# FUDGE and GIDIplus development

Caleb Mattoon
Nuclear Data and Theory Group, NACS/PHY

CSEWG formats / processing session
November 6, 2024

LLNL-PRES-857358
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

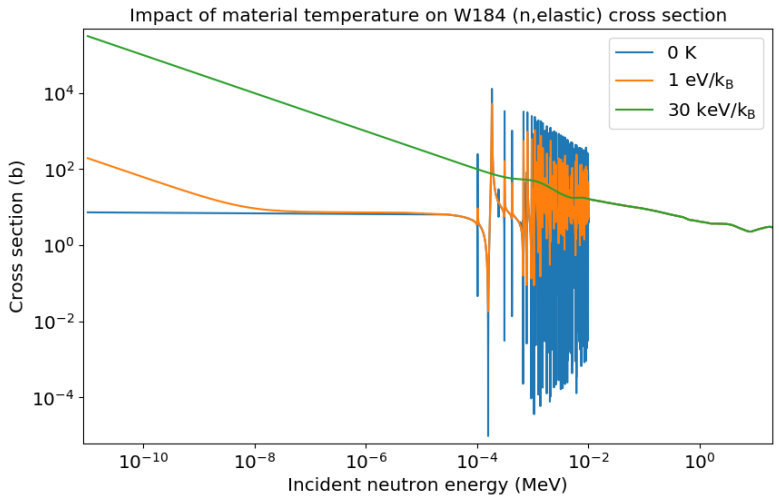
Lawrence Livermore National Laboratory

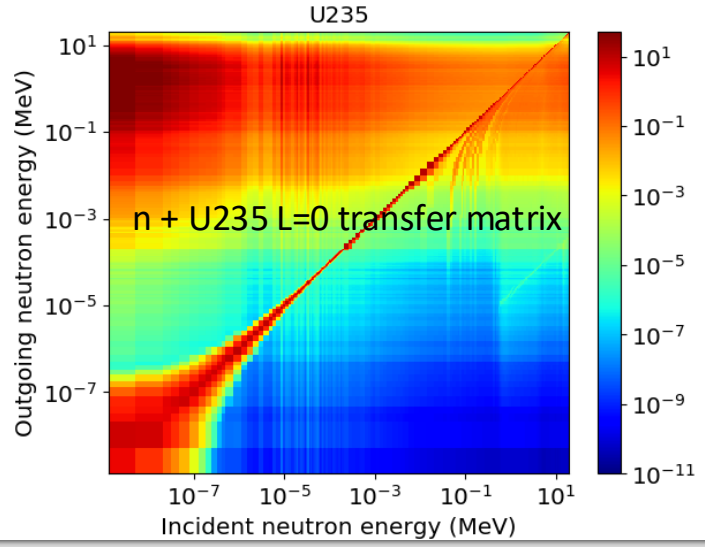
# LLNL codes for managing and processing GNDS nuclear data

- FUDGE: For Updating Data and Generating Evaluations
  - Python-based code for reading, writing, modifying, viewing and processing nuclear data
  - Computationally intensive routines written in C and C++

- GIDI+: General Interaction Data Interface+
  - Suite of C++ APIs for accessing GNDS data for use in transport codes
  - Includes API for sampling GNDS data as needed by Monte Carlo codes

- Both codes are open source and used externally



# FUDGE: *For Updating Data and Generating Evaluations*
# LLNL's primary toolkit for managing GNDS data

- FUDGE was originally developed to support LLNL's ENDL libraries, but most development is focused on GNDS:
  - translating ENDF-6 and ENDL to and from GNDS,
  - checking and processing GNDS-formatted evaluations,
  - providing an easy interface for accessing, visualizing and modifying data.

