Nuclear Data Processing Capabilities in OpenMC

Paul K. Romano[1] Sterling M. Harper[2]

[1]Argonne National Laboratory
Mathematics and Computer Science Division

[2]Massachusetts Institute of Technology
Department of Nuclear Science & Engineering

OpenMC

- Open source continuous-energy Monte Carlo particle transport code developed by ANL, MIT, and other contributors
- Currently only neutrons (developmental photon capability)
- Particular focus on HPC, ease of use, methods development
- Input scripts (Python API) generate XML files, HDF5 output files
- Most applications to-date have been reactors / criticality
- Until recently, relied on ACE format data produced by NJOY

Code: `https://github.com/mit-crpg/openmc`
Docs: `http://openmc.readthedocs.io`

Data Needs for OpenMC

Increasingly, require data that ACE doesn't provide, e.g.

- Development of new methods:
  - Windowed multipole
  - On-the-fly URR treatment
- Implementing new tallies:
  - Fission energy release
  - Decay rates

Question: supplement ACE or replace it?

Motivation for HDF5

Decided to move away from ACE:

- Archaic format, binaries not cross-platform
- Not easily extensible
- Not designed for multi-temperature problems
- Overloaded variables can cause confusion
- Only produced by a code under export control

Moving to **HDF5**[1] can help solve many of these problems:

- Binary files containing filesystem-like hierarchy
- Library enables applications to read/write data portably
- Bindings for many languages (C, C++, Fortran, Java, Python)

[1] Hierarchical Data Format version 5

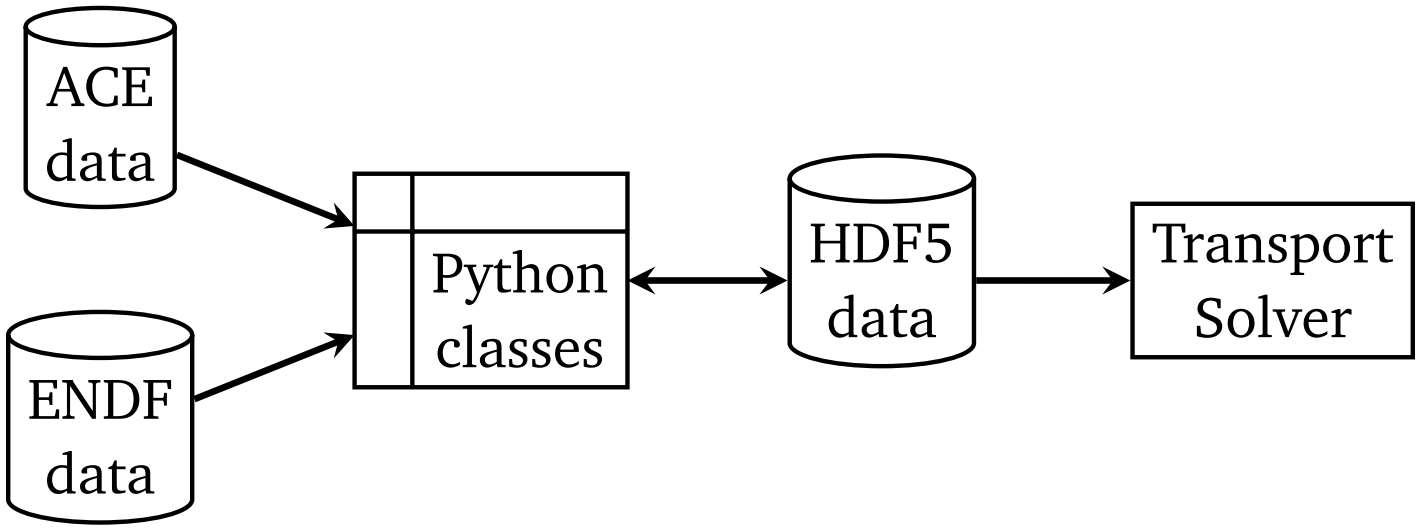
OpenMC: Python API

- Tightly couples input generation, simulation execution, and postprocessing/data analysis
- Leverages existing Python scientific computing ecosystem, namely NumPy, SciPy, h5py, and Pandas
- Multi-group cross section generation (`openmc.mgxs`), TRISO particle packing (`openmc.model`), geometry plotting, etc.

