LA-UR-21-26524
Approved for public release; distribution is unlimited.

Title: ENDFtk: A Robust C++/Python API for Reading/Writing ENDF-formatted Data

Author(s): Gibson, Nathan Andrew
Haeck, Wim

Intended for: MCNP 2021 Symposium, 2021-07-12/2021-07-16 (Online, New Mexico, United States)

Issued: 2021-07-09

Los Alamos NATIONAL LABORATORY

ENDFtk: A Robust C++/Python API for Reading/Writing ENDF-formatted Data

Nathan A. Gibson and Wim Haeck

Tuesday, July 13, 2021

Outline

- Status and Capabilities
- Examples:
  - Read PFNS Data
  - Perturb Capture XS
  - Visualize Angular Distribution
- Future Work
- Getting `ENDFtk`

Status and Capabilities

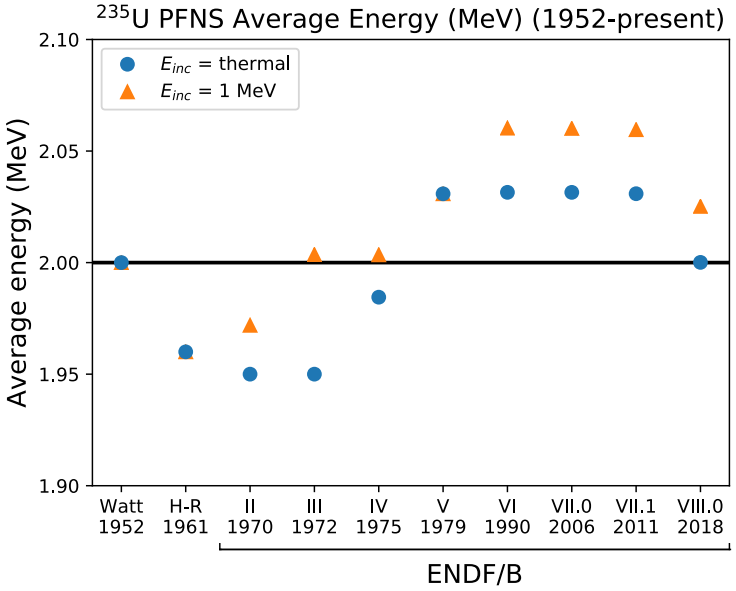
- C++ and Python API
- Fully documented API connected to Python's `help()`
- Used in NJOY and increasing number of other internal efforts
- Open source
- Latest release: v0.3.0

MF	Description	ENDFtk support	Python support	Remarks
1	General information	Full	Full	
2	Resonance parameters	Full	Full	Includes MT152 and MT153
3	Reaction cross sections	Full	Full	
4	Angular distributions	Full	Full	
5	Energy distributions	Full	Full	
6	Product energy-angle distributions	Full	Full	Includes THERMR LAW
7	Thermal neutron scattering law data	Full	Full	
8	Decay and fission product yields	Partial	Full	Only MT457 is supported
9	Multiplicities of radioactive products	Full	Full	
10	Radioactive nuclide production	Full	Full	
12	Photon production yield data	Full	Full	
13	Photon production cross sections	Full	Full	
14	Photon angular distributions	Full	Full	
15	Continuous photon energy spectra	Full	Full	
23	Photon interaction cross sections	None	None	
26	Photo-atomic distributions	None	None	
27	Atomic form factor functions	None	None	
28	Atomic relaxation data	None	None	
30	Covariance of model parameters	None	None	
31	Covariances of fission	None	None	
32	Covariances of resonance parameters	None	None	
33	Covariances of cross sections	None	None	
34	Covariances of angular distributions	None	None	
35	Covariances of energy distributions	None	None	
40	Covariances for nuclide production	None	None	

Examples

- Read PFNS Data
- Perturb Capture XS
- Visualize Angular Distribution

Example 1: Read PFNS Data



Historical question:
How has estimate of <Eout> from U-235 fission evolved through ENDF releases?

Example 1: Read PFNS Data

```
import ENDFtk

# open file, parse section
tape = ENDFtk.tree.Tape.from_file('/path/to/endf/file')
section = tape.MAT(mat).MF(5).MT(18).parse()

# switch on distribution
distribution = section.partial_distributions[0].distribution
if isinstance(distribution, ENDFtk.MF5.TabulatedSpectrum):
    incoming, outgoing = from_tabulated_spectrum(distribution)
elif isinstance(distribution, ENDFtk.MF5.MaxwellianFissionSpectrum):
    incoming, outgoing = from_maxwellian(distribution)
elif isinstance(distribution, ENDFtk.MF5.WattSpectrum):
    incoming, outgoing = from_watt_spectrum(distribution)
```

Example 1: Read PFNS Data

```
def from_maxwellian(distribution):

    # Maxwellian parameters
    energies = distribution.energies
    thetas = distribution.thetas

    # loop over incoming energies
    energy_out = []
    for incoming, theta in zip(energies, thetas):
        average_energy = 1.5 * theta
        energy_out.append(average_energy)

    return energies[:], energy_out
```

