FUDGE and GNDS: an overview

IAEA Consultancy Meeting on Model Code Output and Nuclear Data Form Structure

C. M. Mattoon, G. Gert

March 15, 2021

LLNL-PRES-820740

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

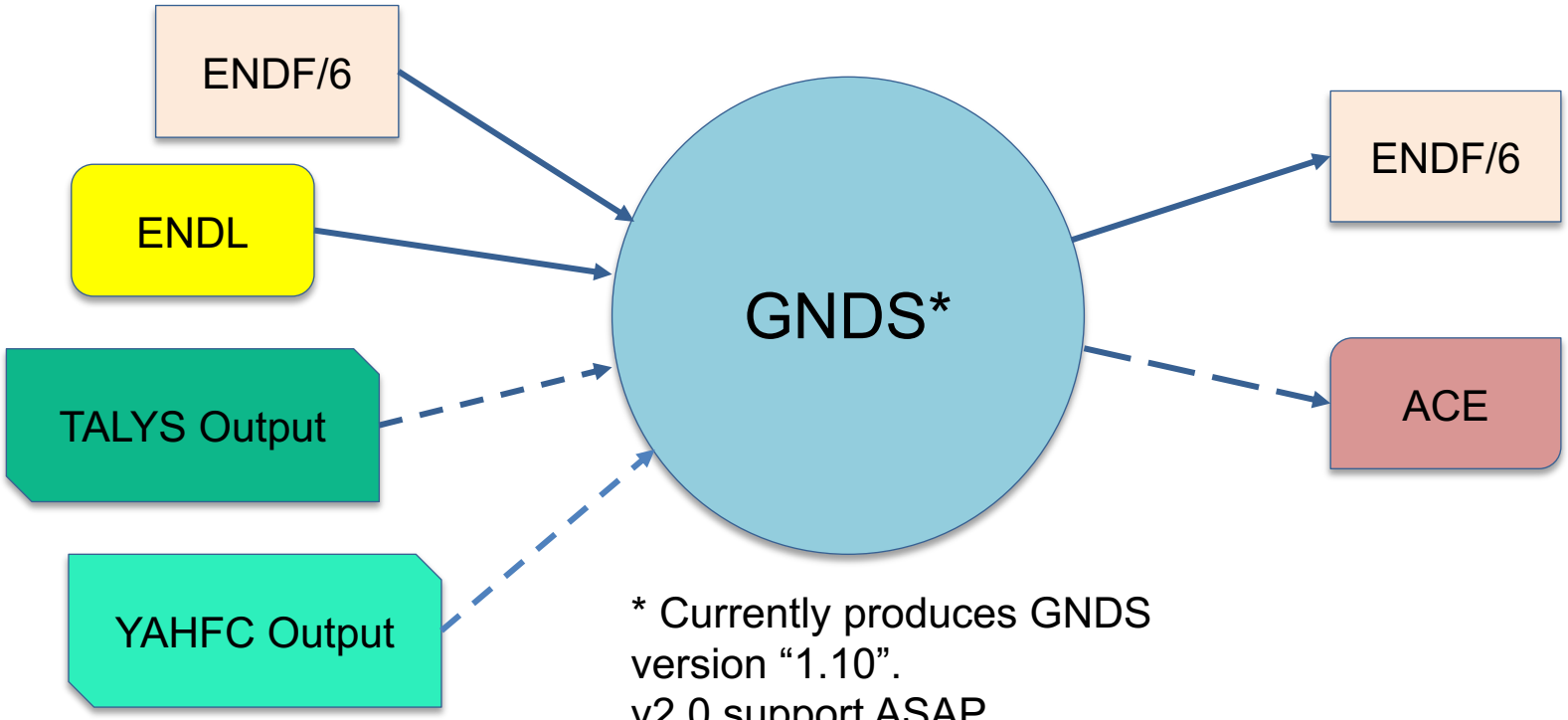
Lawrence Livermore National Laboratory

FUDGE and GNDS form the center of a renewed focus on nuclear data at LLNL

- Introduction
- Generating GNDS by translation or directly
  - Translation helps uncover errors in older formats
  - TAGNDS: translate TALYS output to GNDS
    - With new resonance parameter capability
- Physics checking capabilities in FUDGE
- Processing GNDS
  - processProtare, main driver for processing
  - Recent additions
  - Exporting processed data
- Visualization
- FUDGE is available and open-source!
  - https://github.com/LLNL/fudge

1) Generating evaluations in GNDS format

- FUDGE supports translating to and from GNDS, either directly or as a library used within other codes



Translating libraries into GNDS

Translation must preserve original physics content/meaning

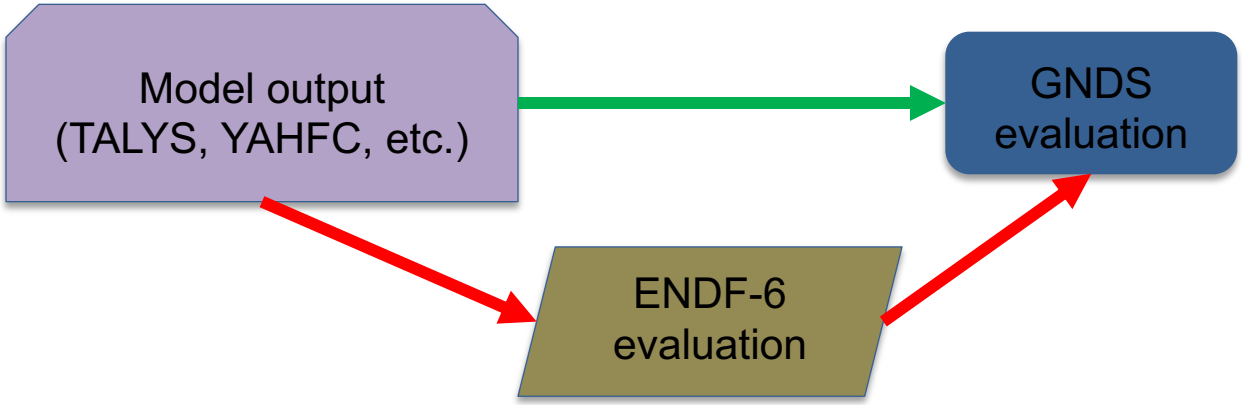
- Translating ENDF-6 files with FUDGE:

```
> python ~/fudge/bin/rePrint.py ~/ENDF-VIII.0/neutrons/n-029_Cu_063.endf
> ls *.xml
test.endf6-covar.xml  test.endf6.xml                          # GNDS files produced by translator
> ls *.noLineNumbers
test.endf6.noLineNumbers  test.endf6.orig.noLineNumbers       # ENDF-6 files, one produced by translating
                                                              # GNDS back to ENDF-6. Compare the two files
                                                              # to test fidelity of translation
```

