OAK RIDGE National Laboratory

AMPX and SAMMY Status Report

Andrew Holcomb, Doro Wiarda

G. Arbanas, J. Brown, C. Chapman, K. Guber, J. McDonnell, M. Pigni

CSEWG Annual Meeting 2021

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

PUFF/COGNAC Improvements

- If it is a minor roundoff problem, we bump the values back into the valid range, and report it (as in SCALE 6.2)
- If an egregious error (outside of precision) is detected for **ANY** matrix element, PUFF and COGNAC (AMPX covariance modules) will now:
  - Set self correlation matrices to the identity matrix
  - Set cross correlation matrices to the zero matrix
- In practice, this has only affected a small subset of isotopes in the ENDF-8 covariance library

SCALE 6.3 Data

- ENDF-7.1
  - Corrected probability tables for subset of evaluations (in MG & CE)
  - New coupled MG libraries, xn252g47v7.1 and xn56g19v7.1
    - gamma group structures identical to v7.1-200n47g and v7.1-28n19g
    - neutron group structures identical to v7-1.252 and v7.1-56
  - New Sodium Fast Reactor (SFR) MG library, 302 groups
  - New general-purpose MG library, 1597 groups
- ENDF-8.0
  - MG libraries
    - Distributed in 252 (thermal), 302 (SFR), and 1597 (generic) groups
    - Shielding libraries in 200n47g and 28n19g format
  - CE library now distributed in HDF5 format
  - Covariance data in 56 groups (includes best estimate augmentation)
  - 56 group perturbation libraries for SAMPLER

SCALE 6.3 Data (cont.)

- Removed all ENDF-7.0 libraries (kept all ENDF-7.**1** libraries)
- Standard composition library is now rev40 (rev38 and rev39 removed)
- Removed ORIGEN JEFF activation libraries with no corresponding transport library structure (*44, 47, 49, 238*)
- Removed obsolete helper data no longer required by modern sequences

Note: SCALE 6.2 data is still fully operational with SCALE 6.3. A limited set of SCALE 6.3 data can be used with the 6.2.* series.

Thermal Scattering Law

- Short Collision Time subroutine and angular gridding improvements currently under review for inclusion in future AMPX release
- Implementation of mixed elastic scatting format underway
  - Combining incoherent elastic & inelastic into same MT number to save space
  - Will be backwards compatible for all TSL files with incoherent elastic & inelastic data

How to Support GNDS

- Internal AMPX C++ structures are the "API" that AMPX and SAMMY will access.
- Add a Reader/Writer that supports GNDS and fills AMPX internal C++ structures.
- Test by processing ENDF formatted and GNDS formatted files and compare results.
- Find an efficient way to support GNDS which also allows us to easily apply updates.
- Updates might not immediately be propagated to the internal AMPX C++ classes if not needed in AMPX.

GNDS access layers in AMPX

- Python generated C++ classes for all objects described in the GNDS specification. Generation is based on the JSON files of the GNDS standard. Approx. 290 classes are generated. All inherit from GNDElement.
- Special names are selected for GNDS objects such as Double, which have names not allowed in C++.
- Namespaces are handled as in the GNDS specification, thus the same name but in different namespaces is allowed.
- Some correction for errors in the specification are built into the Python generation code. If corrections are applied, they are reported.

These classes are a very low-level access API to the GNDS content, that mirror the specification directly.

GNDS access layers in AMPX (cont.)

Classes that fill the AMPX C++ in-memory structures. These classes are needed as:

- To select the correct "style" of the data the user requested, which includes following the inheritance chain.
- Convert GNDS units to AMPX units
- Convert GNDS constructs into AMPX constructs.
- More user-friendly access methods to Particle data base

This layer is currently only reading data, but writing will be added as needed. The first implementation for reading will be for resonance parameters and corresponding covariance matrices for use in SAMMY.

These classes are available in the SCALE 6.3 release.

GNDS access layers in AMPX (cont.)

Python code for generating low-level C++ routines as well as the generated files are available **open source**:

https://code.ornl.gov/RNSD/gnds

Currently works for GNDS-1.9, (a small fix is needed for the master branch – patch available in the repository).

