.3. Temperature dependence

In `openmc.data`, cross sections and other data for a single target material at multiple temperatures do not need to be stored in separate instances as in the ACE format. Instead, reaction cross sections, unresolved resonance probability tables, and thermal scattering data are all stored at discrete temperatures; all other data (e.g., reaction product data) is considered to be independent of temperature. This approach obviates the use of cross-section "suffixes"[1] and enables users to simply identify nuclides and temperatures in their model. It can also significantly reduce the storage requirements for nuclear data both in HDF5 files and in memory during a transport simulation.

2.4. Relationship to GND

Much of the design of the class hierarchy has been influenced by the Generalized Nuclear Data (GND) format [9] being proposed by OECD/WPEC Subgroup 38. That being said, exact conformance with the GND structure is not an explicit goal at present since the specification of the GND data structure hierarchy is not yet complete and will likely change over the next few years as experience is gained. Since the `openmc.data` package is distributed alongside the transport solver, users can be assured that data libraries produced will work with the transport solver from the same version. Once the GND structure reaches a greater level of maturity and stability, changes to make the `openmc.data` class hierarchy and HDF5 format conform to GND could be made.

3. HDF5 library format

Each object in the in-memory class hierarchy generally corresponds to a group within the HDF5 file. However, HDF5 itself does not perform as well with many small datasets compared with fewer, larger datasets. Thus, an effort is made at the lowest level of the class hierarchy to use multidimensional arrays where possible as well as using attributes rather than datasets. Every group and dataset within an HDF5 file must store metadata which consumes on the order of a kB of disk space per group or dataset. Attributes, on the other hand, can be stored in the object header and thus often consume little to no extra disk space. As one example, the `Tabulated1D` object is stored as a single two-dimensional dataset with interpolation information stored as attributes.

A simple experiment was performed to determine how much disk space is saved by using attributes where possible rather than datasets. The API was modified such that all `to_hdf5()` methods would create a dataset wherever an attribute was originally used. The original API and the modified API were then used to convert every nuclide in the ENDF/B-VII.1 library. When using attributes, the resulting library consumed 841 MB. When datasets were used, however, the library required 1050 MB of disk space, a 25% increase.

We note that for each Python class, there is also a corresponding derived type within the OpenMC Fortran codebase that has a `from_hdf5()` type-bound procedure.

[1] For example, in the MCNP ENDF/B-VII.1 data library, a suffix of 80 c refers to data at 300 K.

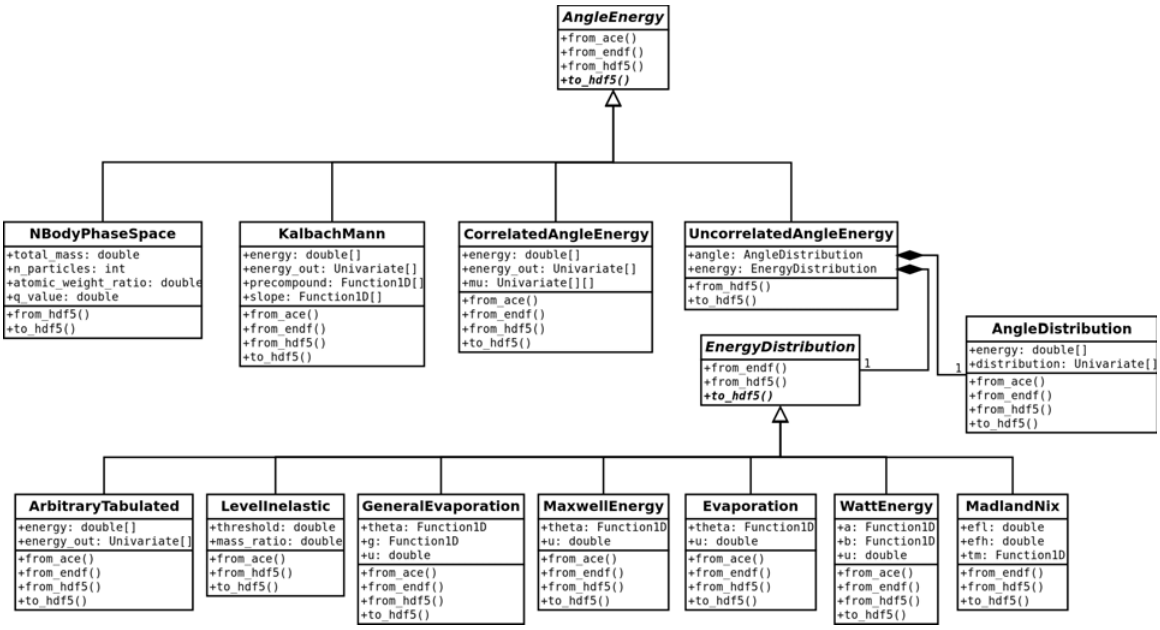

Figure 3. Energy-angle distribution class hierarchy within `openmc.data`.

