LLNL processing codes and APIs for GNDS

Cross Section Evaluation Working Group, Nov. 2018

Bret Beck

Lawrence Livermore National Laboratory

LLNL-PRES-761055

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Collaborators

Caleb M. Mattoon,
Gerry Hedstrom,
Dave Brown (BNL),
Eric D. Jurgenson,
Mare-Anne Descalle,
Teresa Bailey and the Ardra team
Scott Mckinley and the Mercury team

Many more!

Outline

- GNDS
- FUDGE
- LLNL GNDS C++ APIs
- Testing
- Conclusion

GNDS

GNDS design

- GNDS is very hierarchical.
- Defines structure and not format
  - Data can be stored in a file using XML, HDF5, mixture of XML/HDF5, etc.
- Structure follows physics.
- Stores data for one PROTARE (PROjectile, TARget, Evaluation)
  - e.g., n+U236 for ENDF/B-VIII.0
- Supports simultaneous storage of
  - Evaluated, and processed Monte Carlo and multi-group data.
  - multi-temperature data
  - Enabled via styles and component/forms

Outline of a GNDS file

```
<reactionSuite projectile="n" target="Be9" evaluation="ENDF/B-7.1" format="1.9" projectileFrame="lab">
 + <styles></styles>
 + <externalFiles></externalFiles>
 + <documentations></documentations>
 + <PoPs name="protare_internal" version="1.0" format="0.1"></PoPs>
 + <resonances></resonances>
 – <reactions>
    + <reaction label="n + Be9" ENDF_MT="2"></reaction>
    + <reaction label="2n + 2He4" ENDF_MT="16"></reaction>
    + <reaction label="H1 + Li9" ENDF_MT="600"></reaction>
    + <reaction label="H2 + Li8" ENDF_MT="650"></reaction>
    + <reaction label="H3 + Li7" ENDF_MT="700"></reaction>
    + <reaction label="H3 + (Li7_e1 -> Li7 + photon)" ENDF_MT="701"></reaction>
    + <reaction label="He4 + He6" ENDF_MT="800"></reaction>
    + <reaction label="Be10 + photon" ENDF_MT="102"></reaction>
   </reactions>
 + <sums></sums>
 + <applicationData></applicationData>
</reactionSuite>
```

Outline of a GNDS reaction

- Every product can contain processed data
  - E.g., multi-group product (transfer) matrix in the <distribution> node.

```
– <reaction label="H1 + Li9" ENDF_MT="600">
  + <crossSection></crossSection>
  – <outputChannel genre="twoBody">
    + <Q></Q>
    – <products>
      – <product pid="H1" label="H1">
        + <multiplicity></multiplicity>
        + <distribution></distribution>
        </product>
      – <product pid="Li9" label="Li9">
        + <multiplicity></multiplicity>
        + <distribution></distribution>
        </product>
      </products>
    </outputChannel>
  </reaction>
```

FUDGE

FUDGE

- FUDGE
  - Originally named “For Updating Data and Generating ENDL”
  - Changed to “For Updating Data and Generating Evaluations”
    - Also handles processing and processed data
- Started around 2002 to manage, manipulate, view and process ENDL data
- Updated to translate ENDL and ENDF to GNDS and GNDS to ENDF
- Updating to manage, manipulate, view, check and process GNDS data
- Download fudge via https://ndclx4.bnl.gov/gf/project/gnd/
  - Now includes all deterministic processing
  - Python 2.7
  - Next release should work with Python 3.6+

Translation of ENDF/ENDL to GNDS

- ENDL translation complete

- FUDGE translates all of the following ENDF sub-libraries

neutrons	protons	deuterons	tritons
helium3s	alphas	gammas	photoat
standards	electrons	decay	atomic_relax
thermal_scatt	nfy	sfy	

- FUDGE handles all properly formatted ENDF-6 formatted files except for the new fission stuff
  - ENDF/B-VII.1, VIII.0