C++ classes support reading and writing.

We are in the process of updating to the upcoming GNDS-2.0 specification.

SAMMY/AMPX and GNDS – future plans

- SAMMY has its own ENDF reading and writing routines.
- **We plan on switching to the AMPX reading and writing routines.** This was delayed in favor of using in-memory C++ AMPX classes for resonance parameter and all covariance information.
- Having the relevant information in the in-memory classes will make it much easier to switch out the reading and writing to use AMPX methods.
- AMPX is in the final stages to add support for reading GNDS formatted ENDF files.
- **Switching to these routines will bring GNDS support to SAMMY.** In this context, writing of GNDS files will also be added.

SAMMY overview

- Updates in the covariance matrix storage and access which allowed more robust handling of radius parameters and pup'ed parameters
- Derivatives and cross sections are now stored in a C++ grid
- New framework for the 0K resolved resonance reconstruction.
- Preliminary API for use with alternative fitting routines

Covariance storage

- All covariance information is stored in the same C++ object used in AMPX. Including:
  - Resonance parameter covariances (including pup'ed parameters)
  - Data covariance
  - Implicit data covariance information
- This allows the removal of various scratch files.
- Created a helper C++ class that keeps track:
  - Were adjusted parameters are stored in the covariance matrix (decoupling the input order and calculation order)
  - Whether a parameter is a pup'ed parameter

This helps especially for parameters that are coupled, for example all radii are the same or all gamma width for a given spin group (several bugs related to this were found and corrected),

In the future this will allow more user-friendly output formats for covariance information.

Updated Derivative Classes

- Last year we reported that we added a C++ class for the final cross section and derivatives.
- We added a derived class that keeps track of cross section and derivatives throughout SAMMY:
  - The class manages zero suppression for derivatives that can only be non-zero for a given isotope Needed if the sample contains more than one isotope, where the cross sections and derivatives will be collapsed to the sample cross section before the adjustment
- The use of the class again allowed the removal of several scratch files.
- Internally SAMMY has two types of adjusted parameters:
  - Only non-zero for a given isotope (like resonance parameters)
  - Potentially non-zero for all isotopes (like normalization parameters)
- While the new class allows to remove those two types of parameters (as it handles the needed zero-suppression) it is currently only used in the resolved resonance reconstruction.
- With the use of the new class, the order of the parameters (needed to distinguish between the two) will no longer matter. This will make it much easier to add additional parameters.

Frame-work for 0K resolved resonance range

SAMMY support several different resonance formats in the resolved range. In addition, we would like to connect the AMPX version to this mix.

- For each of these formalism there was a lot of duplicate code related to the looping over the energies, transformation to transmission if desired, etc.
- All formalism depend on SAMMY global parameters, which makes code re-use hard.
- We created (still in Fortran):
  - CrossSectionCalculator*, containing information common to all formalism (like resonance parameter and covariance information) plus a function to calculate the cross section and derivatives for one energy. Child classes for all supported RR formalisms exist.*
  - CrossSectionCalcDriver, a class that selects the correct class of *CrossSectionCalculator*
  - ZeroKCrossCorrections, a class that calculates the 0K cross section and derivatives on all energies and applies desired 0K corrections.
- We reused the existing Fortran code for all the formalism but references to SAMMY global parameters have been mostly removed.
- We plan to connect our existing AMPX resonance API to this framework.

Preliminary API

The program flow in SAMMY is not very yet structured so that we can easily add an API to allow for access to SAMMY from external programs.

However, it would be advantageous to use SAMMY to calculate cross section data and derivates including experimental corrections to evaluate for example different fitting algorithms.

In order to investigate this, we have added a very preliminary API that:

- Reads the usual SAMMY input
- Stops right before an adjustment step, i.e. after calculating the cross section and derivatives.

This API is currently used for in the Monte Carlo evaluation project in conjunction the Metropolis-Hastings algorithm

This work was supported by the Nuclear Criticality Safety Program, funded and managed by the National Nuclear Security Administration for the Department of Energy.