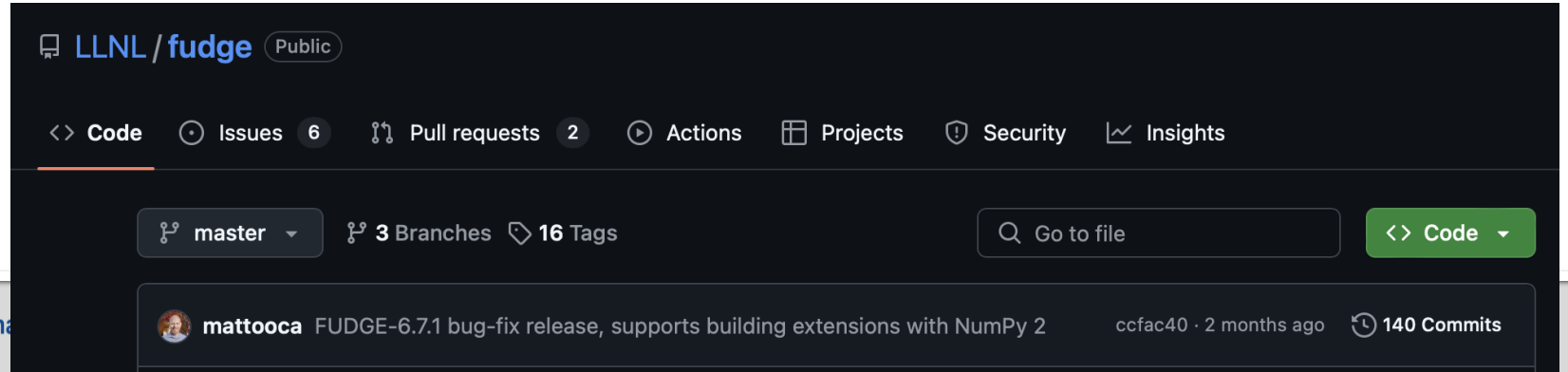
- FUDGE consists of a class library that closely mirrors the GNDS hierarchy, plus lots of scripts to help with common nuclear data needs

# FUDGE-6.7.1 was released on Github in September

- Available at https://github.com/LLNL/fudge
  - Requires Python3.7 or later, numpy, matplotlib, C++ compiler for extensions.
  - Install with pip or with make
  - Release schedule: around 4 public versions per year

- Recent updates:
  - Full support for translating and processing ENDF/B-VIII.1
  - Improved physics checking tools, focus on diagnosing and fixing energy balance

- Next release coming soon. Focus areas:
  - Improved support for the GRIN project (see later)
  - Better tools to support evaluators



# If you only remember one script, the most important is ‘fudgeScripts.py’:

- Run this script to see a summary of most other available scripts:

```
fudgeScripts.py

Scripts in FUDGE:
    GNDSType.py                 - This script prints the GNDS type of each file listed.
    ZA_Info.py                  - For each argument entered, which must be an isotope name specified by either its ZA (1000 * Z + A) or its PoPs id,
    addFlux.py                  - Adds a flux definition (label and f(T,E,mu) data) to a fluxes file (e.g., fluxes.xml).
    addMultigroup.py            - Adds a multi-group boundary definition (i.e., label and the multi-group boundaries) to a groups file
    buildMapFile.py             - Creates a map file from a list of GNDS reactionSuite and map files.
    ...
    peek.py                     - Prints an outlines of the reactions, and their energy domain and products for a GNDS reactionSuite file.
    processProtare.py           - Processes a GNDS reactionSuite file for Monte Carlo and/or deterministic transport at various temperatures.
    processURR.py               - This script process unresolved resonances to create probability tables.
    resonanceSummary.py         - Prints summary information about resonances for a GNDS reactionSuite file.
    stylesTree.py               - Prints a tree representation of the styles in each specified GNDS reactionSuite.
    temperatures.py             - Prints the list of temperatures in a GNDS reactionSuite and labels for each processed style for each temperature.

Scripts in PoPs:
    Q.py                   - Calculates the Q-value for the list of ingoing and outgoing particles (optionally, threshold).
    ...

Scripts in xData:
    convolute1d.py    - Reads 1d data from a file and convolutes its data with data from another file or a Gaussian.
    plotXYs1dFiles.py - Reads 1d data from each file listed into an XYs1d instance and plots all using XYs1d.multiPlot.
    sumXYs1dFiles.py  - Reads 1d data from each file listed into an XYs1d instance, sums them and prints the sum.
```

More information about each script is available through the ‘-h’ option.