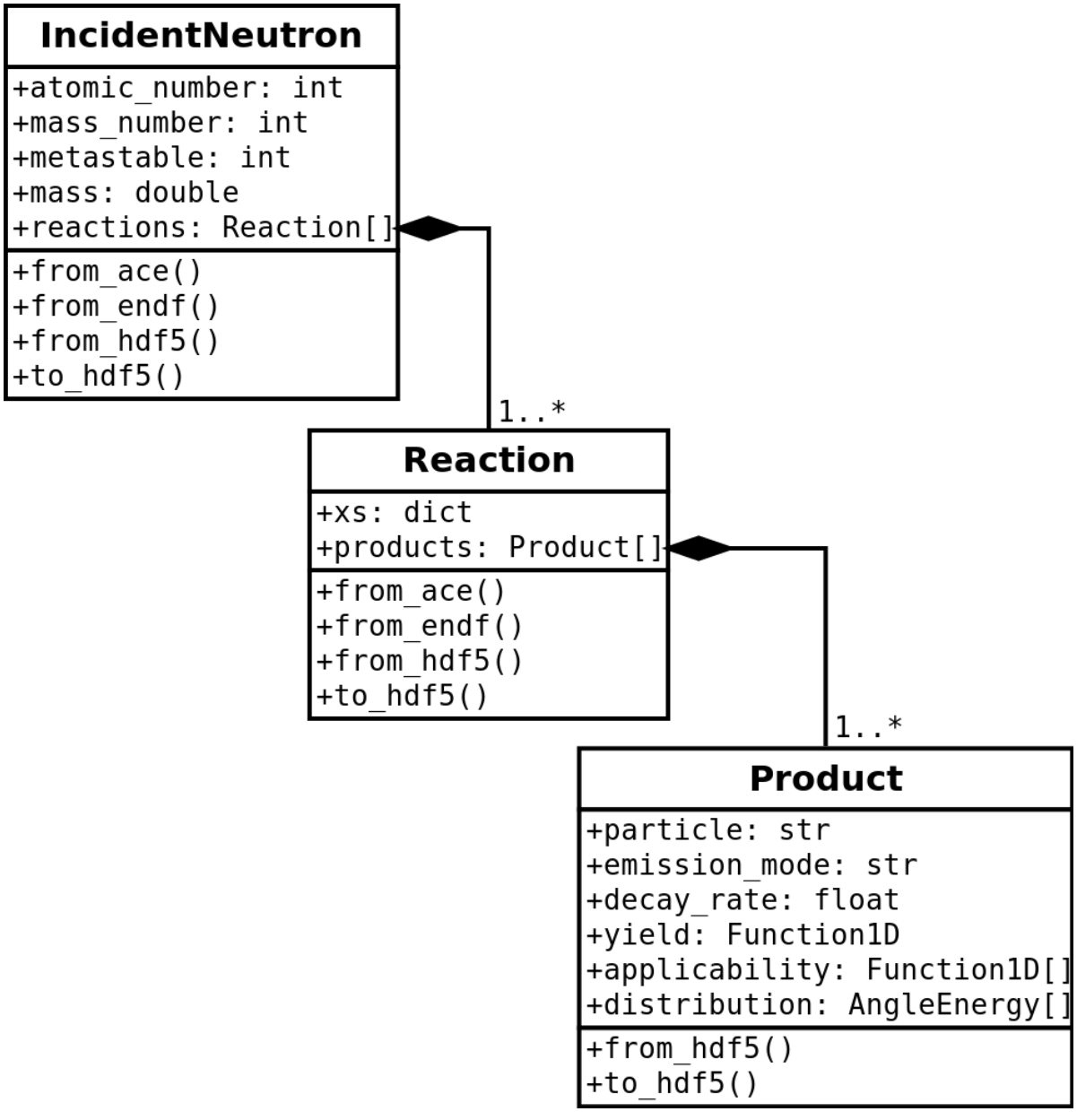
OpenMC: Python API example (Jezebel)

```
from openmc import *

# Define material
plutonium = Material()
plutonium.add_nuclide('Pu239', 3.7047e-2)
plutonium.add_nuclide('Pu240', 1.7512e-3)
plutonium.add_nuclide('Pu241', 1.1674e-4)
plutonium.add_element('Ga', 1.3752e-3)
Materials([plutonium]).export_to_xml()

# Create a sphere of plutonium
surf = Sphere(R=6.3849, boundary_type='vacuum')
main_cell = Cell(fill=plutonium, region=-surf)
main_univ = Universe(0, cells=[main_cell])
Geometry(main_univ).export_to_xml()

# Define settings for the simulation
settings = Settings()
settings.particles = 10000
settings.batches = 1000
settings.inactive = 50
center = (0., 0., 0.)
settings.source = Source(space=Point(center))
settings.export_to_xml()

# And run!
run()
```

