WPEC SG38 and GND: Designing a New Format for Storing Nuclear Data

Presented at P(ND)²
October 16, 2014

Caleb Mattoon
International collaboration (WPEC SG38) has been formed to design a new format for storing nuclear data, and to oversee the transition from ENDF-6 to the new format.

- Contents of this presentation:
  - Why is change needed?
  - How will it impact nuclear data evaluators and users?
  - What about infrastructure (processing, plotting etc.)?
  - Progress so far, and future work... and predictions!
Evaluated Nuclear Data condenses nuclear reaction theory and experimental data together into a set of recommended values.

- Includes cross sections, product multiplicities and distributions, covariances, etc.
- Stored in standard computer-readable form
  - Evaluated Nuclear Data Format (ENDF) has been that standard for nearly 50 years.
Why is change needed?
Reason #1: technical issues with ENDF-6.

- Hard precision limit: 11 digits including '.', '+', '-', etc.

- Format is difficult to extend. The three digits allotted for reaction ids (MT numbers) are nearly used up.

- No standard way to store processed data, making code comparisons more difficult.

- My least-favorite line in the ENDF-6 format manual:
  - Section 6.2: “LIP: Product modifier flag. Its main use is to identify the isomeric state of a product nucleus… The exact meaning assigned to LIP should be explained in the File 1, MT=451 comments.”
Lawrence Livermore National Laboratory

Reason #2: ENDF-6 is hard to read. Bugs can and do lurk inside ENDF files undetected!

- ENDF-6 strongly influenced by punch-card origins:

  ```plaintext
  ... 7.418400+4 1.823710+2  0  3  82  07437 6 5  1
  1.000000+0 1.000000+0  0  1  1   457437 6 5  2
  45  2   7437 6 5  3
  1.000000-5 0.000000+0 6.000000+6 0.000000+0 6.500000+6 9.700390-17437 6 5  4
  ... 1.150000+8 6.942060+0 1.300000+8 7.292060+0 1.500000+8 7.686070+07437 6 5  18
  0.000000+0 0.000000+0  1  2  1   457437 6 5  19
  45  22   7437 6 5  20
  0.000000+0 1.000000-5  0  0  4   27437 6 5  21
  0.000000+0 2.000000+5 1.000000-5 0.000000+0   7437 6 5  22
  0.000000+0 6.000000+6  0  0  4   27437 6 5  23
  0.000000+0 2.000000+5 1.000000-5 0.000000+0   7437 6 5  24
  ... (Example is W184 from ENDF-VII.1, ~105K lines total)
  ```

- Not human-readable!

- Tools like FIZCON and PSYCHE are able to read and check ENDF files, yet errors continue to appear in evaluated libraries.
Reason #3: we need to recruit new talent to replace retiring nuclear data experts!

- New generation prefer to use modern software tools and practices.
  - Just continuing to use old format + old infrastructure will *not* ensure reliable nuclear data.
  - Instead, adapt to new methods... while comparing rigorously to old codes for quality assurance!
Goal: design structure that builds on strengths of ENDF-6 but is more flexible, human readable and compatible with modern tools!

- LLNL already has a candidate: Generalized Nuclear Data or GND.
- LLNL can’t be isolated: to replace ENDF, GND must become an international standard, allow easy data exchange anywhere!
- Thus WPEC SG38 is leading the effort to turn GND into the new standard.
GND organizes data in a hierarchy. Most data are stored inside ‘reaction’ elements:

```xml
<reaction label="..." date="..." ENDF_MT="...">
  <crossSection nativeData="...">
    <!-- options: linear, piecewise, resonancesWithBackground, ... -->
    <outputChannel name="...">
      <product name="..." label="...">
        <distributions>...</distributions>
        <multiplicity>...</multiplicity></product>
      </product>
    </outputChannel>
  </crossSection>
</reaction>
```
Reactions involving same target/projectile are collected together in a ‘reactionSuite’:

<reactionSuite projectile="..." target="..." version="..." temperature="...">
  <styles> e.g. ‘evaluated’ or ‘processed’ </styles>
  <documentations> support ascii, html, etc. </documentations>
  <particles> ground state, levels, gammas ... </particles>
  <resonances> resolved and/or unresolved ... </resonances>
  <reaction>...</reaction>
  <reaction>...</reaction>
  <reaction>...</reaction>
  ...
</reactionSuite>
Make use of general-purpose low-level data containers

- Build on one of the strengths of ENDF-6: basic, flexible containers like ‘TAB1’ and ‘TAB2’
  - Capable of storing many different types of data
  - Support multi-dimensional data including interpolation rule(s)
- New containers should also support axis labels, units, etc.

```xml
<crossSection nativeData="linear">
  <linear xData="XYs" length="123" accuracy="0.001" interpolation="linear,linear">
    <axes>
      <axis index="0" label="energy_in" unit="eV"/>
      <axis index="1" label="crossSection" unit="b"/></axes>
    <data> 7894.169 0 8e3 0.03555347 9e3 0.2396638 ... </data></linear></crossSection>
```
GND-formatted data already accessible through data centers:

- ENDF-VII.1 available in GND from the NNDC:

- Multiple libraries also available from IAEA NDS:
  - [https://www-nds.iaea.org/exfor/servlet/E4sSearch2](https://www-nds.iaea.org/exfor/servlet/E4sSearch2)
  - To access, choose target/projectile, submit a search and then select the ‘extended’ button on results page
What will be the impact on nuclear data users?

- In the short term, not much impact: backwards-compatibility with ENDF-6 is an SG38 requirement
  - Support translating back to ENDF-6, or exporting directly to processed formats like ACE, NDI, etc.