Example 2: Perturb Capture XS

Motivation:

- Demonstrate read/write
- Perturbations needed for sensitivity analyses, UQ
- Reading MF3 common visualization need
- Writing could be useful for evaluators

Task:

- Double cross section values for MF3/MT102 in Fe-56

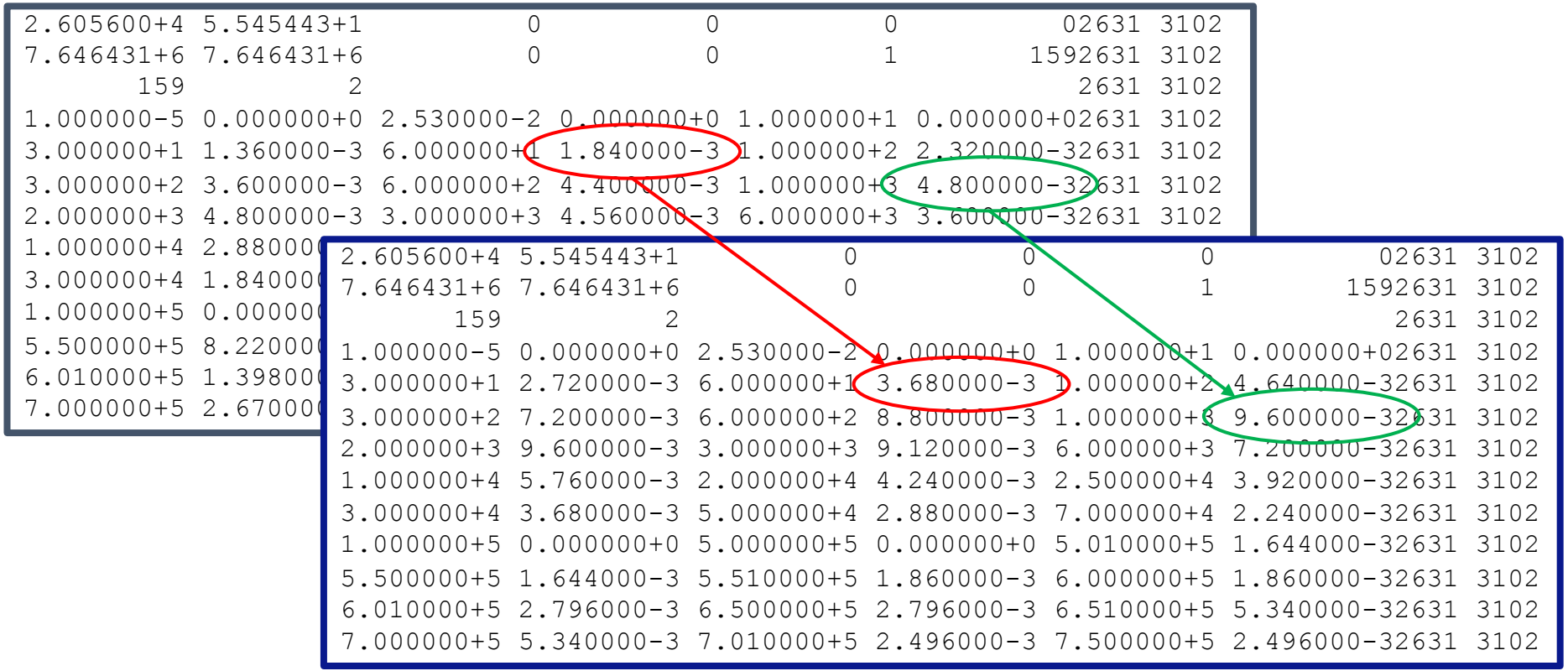
Example 2: Perturb Capture XS

```
import ENDFtk, numpy as np

# read existing tape
tape = ENDFtk.tree.Tape.from_file('fe56.endf')
section = tape.MAT(2631).MF(3).MT(102).parse()

# manipulate data
new_data = np.array(section.cross_sections) * 2
```

Example 2: Perturb Capture XS

```
# create new section
perturbed = ENDFtk.MF3.Section(
    mt=102, zaid=section.ZA,
    awr=section.AWR, qm=section.QM,
    qi=section.QI, lr=section.LR,
    boundaries=section.boundaries,
    interpolants=section.interpolants,
    energies=section.energies,
    xs=data
    )

# print to string
print(perturbed.to_string(2631, 3))
```