# Updated the FUDGE physics checker to focus on most important warnings

- Each warning now has a severity level (Pedantic / Minor / Moderate / Severe / Fatal)
  - checkGNDS.py supports filtering by severity and/or by warning types.
  - Default warning threshold = Moderate

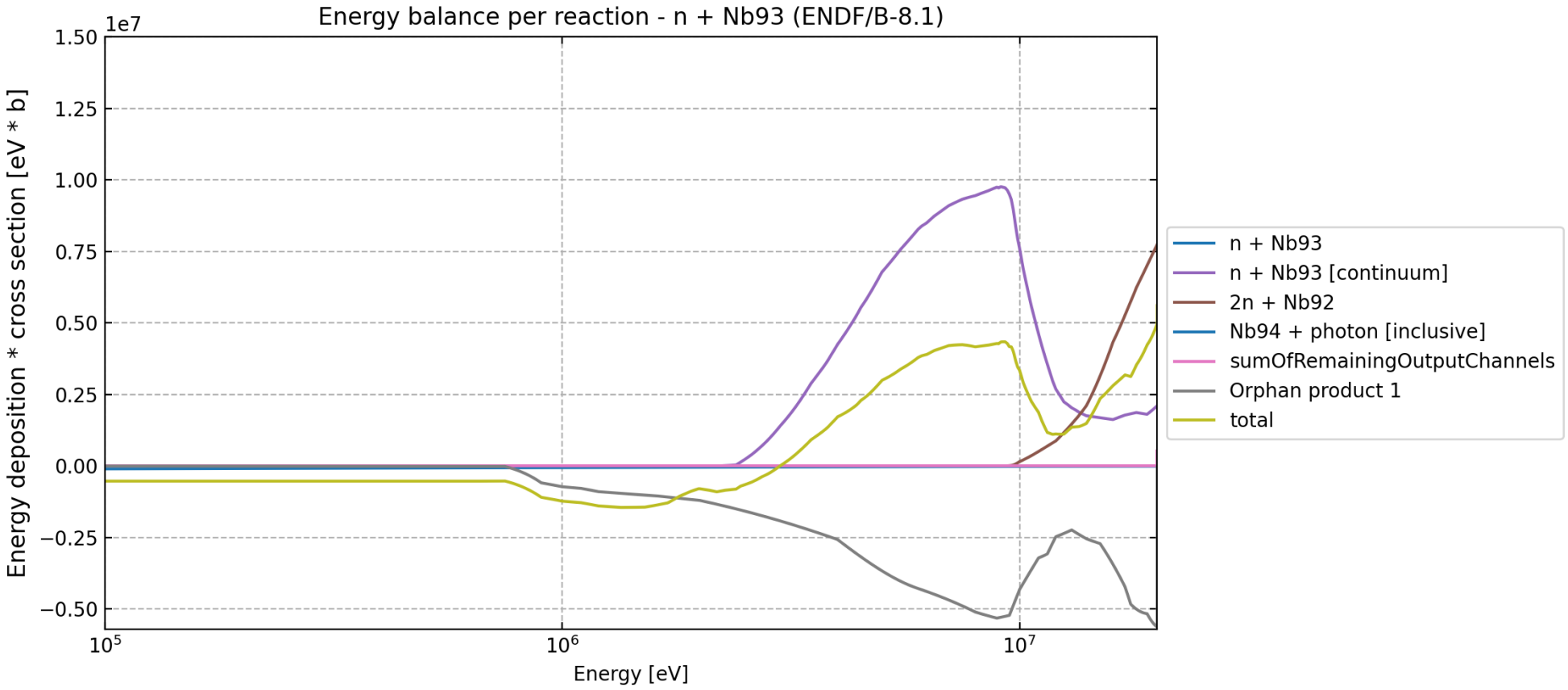
- checkGNDS used extensively to test ENDF-VIII.1 candidates:

```
checkGNDS.py d-001_H_003.xml -e --threshold Moderate
ReactionSuite: H2 + H3
   reaction label n + (He4_e1 -> H1 + H3)
       Energy balance (after decay) for products: n, H1, H3
           Severe warning: Energy imbalance at incident energy 3.713e6 eV (index 0).
             Total deposited = 112.9% (H1 = 70.02%, H3 = 22.84%, n = 20.06%)
           ...
           Severe warning: Energy imbalance at incident energy 2.e7 eV (index 928).
             Total deposited = 136.3% (H1 = 70.16%, n = 43.24%, H3 = 22.88%)
   reaction label n + He4 + photon [continuum]
       Energy balance for products: n, He4, photon
           Moderate warning: Energy imbalance at incident energy 1906250. eV (index 414).
             Total deposited = 94.99% (photon = 85.97%, n = 4.866%, He4 = 4.155%)
           ...
           Moderate warning: Energy imbalance at incident energy 5.5625e6 eV (index 605).
             Total deposited = 95% (photon = 72.39%, n = 14.24%, He4 = 8.367%)

 Some warnings were screened
   Pedantic: 2 occurrences
   Minor: 452 occurrences
```

