Status of GNDS support in AMPX/SAMMY and future outlook

D. Wiarda

A. Holcomb

G. Arbanas

J. Brown

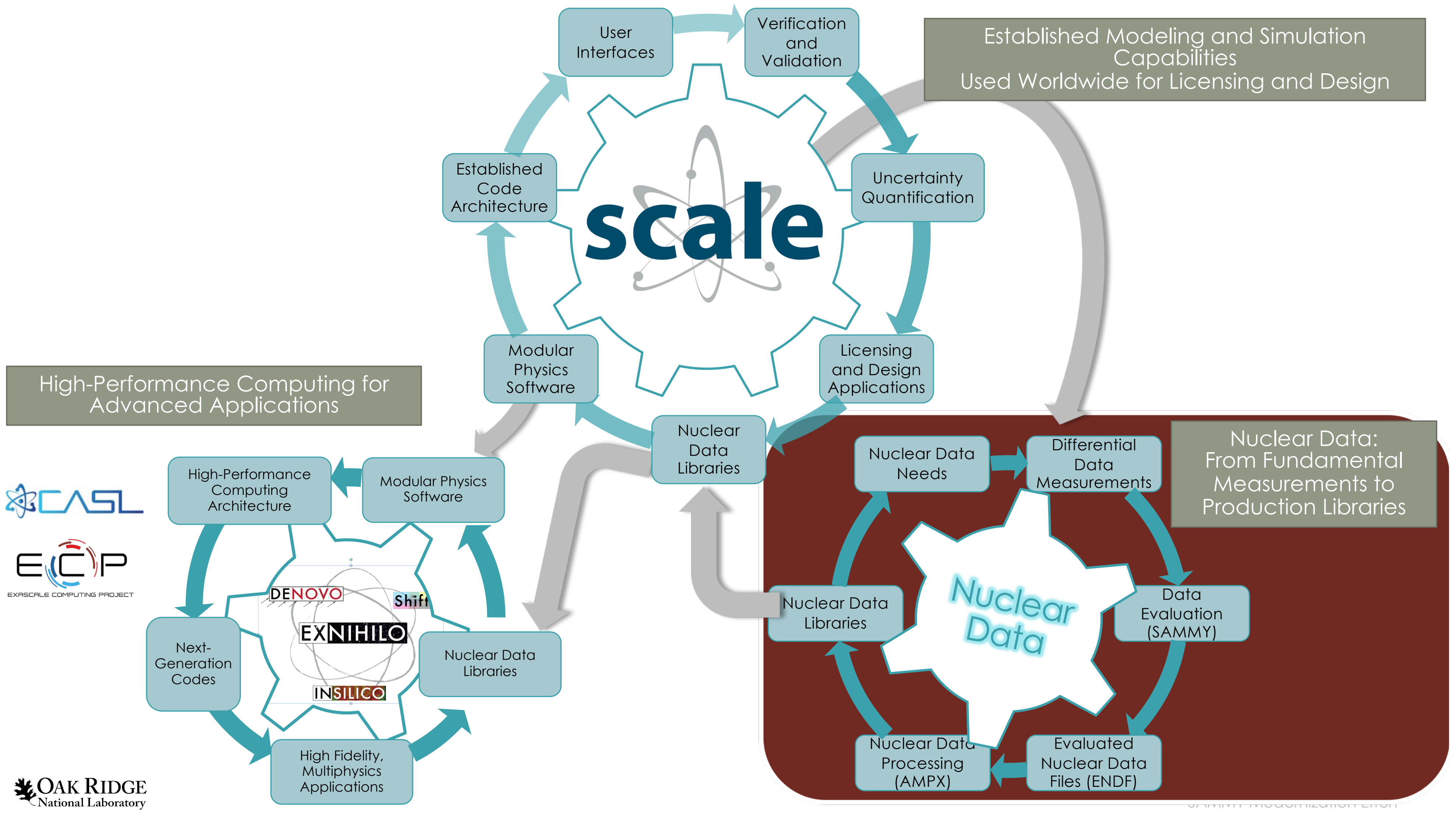
Oak Ridge National Laboratory

Consultancy Meeting on model code output & application nuclear data form structure