- Some evaluations cannot be translated yet, however. Common causes of translation errors:
  - Bad data in the original evaluation. For example,
    - cross section values not sorted in ascending order
    - inconsistencies between MF2 and MF32
  - Infrequently-used ENDF-6 options that are not yet supported by the translator. For example,
    - Adler-Adler resonance parameters
    - Isotope-specific resonance parameters listed in an elemental evaluation

Translation is important, but we also need ability to directly generate GNDS

- Avoids limitations inherent in ENDF/B format
- Requires some initial effort, but eventually the direct route to GNDS should be simpler and more maintainable



- Thanks to IAEA support, we now have a TALYS-GNDS translator
  - **Translator TAGNDS** reads TALYS output into GNDS classes in FUDGE, then serializes the result to XML

Quick guide to using TAGNDS:

- Install FUDGE
- Add FUDGE and TAGNDS to the PYTHONPATH
- Run TALYS on an input file
  - Input should enable options ‘endf’, ‘outSpectra’, ‘outLegendre’, etc.
  - Redirect output to a file called ‘output’ in the same directory, i.e. talys < input > output
- Use generateGNDS.py to translate results into GNDS ‘reactionSuite’
  - Future work: finish TARES integration, support other parts of the ‘T6’ suite

Similar effort is underway to generate GNDS evaluations using YAHFC

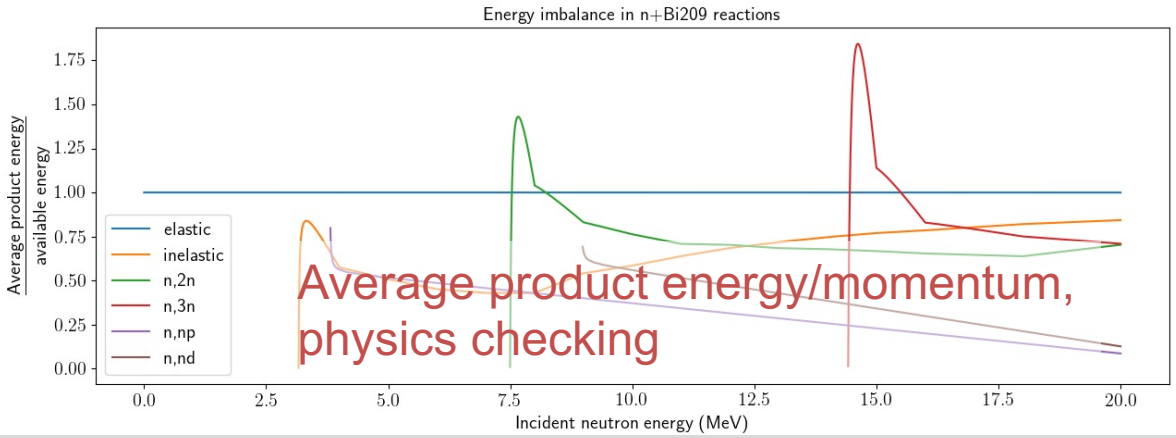
- YAHFC: Yet Another Hauser Feshbach Code, written at LLNL by E. Ormand

- GNDS translator developed and maintained by Ian Thompson, now being tested and improved for generating new evaluations

2) Once we have GNDS evaluations, FUDGE supports physics checks to improve quality

- Checks include
  - resonance statistical properties, Q-value/threshold mismatch, ZA mismatch, domain mismatch (e.g. between cross section and multiplicity), checks for poor interpolation choices (e.g. 'flat' interpolation along incident energy), energy imbalance, etc.

- FUDGE checking is integrated into the NNDC ADVANCE system
  - But often produces LOTS of warnings. **Needed**: severity levels to help evaluators focus on high-priority problems

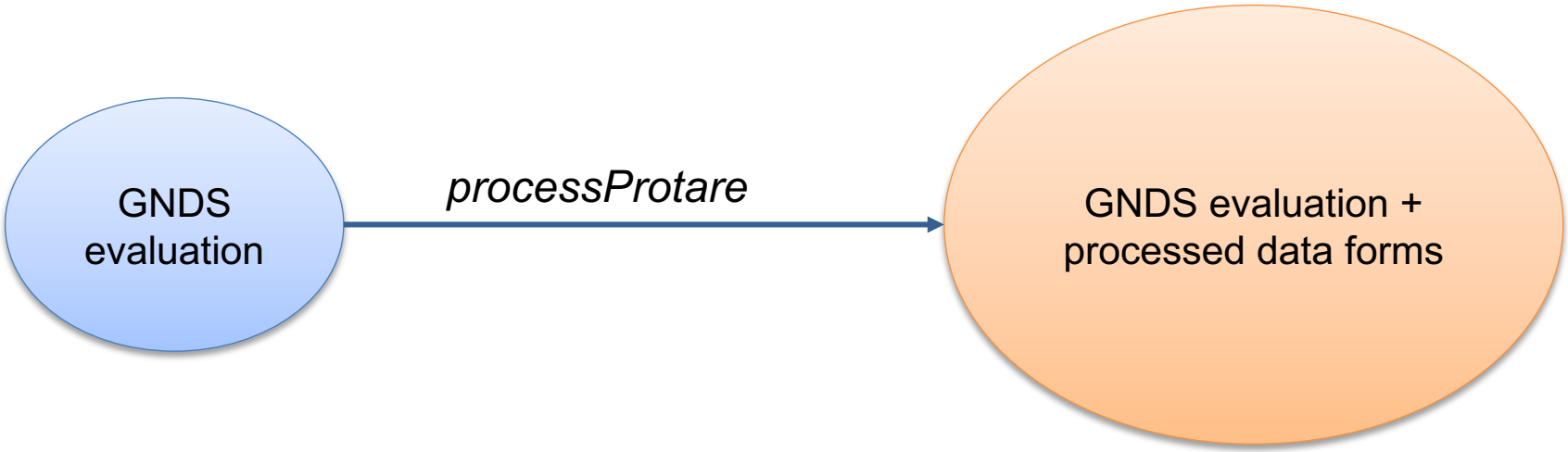


Examples of issues caught by FUDGE .check()