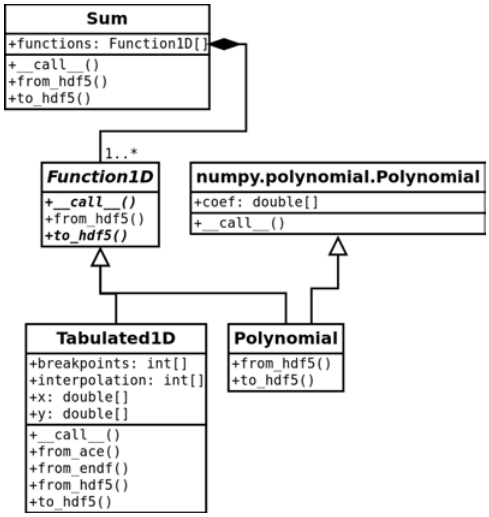

Figure 4. Class hierarchy for functions within `openmc.data`.

For distributions, each derived type also has a `sample()` type-bound procedure that can be called polymorphically.

4. Conclusions

A new `openmc.data` Python package gives users the ability to parse, analyze, and transform nuclear data in ENDF and ACE formats. It can also serialize and deserialize data to/from an HDF5 format that is used by OpenMC's transport solver. As OpenMC continues to expand in scope and capability, future data needs can be met easily through changes in the HDF5 data format. For example, a capability to augment ACE data with fission energy release data from an ENDF file has already been implemented and is used by OpenMC to tally recoverable energy release from fission.

Many benefits arise from having the nuclear data processing capabilities written in Python versus a lower-level language. Simple tasks to inspect or modify nuclear data can be performed in a short script. Additionally, a rich ecosystem of third-party packages for scientific computing already exists and can be leveraged, for example, NumPy, SciPy, and Pandas. A Python API also exists for OpenMC input generation and post-processing of results, and thus the `openmc.data` package is able to reuse existing functionality.

Currently, ENDF data can be deserialized; in order for it to be usable by the transport solver (i.e., convert it to HDF5), additional functionality will be needed in the API, namely, resonance reconstruction, Doppler broadening, and probability table generation. Work is underway to add resonance reconstruction using Cython to attain performance equivalent to C or Fortran code. Doppler broadening and probability table generation could be added in a similar manner. These additions would obviate the need to generate ACE-format data in the first place. Other additions to the `openmc.data` package may include an ability to merge windowed multipole data [10], adding classes for covariance data, and supporting photoatomic and photonuclear data.

The authors gratefully acknowledge the review and feedback provided by Adam Nelson, Colin Josey, and other members of the OpenMC development team during the course of this work. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

References

[1] M. Herman, A. Trkov, Tech. Rep. BNL-90365-2009, Brookhaven National Laboratory (2009)
[2] M.B. Chadwick, M. Herman, P. Obložinskỳ, M.E. Dunn, Y. Danon, A.C. Kahler, D.L. Smith, B. Pritychenko, G. Arbanas, R. Arcilla et al., Nucl. Data Sheets **112**, 2887 (2011)

[3] R.E. MacFarlane, D.W. Muir, R.M. Boicourt, A.C. Kahler, Tech. Rep. LA-UR-12-27079, Los Alamos National Laboratory, Los Alamos, New Mexico (2012)
[4] X-5 Monte Carlo Team, Tech. Rep. LA-CP-03-0284, Los Alamos National Laboratory (2008)
[5] T. Goorley, Tech. Rep. LA-UR-14-24680, Los Alamos National Laboratory (2014)
[6] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, T. Kaltiaisenaho, Ann. Nucl. Energy **82**, 142 (2015)
[7] P.K. Romano, N.E. Horelik, B.R. Herman, A.G. Nelson, B. Forget, Ann. Nucl. Energy **82**, 90 (2015)
[8] The HDF Group, *Hierarchical Data Format, version 5* (1997–2016), `http://www.hdfgroup.org/HDF5/`
[9] C.M. Mattoon, B.R. Beck, Eur. Phys. J. A **51**, 183 (2015)
[10] C. Josey, P. Ducru, B. Forget, K. Smith, J. Comput. Phys. **307**, 715 (2016)