Recent Developments in the Nuclear Data Processing Code AMPX

AMPX Team: J. Brown, C. Chapman,

J. McDonnell (speaker), D. Wiarda (ORNL)

A. Holcomb (NEA)

ND 2022

July 28, 2022

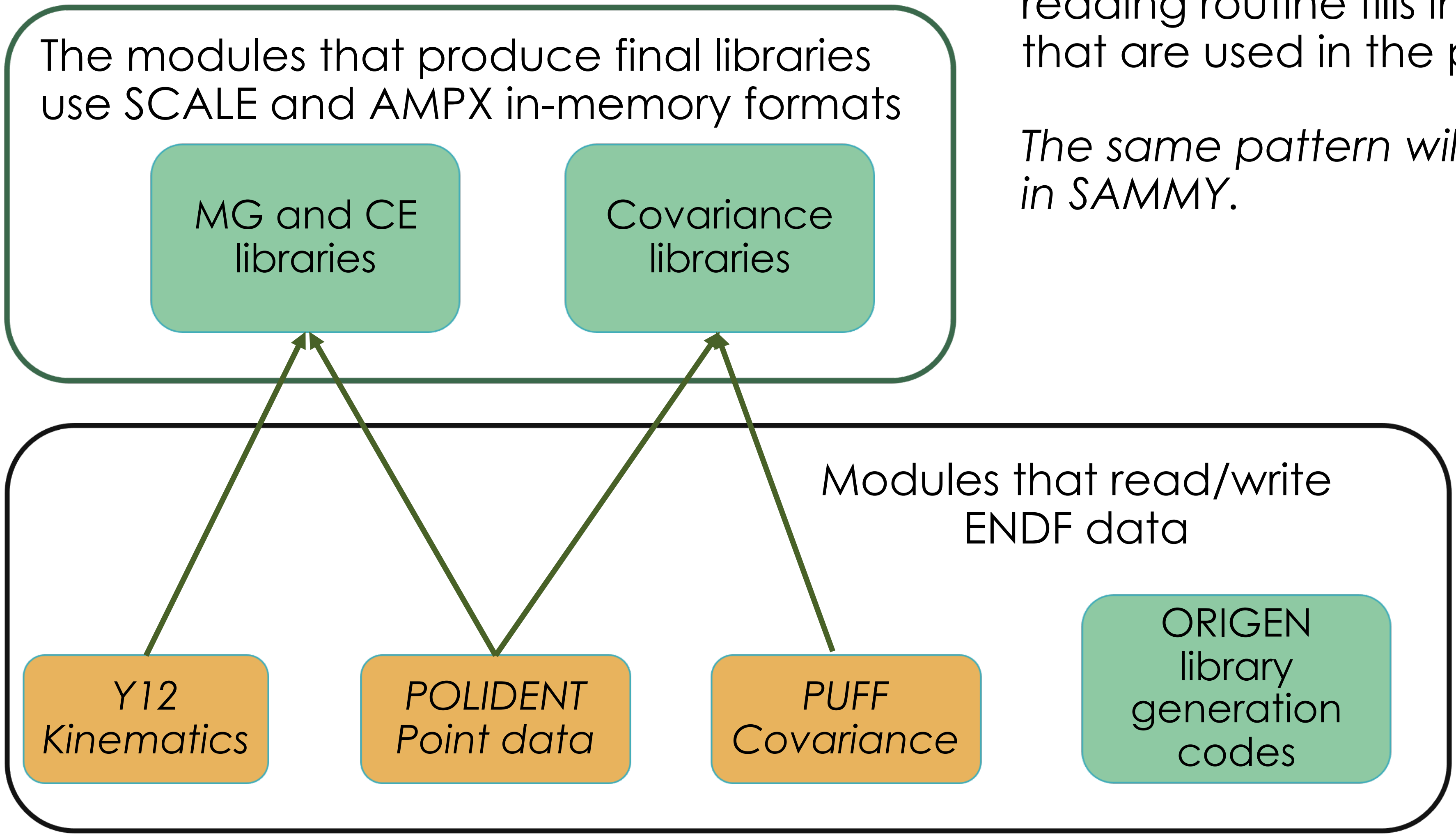
Summary

- Overview about ENDF and GNDS reading in AMPX
- AMPX Covariance Processing Updates
- Thermal Scattering Law Updates
- Data Libraries for SCALE 6.3.0

ENDF data in AMPX

No processing code in AMPX directly accesses the ENDF files. An ENDF reading routine fills in-memory structures that are used in the processing codes.