Data extensions to Python API: `openmc.data`



Class Hierarchy



How it works

```
In [1]: from openmc.data import *

In [2]: u235 = IncidentNeutron.from_ace('U235.ace')

In [3]: for rx in u235.reactions.values():
   ...:     print(rx)
   ...:
<Reaction: MT=2 (n,elastic)>
<Reaction: MT=16 (n,2n)>
<Reaction: MT=17 (n,3n)>
<Reaction: MT=18 (n,fission)>
<Reaction: MT=37 (n,4n)>
<Reaction: MT=51 (n,n1)>
...
<Reaction: MT=4 (n,level)>

In [4]: u235.export_to_hdf5('U235.h5')
```

Temperature dependence

Differentiates between temperature-dependent:

- Cross sections
- Unresolved resonance probability tables
- $S(\alpha, \beta)$ tables

...and temperature-independent data:

- Reaction product yields
- Reaction product energy-angle distributions

Can results in large savings in memory and disk space

- JEFF 3.2 ACE files: 38 GB
- JEFF 3.2 HDF5 files: 5 GB

Temperature dependence

```
In [4]: u235_files = glob.glob('92235.71?nc')

In [5]: u235 = IncidentNeutron.from_ace(u235_files[0])

In [6]: for other_temp in u235_files[1:]:
   ...:     u235.add_temperature_from_ace(other_temp)
   ...:

In [7]: u235.reactions[18].xs
Out[7]:
{'1200K': <openmc.data.function.Tabulated1D at 0x7f312413aa90>,
 '2500K': <openmc.data.function.Tabulated1D at 0x7f312410db38>,
 '294K': <openmc.data.function.Tabulated1D at 0x7f312410da20>,
 '600K': <openmc.data.function.Tabulated1D at 0x7f31242284e0>,
 '900K': <openmc.data.function.Tabulated1D at 0x7f31241344e0>}

In [8]: u235.reactions[18].products
Out[8]:
[<Product: neutron, emission=prompt, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: neutron, emission=delayed, yield=tabulated>,
 <Product: photon, emission=prompt, yield=tabulated>]
```

Adding data beyond ACE

With HDF5, easy to extend format to include data not in ACE:

- Fission energy release
- Decay data
- Windowed multipole (for on-the-fly Doppler broadening)

```
In [2]: u235 = IncidentNeutron.from_ace('U235.ace')

In [3]: fer = FissionEnergyRelease.from_endf('U235.endf')

In [4]: u235.fission_energy = fer
```

Ongoing Work

- Adding true "processing" capabilities:
  - Resonance reconstruction (No RML yet)
  - Doppler broadening
  - Probability table generation
  - Energy deposition
- Relying on Cython for better performance
- Work to support on-the-fly Doppler broadening
- Future additions:
  - Photoatomic/photonuclear data
  - Covariances

Longer-term: Steal/use NJOY21 or Fudge

Conclusions

New `openmc.data` Python packages enables:

- Conversion of ACE → HDF5 for OpenMC transport solver
- Supplementing ACE data
- Large memory savings by not redundantly storing temperature-independent data
- Easy inspection of ACE and ENDF data

Many similarities to GND/Fudge, but not explicit short-term goal to be compatible.

Acknowledgments

- PyNE: The Nuclear Engineering Toolkit
- OpenMC team: Adam Nelson, Colin Josey
- Work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.