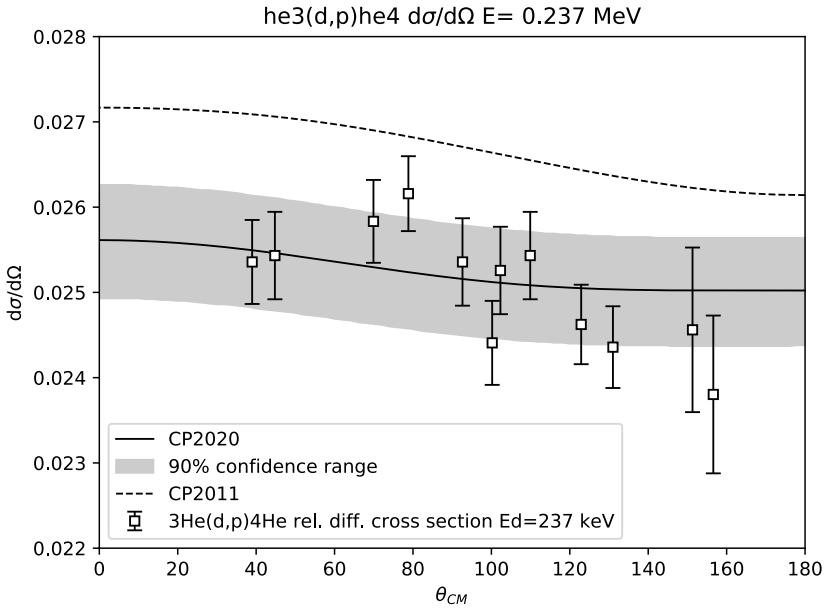
Example 2: Perturb Capture XS



```
2.605600+4 5.545443+1          0          0          0          02631 3102
7.646431+6 7.646431+6          0          0          1        1592631 3102
       159          2                                              2631 3102
1.000000-5 0.000000+0 2.530000-2 0.000000+0 1.000000+1 0.000000+02631 3102
3.000000+1 1.360000-3 6.000000+1 1.840000-3 1.000000+2 2.320000-32631 3102
3.000000+2 3.600000-3 6.000000+2 4.400000-3 1.000000+3 4.800000-32631 3102
2.000000+3 4.800000-3 3.000000+3 4.560000-3 6.000000+3 3.600000-32631 3102
1.000000+4 2.880000
3.000000+4 1.840000
1.000000+5 0.000000
5.500000+5 8.220000
6.010000+5 1.398000
7.000000+5 2.670000
```

```
2.605600+4 5.545443+1          0          0          0          02631 3102
7.646431+6 7.646431+6          0          0          1        1592631 3102
       159          2                                              2631 3102
1.000000-5 0.000000+0 2.530000-2 0.000000+0 1.000000+1 0.000000+02631 3102
3.000000+1 2.720000-3 6.000000+1 3.680000-3 1.000000+2 4.640000-32631 3102
3.000000+2 7.200000-3 6.000000+2 8.800000-3 1.000000+3 9.600000-32631 3102
2.000000+3 9.600000-3 3.000000+3 9.120000-3 6.000000+3 7.200000-32631 3102
1.000000+4 5.760000-3 2.000000+4 4.240000-3 2.500000+4 3.920000-32631 3102
3.000000+4 3.680000-3 5.000000+4 2.880000-3 7.000000+4 2.240000-32631 3102
1.000000+5 0.000000+0 5.000000+5 0.000000+0 5.010000+5 1.644000-32631 3102
5.500000+5 1.644000-3 5.510000+5 1.860000-3 6.000000+5 1.860000-32631 3102
6.010000+5 2.796000-3 6.500000+5 2.796000-3 6.510000+5 5.340000-32631 3102
7.000000+5 5.340000-3 7.010000+5 2.496000-3 7.500000+5 2.496000-32631 3102
```

Example 3: Visualize Angular Distribution



Need for visualization

- What's really in this ENDF file?
- Does it compare to application format?
- Does my updated evaluation match experimental data?

Example 3: Visualize Angular Distribution

```
import ENDFtk

# open file, parse section
tape = ENDFtk.tree.Tape.from_file('/path/to/endf/file')
section = tape.MAT(mat).MF(6).MT(600).parse()
product = section.reaction_products[0]
distribution = product.distribution

# distribution details
law = product.LAW
if law == 2:
    assert(isinstance(distribution, ENDFtk.MF6.DiscreteTwoBodyScattering))
    # ...
```

Example 3: Visualize Angular Distribution

```
import numpy as np
from numpy.polynomial.legendre import legval


# more details
data = distribution.distributions[0]
energy = distribution.incident_energies[0]

# linearize
grid = np.linspace(-1, 1, 300)
coeffs = np.array([1] + dist.coefficients[:])
coeffs = (2*np.arange(dist.NL+1) + 1) * coeffs / 2
values = legval(grid, coeffs)
```

Future Work

- Goal: Complete ENDFtk in FY21
  - Mean values: Complete MF20-28
  - Covariances: Complete MF30-40
  - NJOY formats: GENDF, ERRORR

Getting ENDFtk

Repository, installation instructions:
https://github.com/njoy/ENDFtk

Contact us:

njoy@lanl.gov