The same pattern will be incorporated in SAMMY.



Other modules use AMPX/SCALE internal data formats

AMPX Strategy to Support GNDS

- Internal AMPX C++ structures are the "API" that AMPX and SAMMY will access.
- Requirements:
  - Reader/Writer that supports GNDS and fills AMPX internal C++ structures.
  - Testing via processing ENDF formatted and GNDS formatted files and compare results.
  - An efficient way to support GNDS which also allows us to easily apply updates, prioritized according to the needs of AMPX/SCALE.

GNDS low access classes

- Several low-level access classes are used to access the GND files.
- Code has been updated to work with the GNDS-2.0 branch in the NEA GNDS gitlab
- Code is available from https://code.ornl.gov/RNSD/gnds/
- Branch for GNDS-2.0 is at: https://code.ornl.gov/RNSD/gnds/-/tree/GNDS-2.0

On-Disk file access

The direct access to the XML (or HDF5 or JSON) GNDS file is abstracted into access to elements and attributes:

Interface class *Attribute*	Interface class *Element*	Interface class *EndfDocument*
getName *setName* *getValue* *setValue*	*getNumAttribute* *getAttribute* *getNumElements* *getElement* *getValue* *getValueData* …. Setter functions	*getRootElement* *createRootElement*

Based on pugixml. This is a layer on top of the actual XML DOM reader to allow easy switch to different reader if needed.

Classes are in *src/xml*

Parent classes for all GND classes

- ***EndfDocument***
  - The abstract class to allow access to the elements and attributes of the GNDS file to shield the downstream classes from on-disk format
- ***Definitions***
  - Basic conversion roles from text to numbers
  - Enumerations for Interpolation rules
- ***Container***
  - Establishes the tag name and keeps track of the *label* and *value* attributes
  - Has the base implementation to retrieve/cache element information from disk

Classes are in *src* directory

Classes for selected GPDC objects

In order to make access more fine-grained a couple of low-level GNDS data container have hand-edited C++ source code:

- ***ValuesContainer*** (for ”gpdc::values” child)
- ***TableContainer*** (for “gpdc::table” child)
- ***ArrayContainer*** (for “gpdc::array” chid)
- ***ExternalFiles*** (for “gpdc::ExtrernalFiles” and “gpdc::ExtrernalFile” children)

Most GNDS classes inherit from ***GNDElement*** to manage references and external files.

Classes are in *src* directory

GNDS specification classes

The remaining classes are generated from the JSON definition files, using python classes:

- ***GenerateFromJson*** and helper files

Input is:

- File containing the JSON files to parse (relevant file for GNDS-2.0 is given as ***FormatJsonFile*** in *FromJson* directory)
- The directory containing the GNDS-2.0 gitlab data
- Final directory for the generated classes

Generated classes are available in *src/gnd* directory.

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.

Generated C++ classes

- The classes, by virtue of being generated directly from the JSON specifications are very close to the GNDS specification
- Children are:
  - One shared pointer to the child class if occurrence is one or optional
  - Vector of shared pointer if occurrence is more than one
- Classes allow to read and write GNDS formatted files.
- ToDo: We need to add more unit tests for all the classes.

GNDS access layers in AMPX (cont.)

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

Classes for GNDS-1.9 are available in SCALE 6.3 release and they are currently updated as needed for GNDS-2.0.