- In longer term, applications codes should access data directly from new format.
  - Will facilitate exchanging data, comparing codes
Major goal for SG38: Support backwards compatibility with ENDF-6 as long as possible

- ENDF-6 and new format will need to co-exist for several years. This means:
  - Support translation back and forth.
  - Extend codes that currently handle ENDF to also handle the new format.
  - Lots of testing and comparison!
Major goal for SG38: Support backwards compatibility with ENDF-6 as long as possible

- ENDF-6 and new format will need to co-exist for several years. This means:
  - Support translation back and forth.
  - Extend codes that currently handle ENDF to also handle the new format.
  - Lots of testing and comparison!
Longer term goal: Define Application Programming Interfaces (APIs) for reading and writing data in the new format.

- Using an API:
  - provides a standard way to access data, and
  - protects applications from future changes in the format.

[Diagram showing the flow from ENDF-6 to new/updated processing codes, then to legacy processing codes, and finally to processed formats (ACE, NDI, etc.) and applications, with arrows indicating translation and new API usage.]
What about infrastructure? How can we replace all the codes that currently use ENDF-6?

- LLNL’s FUDGE is a good start:
  - For Updating Data and Generating an Evaluation
  - Supports
    - Reading/Writing GND data
    - Conversion of ENDF to GND and vise-versa
    - Visualization of data – e.g., plotting and printing
    - Manipulation of data
    - Checking of data
    - Processing of data
      - Resonances reconstruction
      - Cross section heating
      - Grouping of data (mainly for Sn as grouping is done on recast in GIDI for MC)
      - Sn transfer matrices
Translating ENDF↔GND

> python rePrint.py ENDF-B-VII.1/neutrons/n-001_H_001.endf

2 [3, 4, 33] : MF=4, LTT = 1
102 [3, 6, 33] : MF=6 : ZAP=0, LAW=2, LANG=0 : ZAP=1002, LAW=4
1 [3, 33]
  processing MT1 gammas

n + H1 -->
  n + H1
    n: label = n: to ENDF6:
    H1: label = H1: to ENDF6:
  H2 + gamma
    gamma: label = gamma: to ENDF6:
    H2: label = H2: to ENDF6:

  summed reaction: total

> ls -lh
  37K  test.endf6.xml
  4.9K  test.endf6-covar.xml
  50K  test.endf6.noLineNumbers
  50K  test.endf6.orig.noLineNumbers
  50K  test.endf6.orig.noLineNumbers.cleanAndFixed

resulting files after translation

ENDF to GND
GND to ENDF
Status of ENDF translation to GND

- For ENDF-VII.1 sub-libraries:
  - Can translate:
    - neutrons/ protons/ deuterons/ tritons/ helium3s/ gammas/
      standards/ electrons/ photoat/ atomic_relax/
  - Evaluations “H1 + H2” and “H2 + H3” have bad data
  - Can be translated into GND-like format, these are not yet integrated with FUDGE:
    - nfy/ sfy/ thermal_scatt/
  - Currently not supported:
    - Decay/
Status of translation for other libraries (incident neutrons only):

- **JEFF-3.1.2**
  - 328 / 381 successfully translated
  - 371 / 381 with the ‘skipBadData’ option
- **JENDL-4**
  - 385 / 406 successfully translated
  - 406 / 406 with skipBadData
- **We are submitting bug reports to library maintainers**
  - Fudge is now part of pre-release testing for TENDL
Processing codes are changing with computer architecture. Processing includes:

- Fast data transformations, can happen when data are read in by user code:
  - Grouping cross sections, flux weighting
  - Converting pdf → cdf

- More computer-intensive, done off-line:
  - Reconstructing cross section from resonances
  - Doppler broadening
  - Generating transfer matrices

- Processed data should be stored in GND (facilitate sharing data)
Transfer matrices: outgoing angle/energy spectra averaged over incident energy bins

Angular dependence is in Legendre expansion
Transfer matrices: outgoing angle/energy spectra averaged over incident energy bins

Angular dependence is in Legendre expansion
Processing codes don’t always agree, especially for double-differential distributions:
Processing codes don’t always agree, especially for double-differential distributions:
Processing codes don’t always agree, especially for double-differential distributions:
GND infrastructure is freely available. More testing and feedback are needed!

- GND 1.3 plus supporting infrastructure now available for download:

- The package includes:
  - Translation tool ‘rePrint.py’ to move ENDF-6 data to and from GND
  - Infrastructure for reading, writing, plotting, physics testing and basic processing of GND files.

- Please test it out and give us feedback!
GIDI (General Interaction Data Interface)

- An API for GND we are developing
  - Deterministic GND data
    - Reading
    - Collapsing
  - Monte Carlo GND data
    - Reading data
    - Sampling data

- Beta version currently in GEANT4

- Being tested in LLNL stand-alone Monte Carlo code Mercury
  - Testing with critical assemblies and ‘broomsticks’
Future work:
SG38 still working to finalize specifications for how data will be stored in GND.

- SG38 plans to create ‘final’ draft of specifications for the new format by December 2015.

- GND and Fudge will be modified as specifications evolve.

- Ongoing work: extending the API, testing and QA, assisting transition away from ENDF-6
Main perspective for the next decade:

- Libraries officially released in both ENDF-6 and in the new format
- Formats will coexist, but new evaluations will begin to add data types not supported by ENDF. Possibly new sub-libraries only available in GND?
- Existing codes gradually extended to handle GND, new codes use it directly