- W-180 in ENDF-VIII.0
  - Negative probabilities in inelastic neutron angular distributions (Legendre series truncated too soon)
  - Energy does not balance for most N-body reactions. Problem is usually worse close to threshold, gets better but not perfect at higher incident energies
  - Tabulated Q-value does not agree with value calculated from masses
    - Limited ENDF-6 precision partly to blame here

3) FUDGE supports processing GNDS files for transport applications

- processProtare.py: main driver for processing for both Monte Carlo and deterministic (multi-group) data



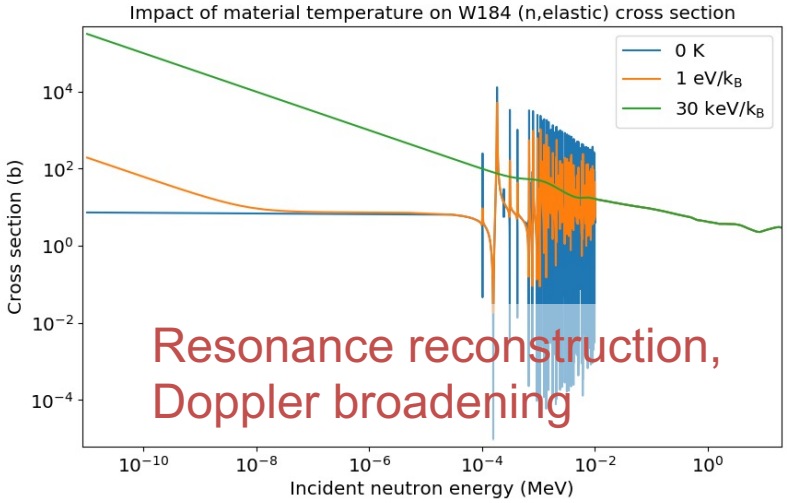
processProtare: processing Monte Carlo and/or multigroup data

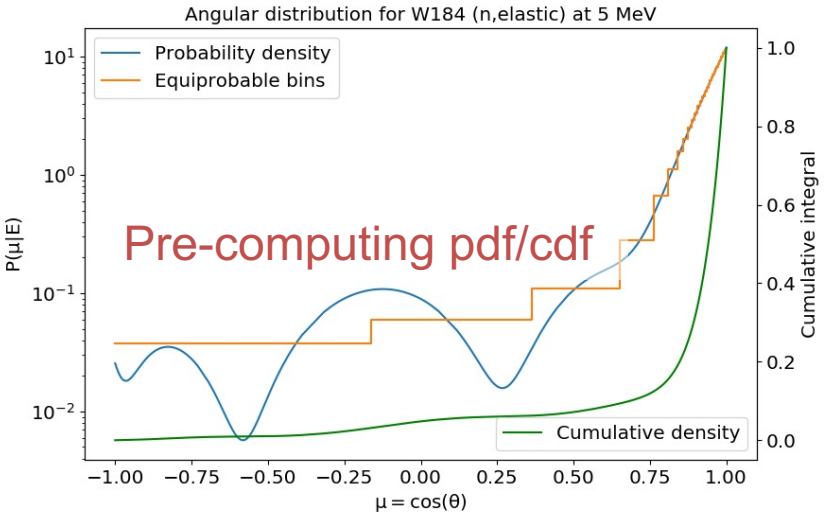
```
>python bin/processProtare.py n-001_H_001.endf.gnds.xml -mc -mg -up \
     --groupFile groups.xml --fluxFile fluxes.xml –g n=LLNL_gid_7 \
     -t 300 -t 600 --temperatureUnit K -vvv
```

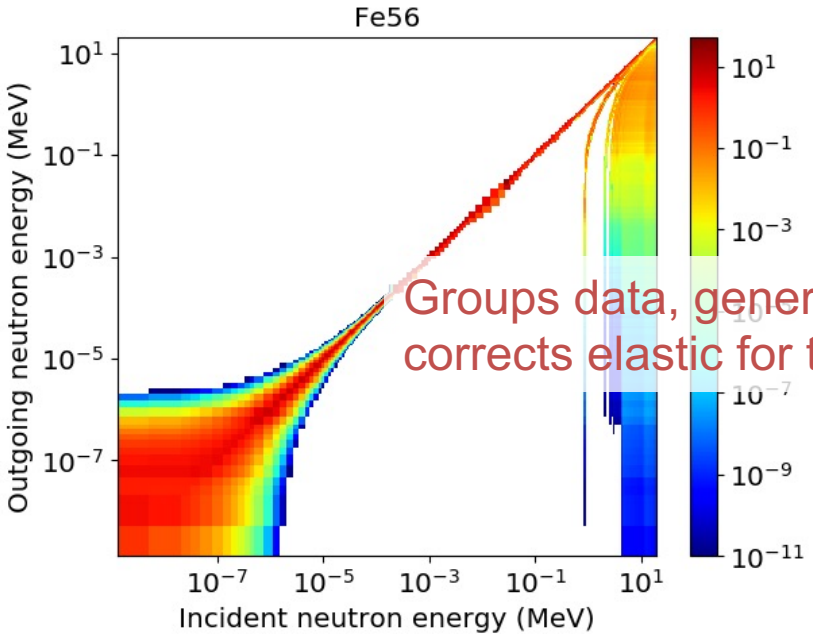
- Result contains both Monte Carlo and deterministic data at two temperatures (300 and 600 K).
- All processed data go into a new GNDS file. GNDS 'styles' help differentiate different types of processed data:

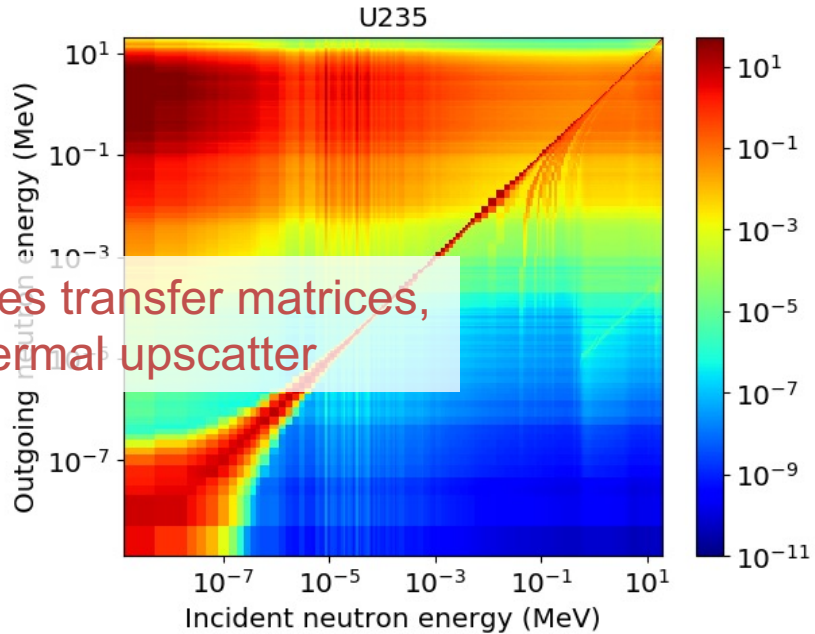
```
<reaction label="n + H1" ENDF_MT="2">
  <crossSection>
    <XYs1d label="eval">...</XYs1d>
    <XYs1d label="heated_000">...</XYs1d>
    <Ys1d label="MonteCarlo_000">...</Ys1d>
    <gridded1d label="MultiGroup_000">...</gridded1d>
    <XYs1d label="heated_001">...</XYs1d>
    <Ys1d label="MonteCarlo_001">...</Ys1d>
    <gridded1d label="MultiGroup_001">...</gridded1d>
  </crossSection>
  <outputChannel genre="twoBody">
    ...
```

FUDGE manages and processes GNDS data for Monte Carlo and deterministic transport







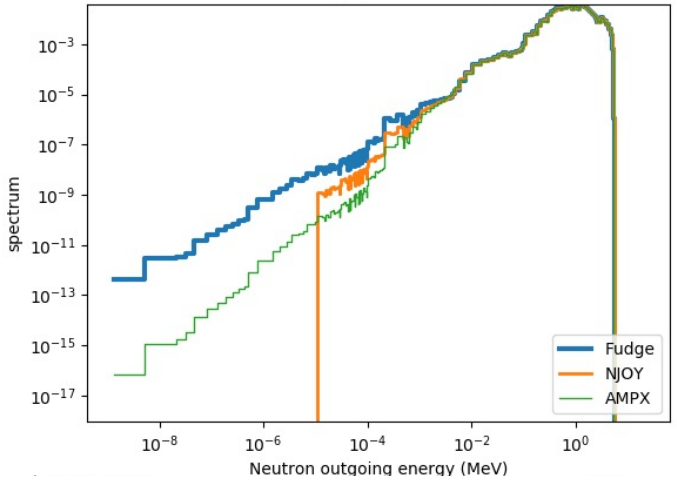


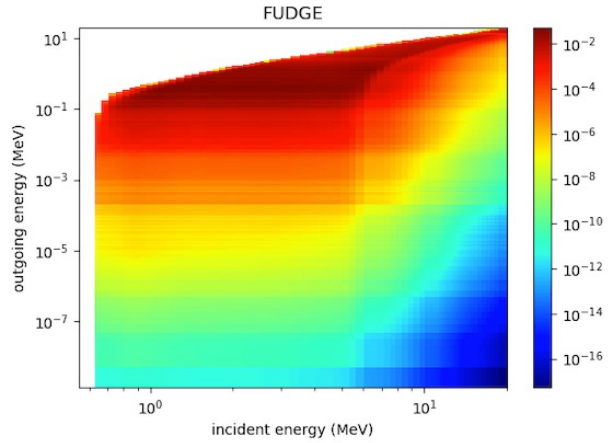
Groups data, generates transfer matrices,
corrects elastic for thermal upscatter

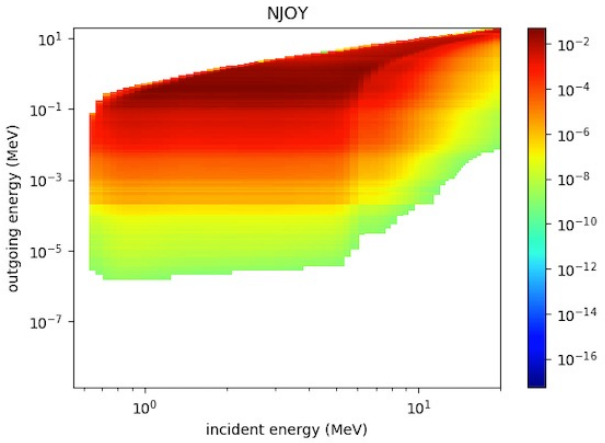
Ongoing effort: compare processing results to AMPX, NJOY, PREPRO, etc.

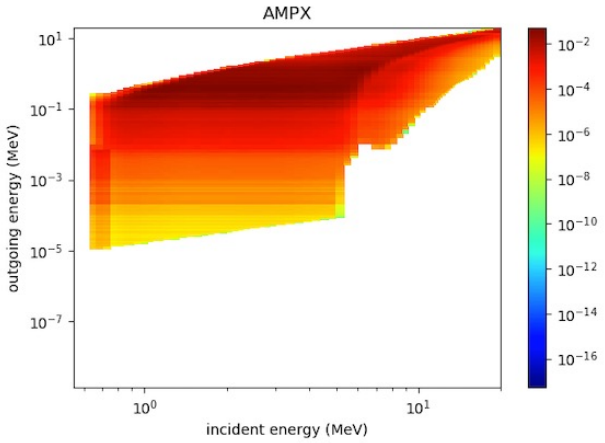
- Different treatment of interpolation is a frequent cause of discrepancies

Grouped outgoing (n,n') spectrum (MT91) from ENDF-VII.1 Pu239









Interpolation rules along higher dimensions are a potential trap!

- ENDF-6 and GNDS both support special rules for interpolation along higher dimensions, but many evaluations still use INT=2 (lin-lin direct) instead. What to do?
  - FUDGE and NJOY both assume the evaluator 'intended' to use unit-base
  - AMPX follows interpolation rules as written, does direct interpolation
  - Other codes??
- Discussion for GNDS specifications: pick a better default interpolation than 'direct'?
- Unit-base and corresponding energies both have limitations, but other alternatives do better.
  - 'Continuous limit of equi-probable bins' (CLEB) from Tamagno at CEA
  - 'Cumulative points' from G. Hedstrom (LLNL, ret.)