# New script ‘energyBalance.py’ helps diagnose energy conservation problems:

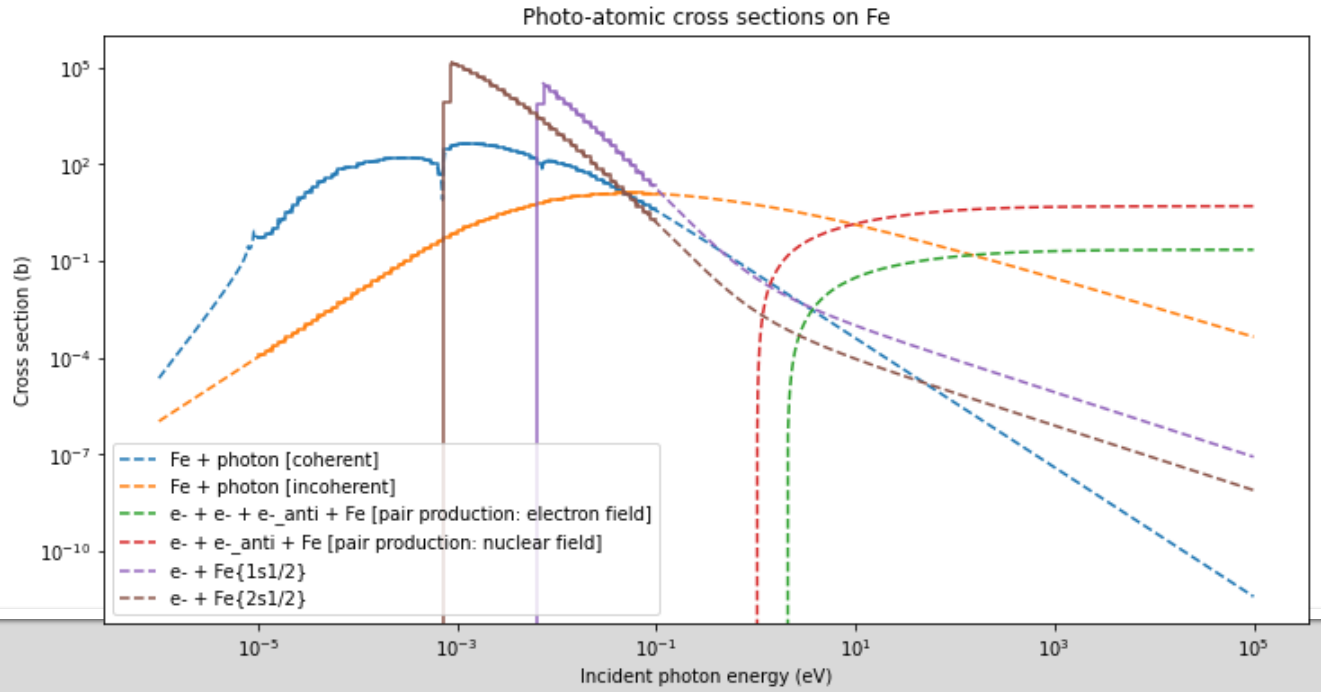
python3 energyBalance.py n-041_Nb_093.xml ebalance --plot -w



# FUDGE supports processing for both Monte Carlo and deterministic transport

```
# convert ENDF-6 to GNDS:
python3 fudge/brownies/bin/endf2gnds.py  photoat-026_Fe_000.endf

# process (results stored in new GNDS file):
python3 fudge/bin/processProtare.py photoat-026_Fe_000.endf.gnds.xml \
    -mc -mg --groupFile groups.xml --fluxFile fluxes.xml \
    --gid photon=photon_electron --fluxID LLNL_fid_1
```



# Another important goal for FUDGE development: expanded support for generating GNDS evaluations

- TAGNDS and parseYAHFC support converting TALYS and YAHFC (Hauser Feshbach) code outputs to GNDS.
- FERDINAND: Ian Thompson's tool for converting between various R-Matrix code inputs, using GNDS as a common exchange format
- New tool by Vincent Cheung translates ENSDF adopted levels and decay data into the GNDS 'PoPs' particle database
- **Still needed**: better support for adding covariances and merging data from multiple sources into a single evaluation.