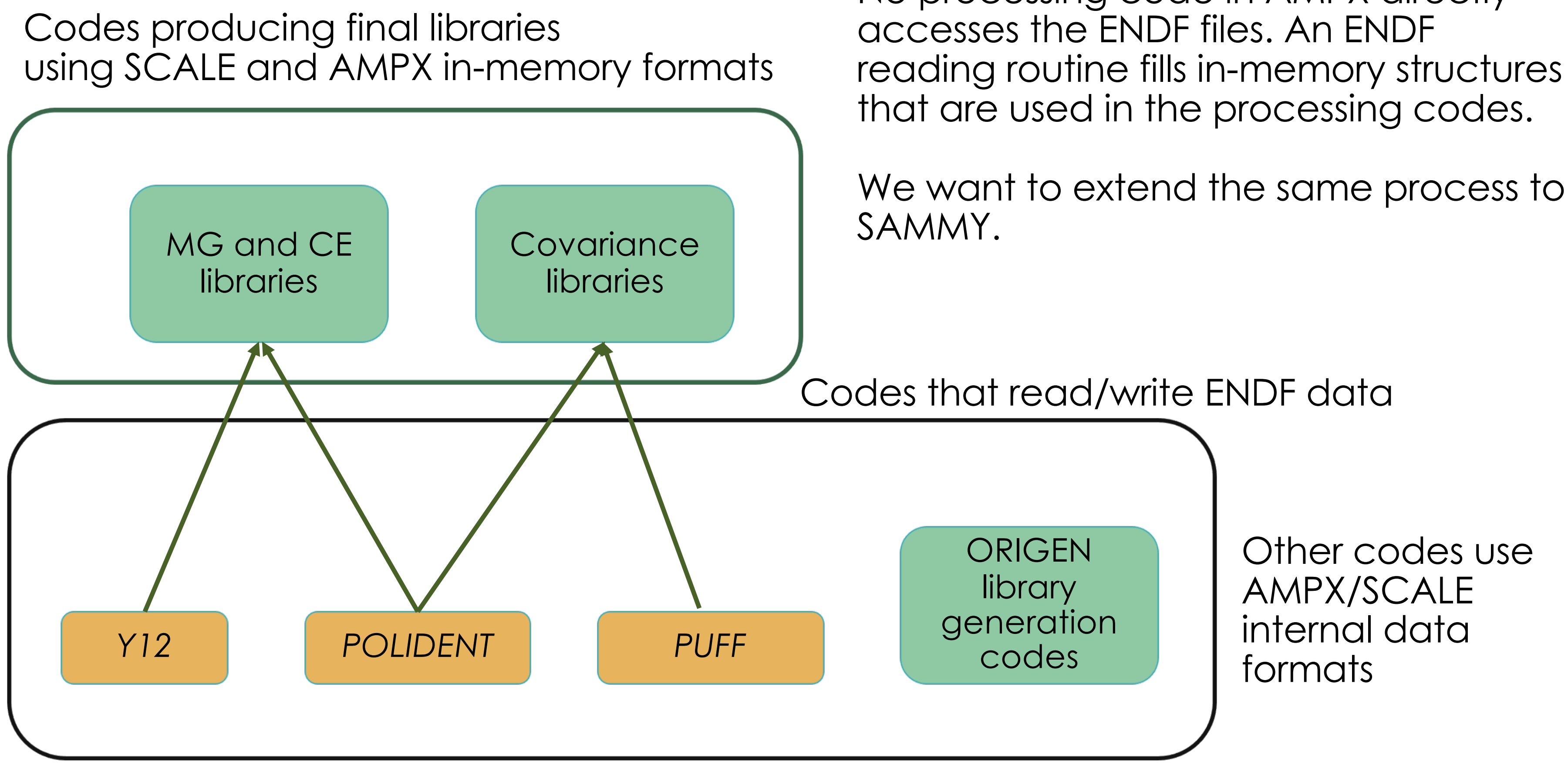
March 15, 2021

Summary

- Current status of ENDF handling
- Progress on GNDS in AMPX
- Current status of SAMMY
- Future plans for SAMMY input and output.