FUDGE now supports exporting processed Monte Carlo data in ACE format

```
>python bin/processProtare.py n-001_H_001.endf.gnds.xml -mc \
     -t 300 –t 600 --temperatureUnit K –vvv        # Monte-Carlo only

>python brownies/LANL/toACE/toACE.py n-001_H_001.endf.gnds.proc.xml \
     H1_300K.ace –i 80 –s MonteCarlo_000 –v

>python brownies/LANL/toACE/toACE.py n-001_H_001.endf.gnds.proc.xml \
     H1_600K.ace –i 80 –s MonteCarlo_001 –v
```

- Coverage mostly complete, TNSL and URR support still need to be expanded (see next slides)

FUDGE support for thermal neutron scattering law (TNSL) data recently expanded

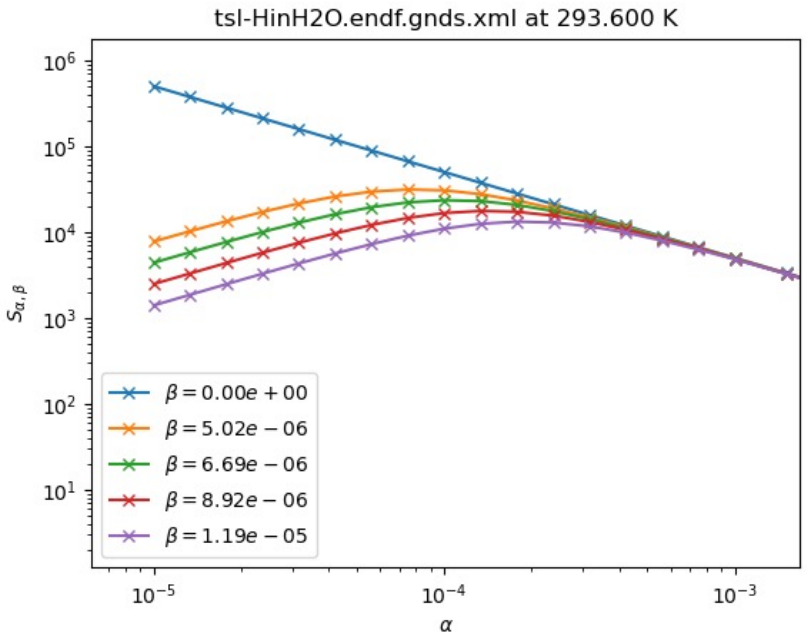
- TNSL evaluations typically contain 1-2* reactions that together replace MT=2 for small incident energies
  - One of
    - Coherent elastic (for crystalline materials)
    - Incoherent elastic (amorphous materials)
  - Plus
    - Incoherent inelastic (all materials)

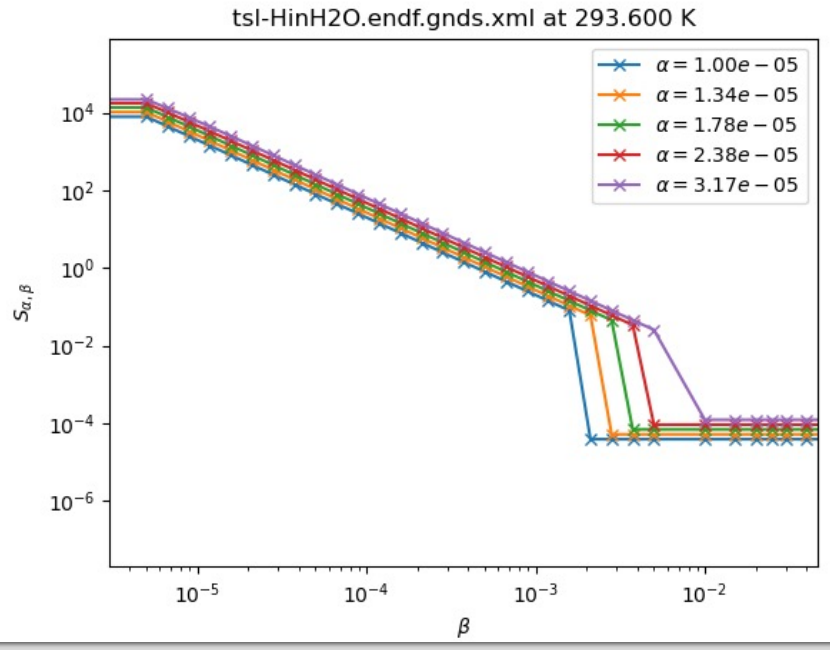
- GNDS 2.0 simplifies handling of TNSL data: breaks the evaluation into three separate reactions
  - Also introduces the 'interaction="TNSL"' attribute to differentiate from pure-nuclear scattering

* Recent ENDF format modification allows storing both coherent and incoherent elastic. GNDS (and FUDGE) handle this change easily

Incoherent inelastic is challenging to process: requires extrapolating beyond tabulated data

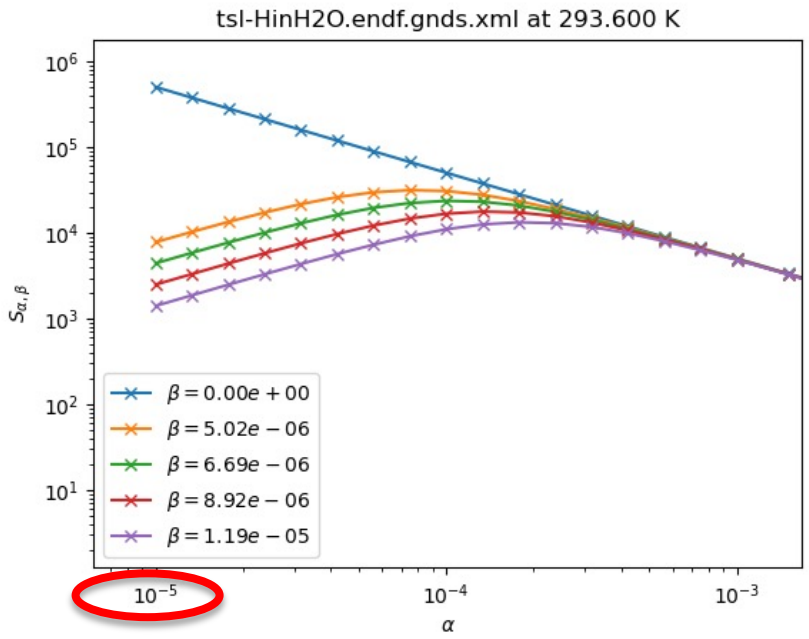
- Evaluations contain tabulated S(alpha, beta, T), but don't extend down to small enough alpha. Processing codes must choose different extrapolation rules
- How to interpolate between beta=0 and beta=5.02e-6?