# GIDIplus: C++ API for reading and sampling processed GNDS data

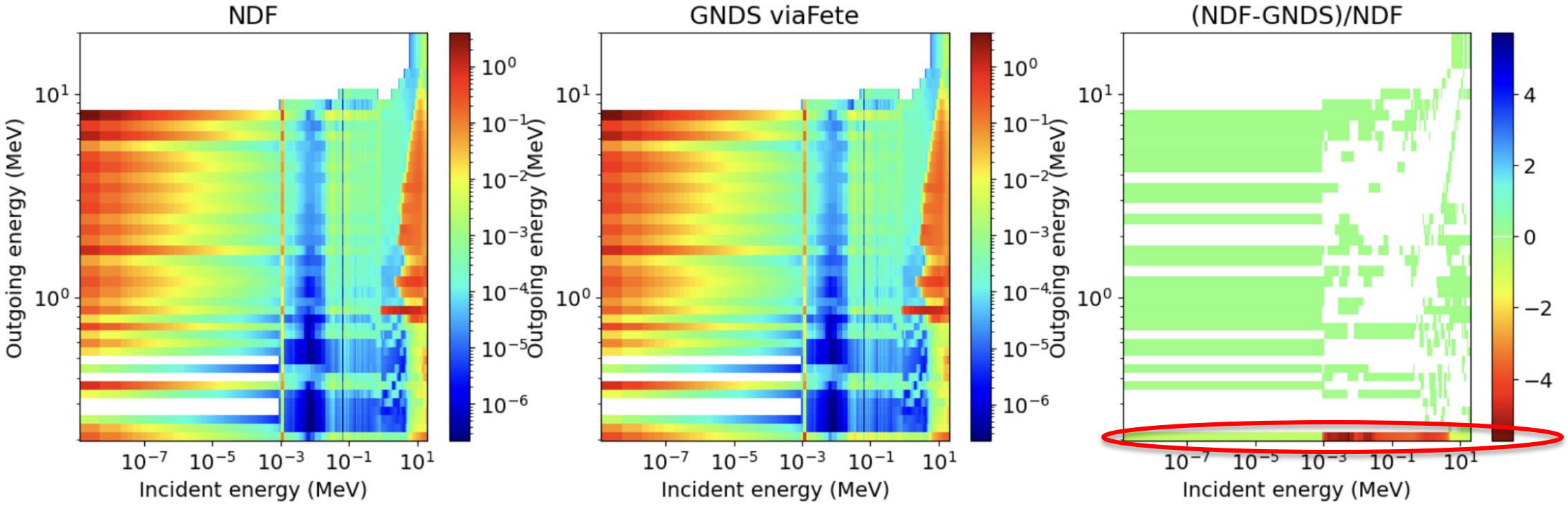
- GIDIplus is open source, available from https://github.com/LLNL/gidiplus
  - Latest public release is 3.28.22, new update coming soon
- LLNL codes are updating to use GIDIplus by default
  - Effort involves extensive testing and review by Mercury (Monte Carlo) and Ardra (deterministic) transport code teams
- Older version of GIDIplus is used in the GEANT-4 package G4LEND
  - We recently updated G4LEND as part of the GRIN project. Updates expected to be part of the next G4 release

# Recent FUDGE and GIDIplus developments were driven partly by an LLNL milestone

- 2024 milestone: assess the possibility of switching to GNDS / GIDI+ as the default source of nuclear data for LLNL transport applications
  - Required extensive comparison between legacy ENDL-based data and new GNDS tools
  - Not just switching the data format (ENDL to GNDS): this also meant converting to new processing code and APIs
- Several important differences came up during the milestone.
  - Example: different formulation for outgoing photon transfer matrices

# Sample difference between legacy and new processing tools uncovered during the milestone:

- ENDL2009.4 n + Fe56 L=0 photon production matrix




Differences especially noticeable
In lowest outgoing energy bin

# Three formulations of transfer matrix element for incident energy group 'g', outgoing energy group 'h', Legendre order $\ell$:

$$\sigma_\ell = \frac{\int_g \phi^\ell(E)\sigma(E) \int_h \pi_\ell(E \to E')dE'}{\int_g dE\phi^\ell(E)}$$

Option 1: conserve # of particles

$$\sigma_\ell = \frac{\int_g \phi^\ell(E)\sigma(E) \int_h \pi_\ell(E \to E')E'dE'}{\bar{E'}_h \int_g dE\phi^\ell(E)}$$