- The following sub-libraries can be processed

neutrons	protons	deuterons	tritons	helium3s
alphas	gammas	photoat	standards	

FUDGE processing example for transport codes

- FUDGE command to process a GNDS file for deterministic and Monte Carlo transport at 3 temperatures

```
bin/processProtare.py --energyUnit MeV -mc -mg -t 2.585e-08 -t 1e-07 -t 1e-4 gnds.file.xml
```

  - Default temperature unit is MeV/K (--temperatureUnit)
  - processProtare.py can be user modified

- File contains
  - evaluation data
  - Reconstructed cross sections (if needed)
  - Coulomb + nuclear elastic μ cutoff (if needed)
  - average product data
  - pdf/cdf data for distributions (e.g., P(E'|E))
  - heated cross sections
  - Common grid heated cross section (for Monte Carlo)
  - multi-group data

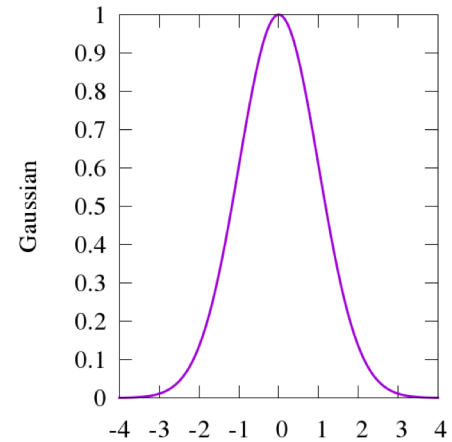
File can be large. We have some with 23 temperatures.

What is missing from FUDGE

- Improvements needed for transport
  - Unresolved resonance treatment
    - Unresolved resonance probability tables
    - We will calculate and store P(σ). More on next slide
  - Neutron thermal scattering law data
    - Need to work with Ayman Hawari and his students
  - Multi-band?

FUDGE: Unresolved resonance treatment

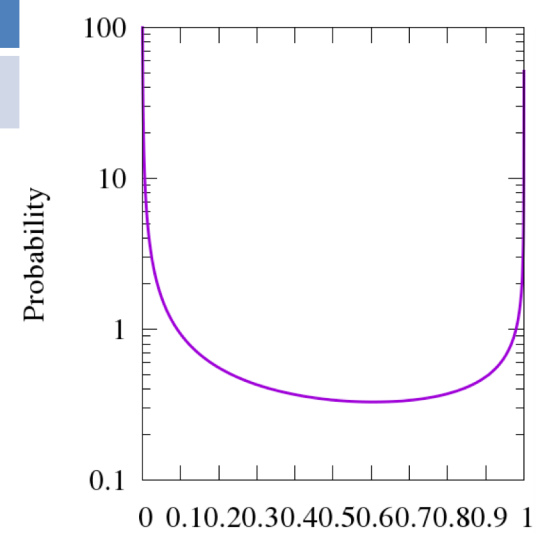
- Currently in development in FUDGE



- Steps:
  1. Generate a realization of the cross section $\sigma(E)$ around an energy point in the GNDS file. Developers are Dave Brown and Caleb Mattoon.