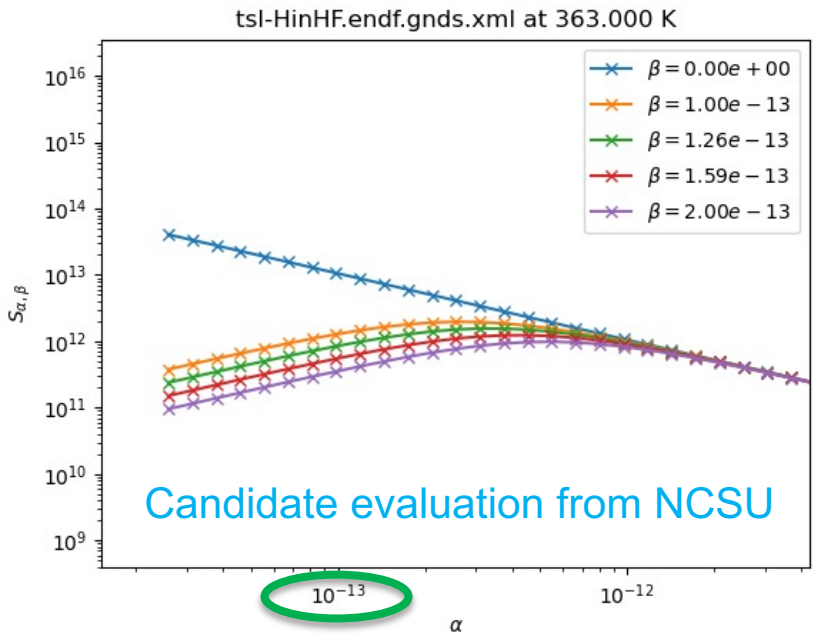




Newer evaluations improve the situation by expanding the alpha,beta grid substantially

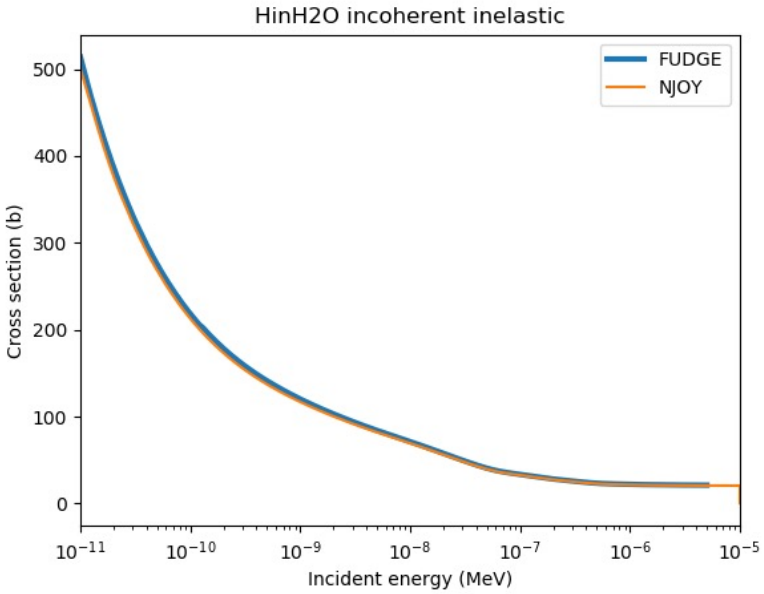
- H in HF: similar question, except evaluation extends to much lower values of alpha and beta

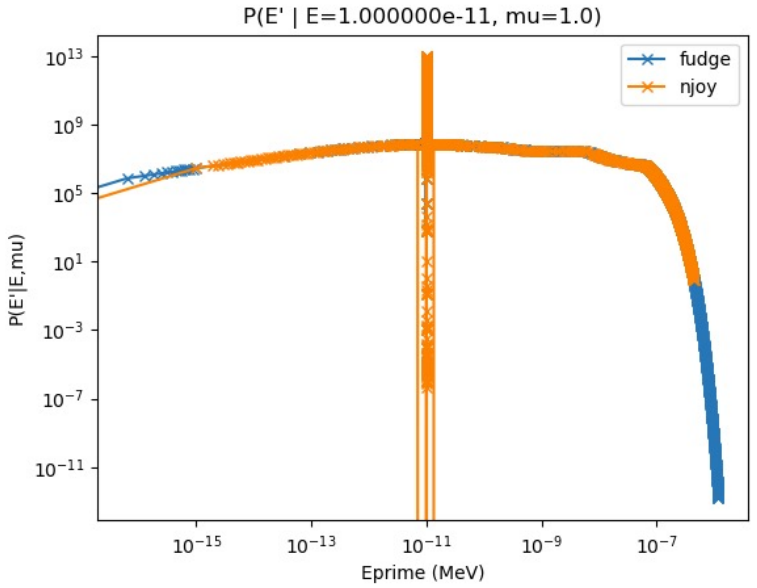




Even with expanded S(alpha,beta), some extrapolation may be necessary

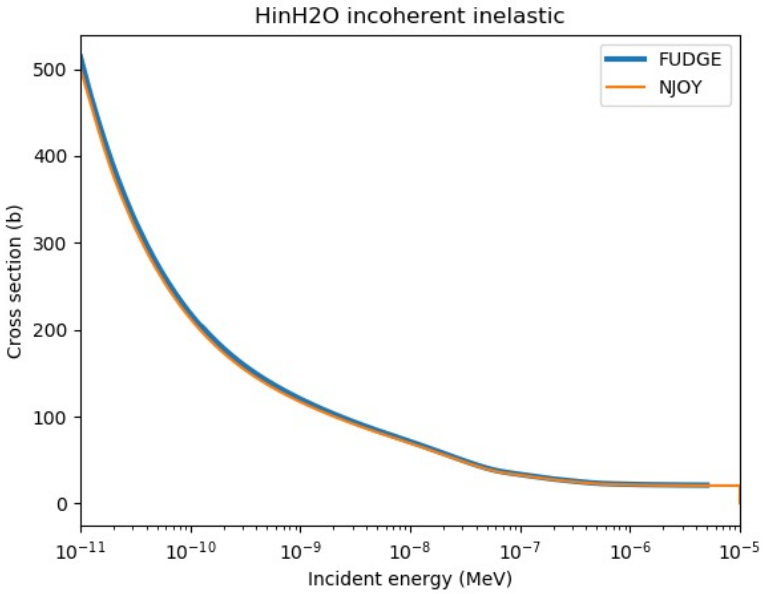
- FUDGE and NJOY (module THERMR) incoherent inelastic cross sections for H in H2O differ by about 2% at small incident energies
  - Mostly seems to be due to treatment of forward scattering near E = E’