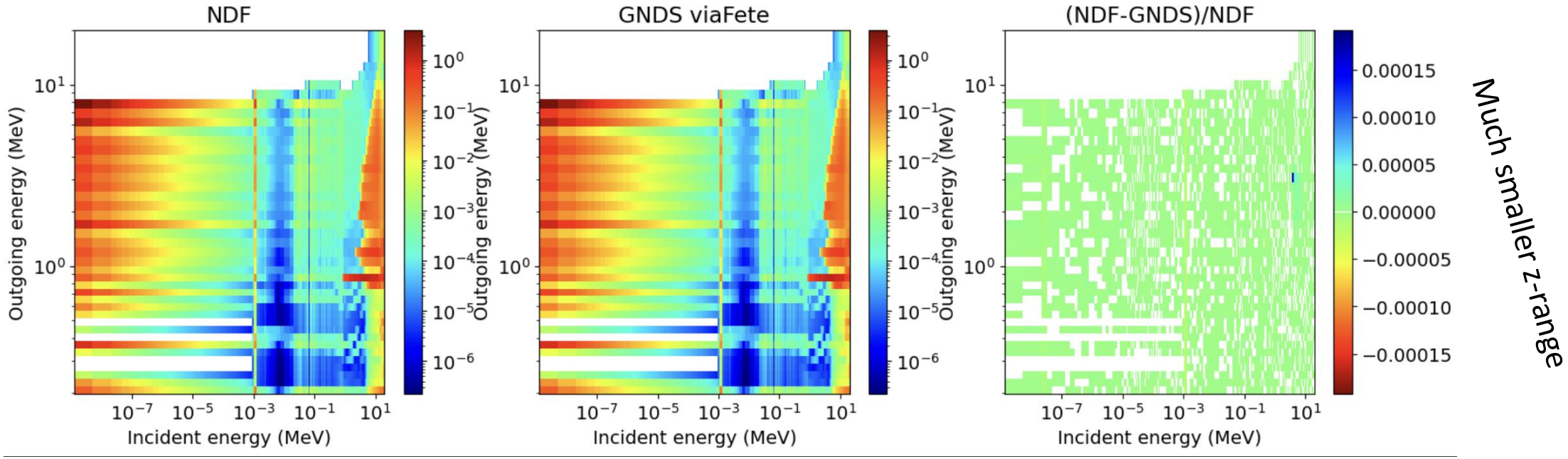
Option 2: conserve energy *production*

$$\sigma_\ell = \frac{\int_g \phi^\ell(E)\sigma(E) \int_h \pi_\ell(E \to E')E'dE'}{\int_g dE\phi^\ell(E)}$$

Option 3: conserve energy

# Sample difference between legacy and new processing tools uncovered during the milestone: Resolved!

- Same transfer matrix comparison (photons from n + Fe56) after switching to option 2:




processProtare.py now uses the energy-conserving option by default, for better agreement with legacy LLNL data.

# Initial FUDGE/GIDI+ training course was held May 21-24, 2024, with another course planned for June 2025 (week after WPEC)

NEA NUCLEAR ENERGY AGENCY

ABOUT US TOPICS NEWS AND RESOURCES

Search the whole site...

## Programme

This three-and-a-half-day introductory class is intended to give nuclear data evaluators and users an introduction to using the open-source codes FUDGE and GIDIplus (available from https://github.com/LLNL/fudge and https://github.com/LLNL/gidiplus) to generate and use nuclear data in the Generalised Nuclear Database Structure or GNDS.

FUDGE is a code (primarily written in Python) developed by Lawrence Livermore National Laboratory (LLNL) to manage generating, checking, visualising, processing, and using nuclear data libraries. The name stands for "For Updating Data and Generating Evaluations". It is primarily intended for use with data stored in Generalised Nuclear Database Structure (GNDS) files.

*Course material mainly consists of interactive Jupyter notebooks*

```python
from fudge import reactionSuite
RS = reactionSuite.read("n-004_Be_009.xml")
print(RS.toString())
```

```
n + Be9 --> n + Be9
            2n + 2He4
            H1 + Li9
            H2 + Li8
            H3 + Li7
            H3 + (Li7_e1 -> Li7 + photon)
            He4 + He6
            Be10 + photon [inclusive]
```

**2025 course announcement coming soon at the NEA**

Lawrence Livermore
National Laboratory