Established Modeling and Simulation Capabilities Used Worldwide for Licensing and Design
scale
User Interfaces
Verification and Validation
Uncertainty Quantification
Licensing and Design Applications
Nuclear Data Libraries
Modular Physics Software
Established Code Architecture
High-Performance Computing for Advanced Applications
High-Performance Computing Architecture
Modular Physics Software
Nuclear Data Libraries
High Fidelity, Multiphysics Applications
Next-Generation Codes
DENOVO
Shift
EXNIHILO
INSILICO
Nuclear Data: From Fundamental Measurements to Production Libraries
Nuclear Data
Nuclear Data Needs
Differential Data Measurements
Data Evaluation (SAMMY)
Evaluated Nuclear Data Files (ENDF)
Nuclear Data Processing (AMPX)
Nuclear Data Libraries
SAMMY Modernization Effort

ENDF data in AMPX


Codes producing final libraries using SCALE and AMPX in-memory formats
MG and CE libraries
Covariance libraries
Codes that read/write ENDF data
Y12
POLIDENT
PUFF
ORIGEN library generation codes
Other codes use AMPX/SCALE internal data formats

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.

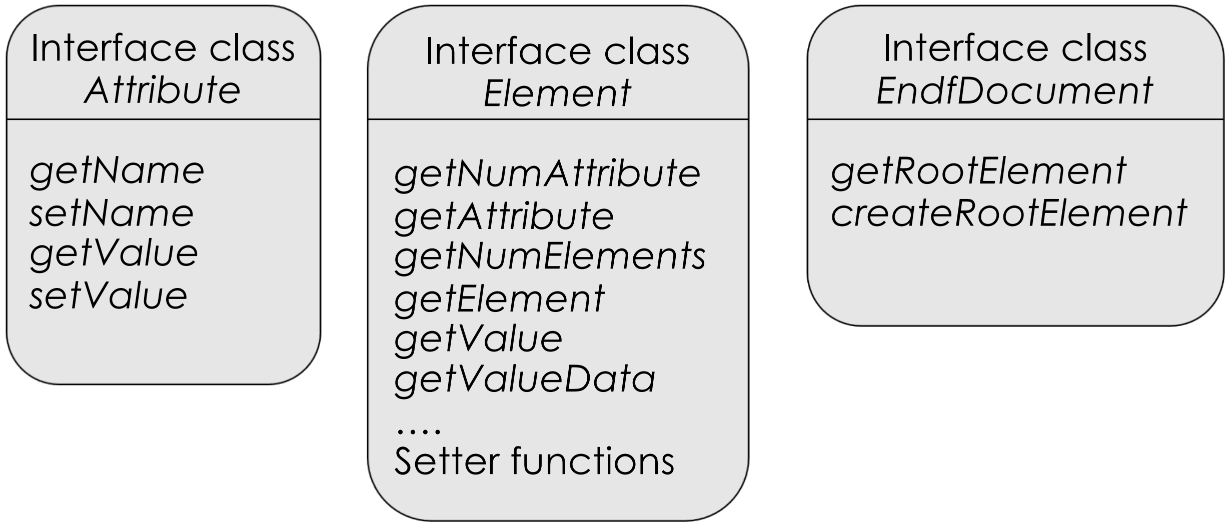
We want to extend the same process to SAMMY.