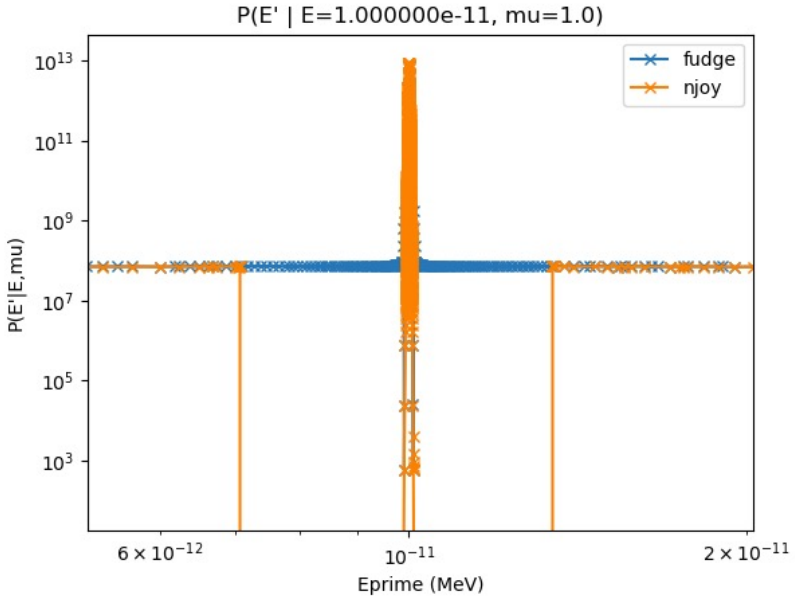




Even with expanded S(alpha,beta), some extrapolation may be necessary

- FUDGE and NJOY (module THERMR) incoherent inelastic cross sections for H in H2O differ by about 2% at small incident energies
  - Mostly seems to be due to treatment of forward scattering near E = E’

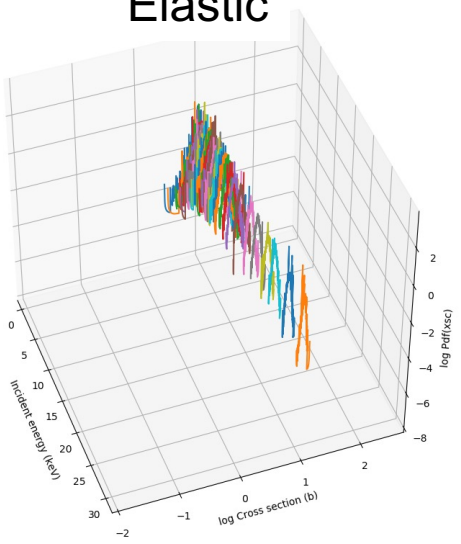




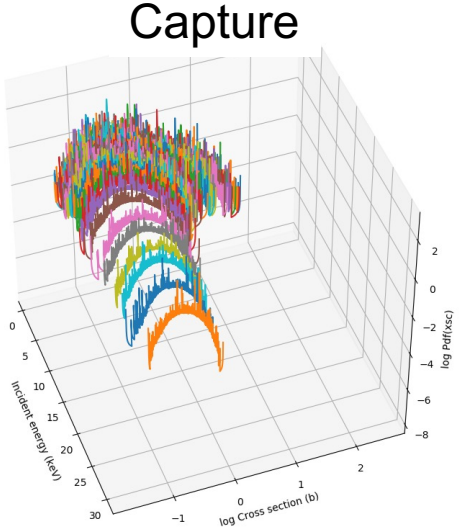
FUDGE also adding URR processing support for self-shielding, etc.

- Draw many random resonance parameter realizations, Doppler broaden and compute cross section probability density. Examples for U233 at 300 K:

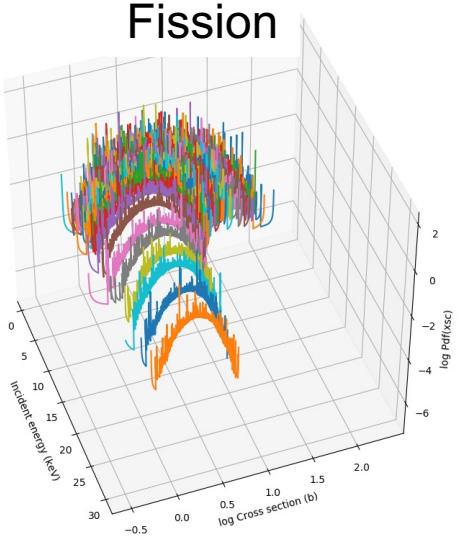
Elastic



Capture

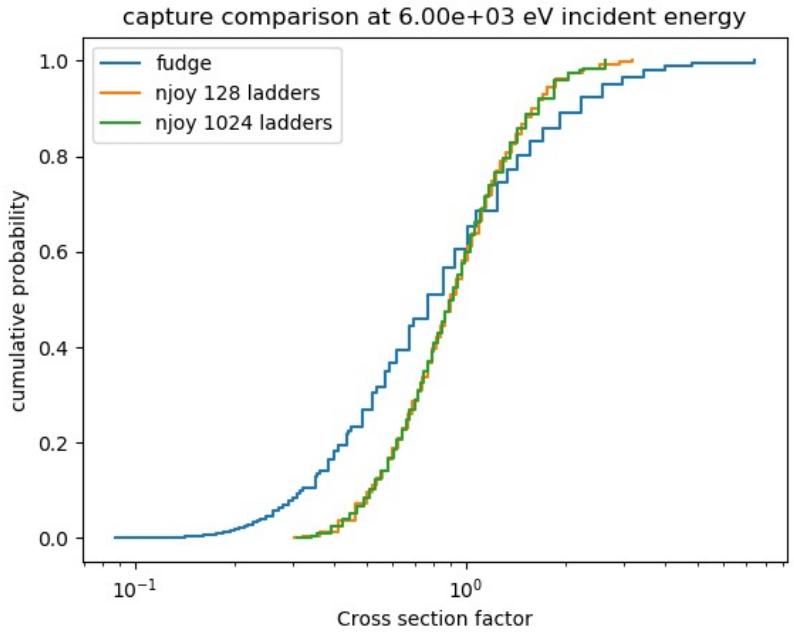


Fission



FUDGE produces wide cross section distributions, especially for neutron capture

- Where NJOY (PURR) results span 1 order of magnitude, FUDGE results span 2-3
- We believe we understand the difference, working to produce more complete analysis of both codes
  - Special thanks to Dave Brown and Matteo Vorabbi (BNL) for contributions here