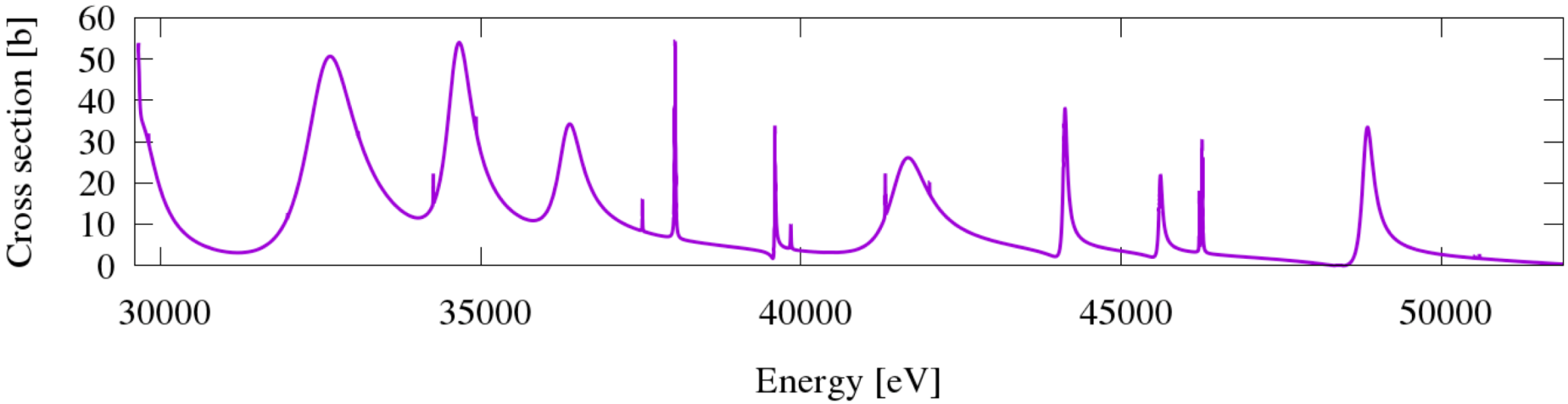
energy	Level spacing	Neutron width	Capture width
0.038	3.84654e-3	2.38392e-7	1.90487e-6

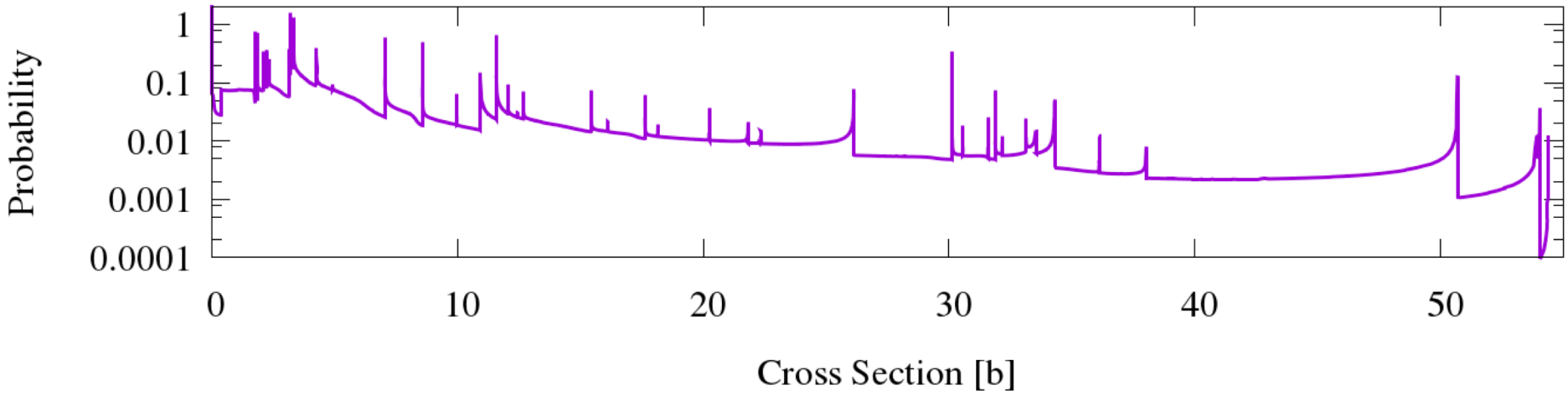
  2. Calculate $P(\sigma)$ from the $\sigma(E)$
     — FUDGE's XYs1d class has a pdfOfY method
  3. Repeat steps 1 and 2 N times and average $P(\sigma)$



- Store average $P(\sigma)$ in GNDS file
  - Still to be defined.