Covariance Processing: 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, these corrections have only affected a small subset of isotopes in the ENDF-8 covariance library

Thermal Scattering Law

- Improved angular gridding algorithm & more robust Short Collision Time subroutine under review for inclusion in future AMPX release
  - More accurate processing of moderators at cryogenic temperatures
- Implementation of proposed mixed elastic scatting format underway
  - Multiple strategies being investigated:
    - Combining elastic & inelastic incoherent functions into same MT
    - Adding separate MT for incoherent elastic

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
  - 56 group perturbation libraries for SAMPLER

- 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
  - 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.

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

Backup Slides

Note on python generator classes

- C++ classes will automatically be generated for all tags listed in the specified JSON definition files.
- Tags like ”table” or “array” will immediately reference the hand-coded classes instead of generating new classes.
- Some resolution for ambiguous tags is attempted and reported (see next slide).
- Some tags need to have special C++ names, as the tag name is a reserved name ('double’ -> 'gndsDouble’)
- JSON namespace become C++ namespaces
- The project has a CMakeLists.txt file that allows to build a library. This library is directly built and used in our SCALE builds.

Future Plans

- Continue integration of AMPX and SAMMY components, focus on modern design and de-duplication, efficient storage
- Migrate AMPX R-Matrix calculation to take advantage of new SCALE/AMPX/SAMMY linear algebra interface
- Modernize PURM to improve run-time performance (perhaps use SLBW + Leal-Hwang doppler broadening)
- GNDS 1.9 -> 1.10 after 1.10 is finalized

Example output for ***GenerateFromJson***

```
bash-3.2$ python GenerateFromJson.py  FormatJsonFile ~/fudge/formats/ ../src/gnd/
Reading: /Users/dw8/fudge/formats//Styles/summary_styles.json
…
Reading: /Users/dw8/fudge/formats//Pops/summary_pops.json
mass refered to from tsl was found in common and pops
   Disregarding the pops version and use common
  Failed to  find xs in  context gpdc
     using xs_in_xs_pdf_cdf1d in gpdc
  Failed to  find cdf in  context gpdc
     using cdf_in_xs_pdf_cdf1d in gpdc
  Failed to  find PoPs in  context resonances
     using PoPs_database in pops
Q refered to from resonances was found in common and pops
   Disregarding the pops version and use common
uncertainty refered to from abstract was found in gpdc and pops
   Disregarding the pops version and use gpdc
  Failed to  find functional in  context abstract
     using functionalNode in abstract
product refered to from fissionFragmentData was found in common and pops
   Disregarding the pops version and use common
energy refered to from processed was found in common and pops
   Disregarding the pops version and use common
energy refered to from processed was found in common and fpy
   Use the common version
  Failed to  find angular in  context processed
     using angular_uncorrelated in transport
products refered to from transport was found in common and pops
   Disregarding the pops version and use common
  Failed to  find PoPs in  context transport
     using PoPs_database in pops
```

Note on python generator classes cont.

All data classes like ***XYs1D***, ***regions1D***, etc now contain an "*abstractNode*" listed as "*functional*" or "*functionalNode*" in the GNDS definitions.

- The python code checks that it is in namespace gpdc and that is one of the classes denoted below

- The content of "*function1ds*" is rewritten as having children:
  - '*XYs1d*', '*regions1d*', '*gridded1d*', 'Ys1d', '*Legendre*' , '*polynomial1d*', '*constant1d*', '*xs_pdf_cdf1d*'

- The content of "function2ds" is rewritten as having children:
  - '*XYs2d*', '*regions2d*', '*gridded2d*'

- The content of "function3ds" is rewritten as having children:
  - '*XYs3d*', '*regions3d*', '*gridded3d*'

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.

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/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 AND AMPX Release

- SAMMY source code is available from https://code.ornl.gov/RNSD/SAMMY
- The code currently needs SCALE 6.3 beta 9 and up to compile, instructions are provided.
- AMPX is part of SCALE and available with SCALE 6.2 and up. Can be requested from the NEA Databank and RSICC.
- We are allowed to distribute AMPX as open source but have not yet completed the correct mix of decoupling and sharing with SCALE.

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.

C++ classes support reading and writing.

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