4) Once evaluations are processed, use 'map files' to assemble into a library

- Map files are flexible, can import individual evaluations or other map files. Examples:

all.map

```
<map library="ENDFB-VIII.0" format="0.1">
 <import path="neutrons.map"/>
 <import path="protons.map"/>
 <import path="deuterons.map"/>
 <import path="tritons.map"/>

</map>
```

neutrons.map

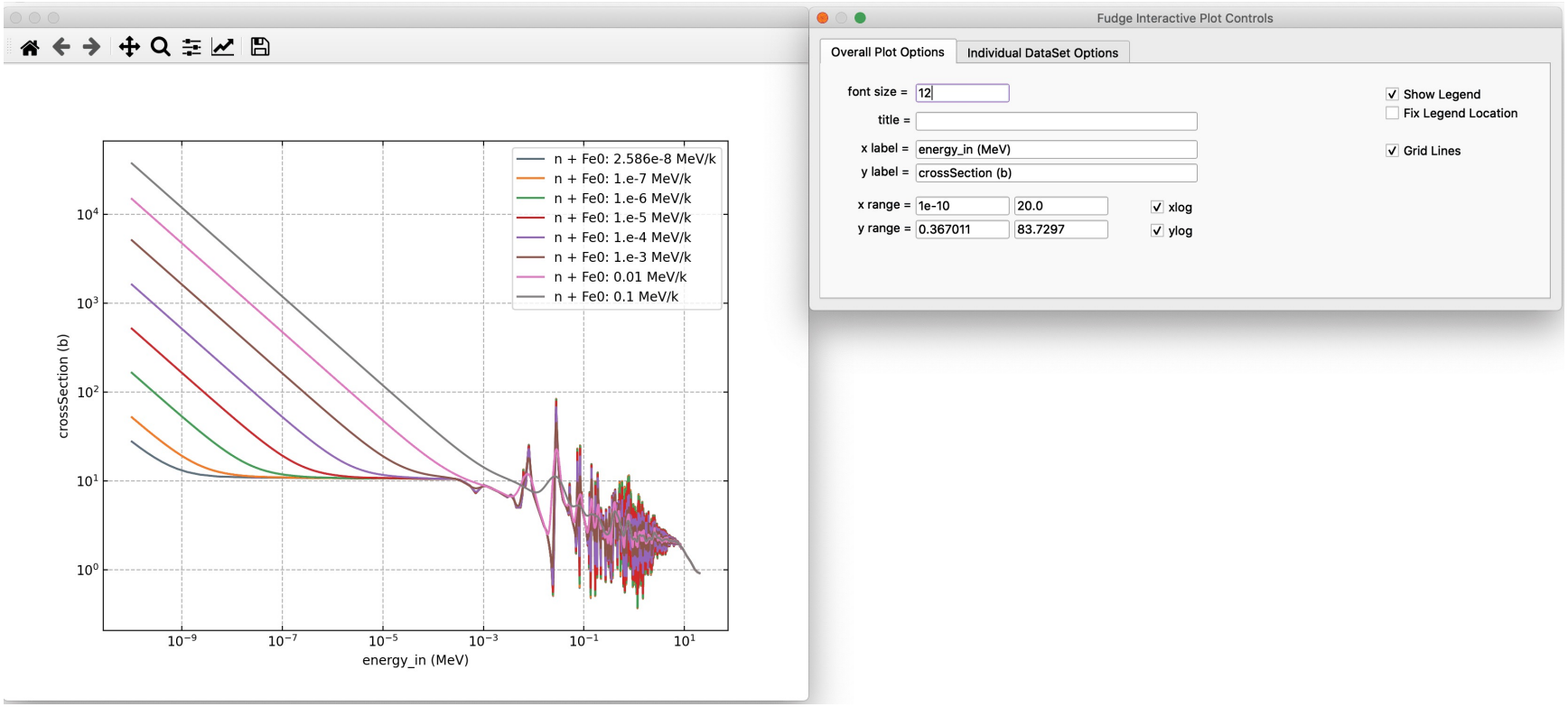
```
<map library="neutrons" format="0.1">
 <protare projectile="n" target="n"  evaluation="ENDF/B-8.0"  path="neutrons/n-000_n_000.xml"/>
 <protare projectile="n" target="H1"  evaluation="ENDF/B-8.0"  path="neutrons/n-001_H_001.xml"/>
 <protare projectile="n" target="H2"  evaluation="ENDF/B-8.0"  path="neutrons/n-001_H_002.xml"/>
 <protare projectile="n" target="H3"  evaluation="ENDF/B-8.0"  path="neutrons/n-001_H_003.xml"/>
 ...
```

- GIDIplus (see Bret Beck's talk tomorrow) uses map files as primary interface to nuclear data libraries

FUDGE supports visualizing multiple kinds of nuclear data

- Previous FUDGE Version
  - Python2
  - Interactive plotting: gnuplot
- Next FUDGE Version
  - Python3
  - Interactive Plotting: PyQt5/Matplotlib
- Designed to work both interactively (from GUI and/or command line) and in script mode.

Example of new interactive plotting capabilities



FUDGE development and deployment:

- Internal version control recently switched from SVN to LLNL GitLab
  - Also published periodically at https://github.com/LLNL/fudge
  - Next release will be FUDGE-4.3
- Pre-requisites
  - Python3
  - Numpy >= 1.15
- Optional
  - matplotlib
  - PyQt5 for interactive plotting
- Makefile
  - Build in place
- `pip install`

Conclusions

- FUDGE has been developed along-side the GNDS standard, and offers extensive support for generating, checking, processing and visualizing GNDS data
- LLNL is migrating to using GNDS for both evaluated and processed data
  - See Bret Beck's talk tomorrow for more discussion of API
- We hope to share experience with other institutions and further encourage deployment of GNDS

Lawrence Livermore
National Laboratory