Unresolved resonance treatment: Fe55 elastic example





FUDGE support for other "processed" formats

- We have developed a GNDS to ACE converter
  - Steps
    - If data are from an ENDF-6 file, convert to GNDS

```
endf2gnds.py U235.endf U235.xml
```

    - Use FUDGE to process for Monte Carlo transport, writes back to GNDS

```
processProtare.py -mc U235.xml U235.mc.xml
```

    - Run toACE.py script to convert processed data into an ACE file

```
toACE.py –i 90 U235.mc.xml U235.ace
```

LLNL transport APIs and codes

PoPs

- PoPs: Properties of Particles
  - PoPs is a structure defined by SG38 for storing particle data
    - Mass, spin, halflife, etc.
  - Allows for aliasing (e.g, ‘98235’ for ‘U235’)
  - Meta-stables (e.g., ‘Am242_m1’)
  - Can be a stand alone database (file) and/or included in a GNDS file

- PoPsCpp
  - C++ API to read PoPs data

We will release PoPsCpp after it is put through LLNL’s review and release process.

GIDI

- GIDI: General Interaction Data Interface
  - C++ API to read GNDS files for transport codes
  - Can get data at any level in GNDS structure
  - Multi-group collapsing
    - For vectors and matrices: as Vector and Matrix classes
    - Transport correction
  - Calculates multi-group energy deposition
  - Complete, including photon-atomic, photon-nuclear and charged particle
    - Excluding unresolved resonance region probability table, neutron thermal scattering law and multi-band data

We will release GIDI after it is put through LLNL's review and release process.

MCGIDI

- MCGIDI: Monte Carlo GIDI
  - C++ API to store and sample for Monte Carlo transport codes
  - Uses GIDI to read data, then puts it into better form for MC needs

```
protareMC = new MCGIDI::Protare( *protare, MC, domainHash, temperatures );
```

  - Handles point-wise cross sections and pdf/cdf distributions
  - Working on
    - point-wise energy deposition
    - multi-group and fixed-grid support for cross sections, deposition energy, etc.

We will release MCGIDI after it is put through LLNL's review and release process.

Testing in LLNL transport codes

- LLNL transport codes have implemented GIDI/MCGIDI
  - Ardra (deterministic) and
  - Mercury (MC) transport codes

- LLNL's V&V test suite
  - Uses Ardra and Mercury
    - Also MCNP
  - Criticality assemblies, time-of-flights and reaction ratios
  - Can compare to benchmarks or other codes/datasets
    - ENDF/B-VII.1 and ENDF/B-VIII.0
    - LLNL's ENDL

Conclusion

Summary

- GNDS
  - New international “format” for storing nuclear evaluated and processed data.
- FUDGE
  - LLNL’s nuclear data management package.
  - Released
    - Download fudge via https://ndclx4.bnl.gov/gf/project/gnd/
    - Python 3.6 version coming in next release
- LLNL GNDS APIs and transport codes
  - APIs, PoPsCpp, GIDI and MCGIDI
  - Transport codes using APIs
    - Ardra: Deterministic transport code
    - Mercury: Monte Carlo transport code
  - All have been tested using LLNL’s V&V test suite
  - We will release PoPsCpp, GIDI and MCGIDI

Current and future work

- Unresolved resonance probability tables
- Neutron thermal scattering law
- Multi-band?

Other needed formats

- Stuff to make sharing easy
  - Way to define a “library” (i.e., a collection of protares)
    - We are doing this at LLNL with a “map” file (see below).
  - Define a common multi-group boundaries structure
  - Define a common flux structure
  - ?

```
<map>
<import path="neutrons/all.map"/>
<import path="protons/all.map"/>
<import path="photo-nuclear/all.map"/>
<import path="photo-atomic/all.map"/>
</map>
```

```
<map>
<protare evaluation="ENDF/B-7.1" projectile="n”
            target="O16" path="n-008_O_016.xml"/>
<protare evaluation="ENDF/B-7.1" projectile="n”
            target="Fe56" path="n-026_Fe_056.xml"/>
<protare evaluation="ENDF/B-7.1" projectile="n”
            target="Th227" path="n-090_Th_227.xml"/>
</map>
```

Job posting

- A job posting to work on FUDGE and APIs will be listed shortly

  http://careers-llnl.ttcportals.com/jobs/search

  - Of web search for “llnl careers”

- Please apply if interested

- Please let other know about posting