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: Part I

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



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

GNDS access layers in AMPX: Part II

Two common base classes:

Class
Container

getValueData
setValueData
getLabel
setLabel
readFromElement
saveToElement

Class GNDElement
inherits Container

Handles links and external file information

GNDS access layers in AMPX: Part III

- 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.

Once they have undergone final review, we will make them publicly available

GNDS access layers in AMPX: Part IV

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

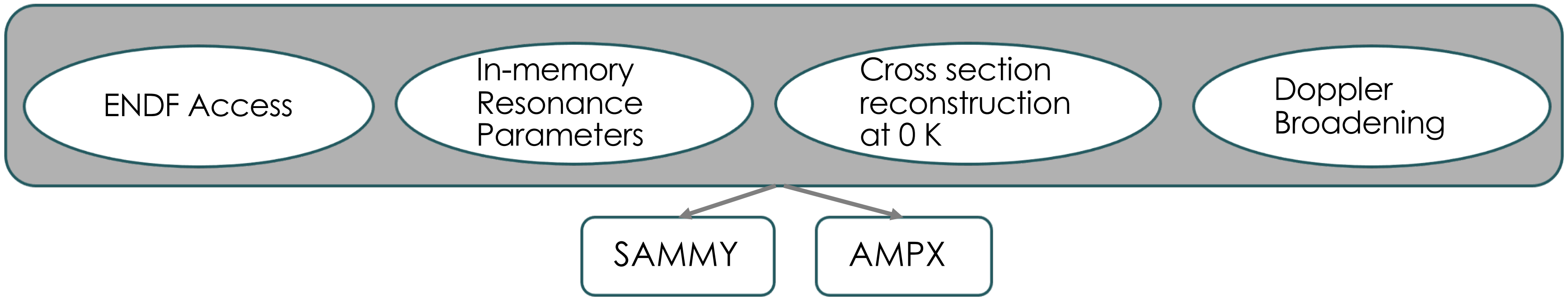
- To select the correct “style” of the data the user requested, which includes following the inheritance chain.
- Convert GNDS units to AMPX units
- Convert GNDS constructs into AMPX constructs.
- More user-friendly access methods to Particle data base

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.

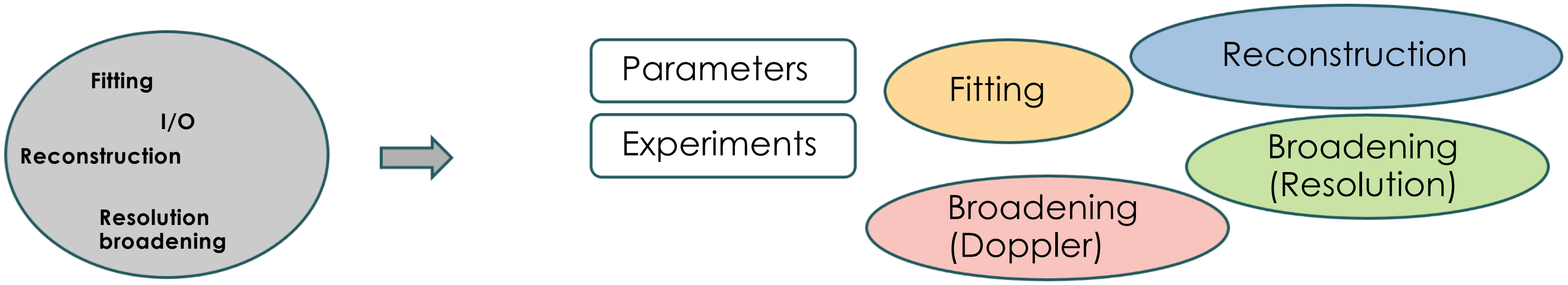
SAMMY Modernization and Maintenance goals

SAMMY is a code mainly written in F77. We plan to modernize as follows:

- Write code once and reuse

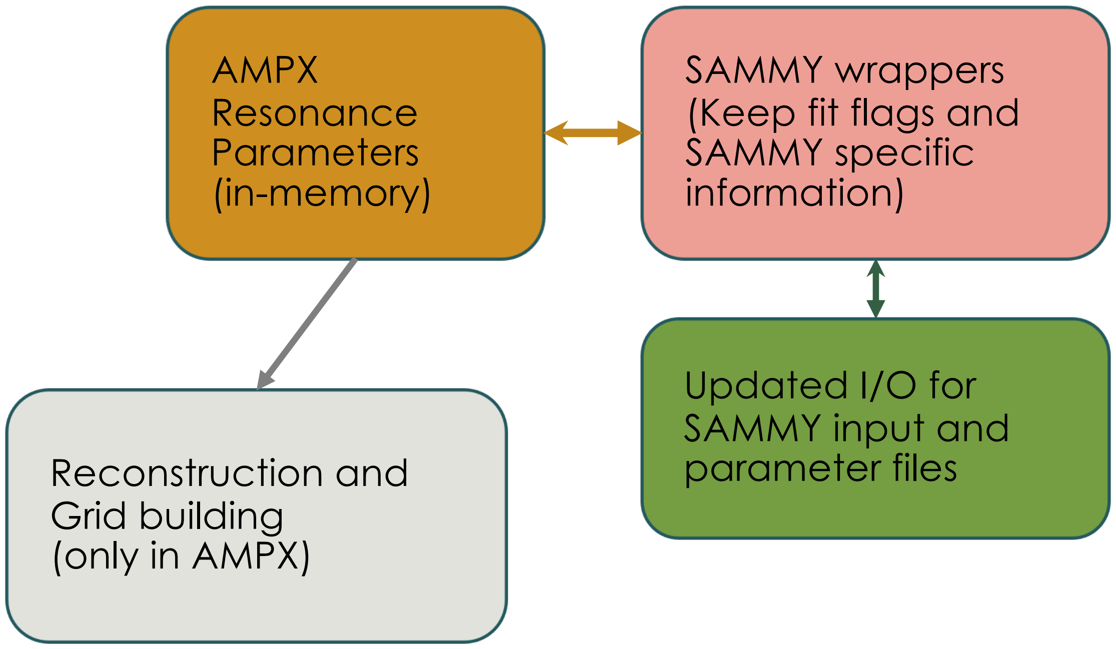

ENDF Access
In-memory Resonance Parameters
Cross section reconstruction at 0 K
Doppler Broadening
SAMMY
AMPX

- Transform SAMMY into a modular code, with independent modules with clear interfaces.



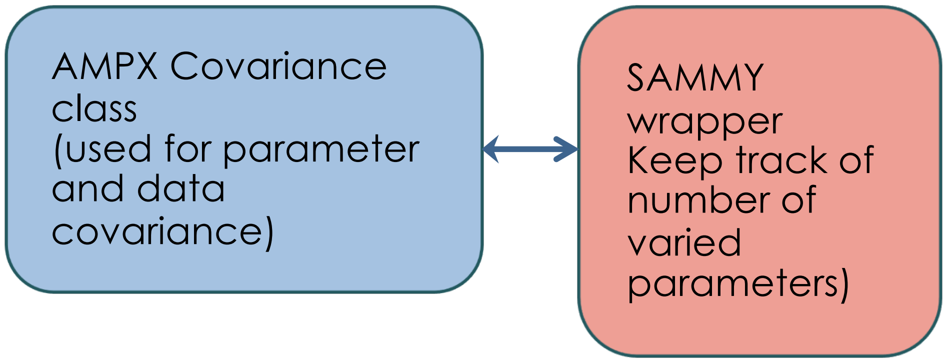
- Add new features, which is now easier as only the desired module needs to be changed.

Resonance parameters


AMPX Resonance Parameters (in-memory)
SAMMY wrappers (Keep fit flags and SAMMY specific information)
Reconstruction and Grid building (only in AMPX)
Updated I/O for SAMMY input and parameter files

- In-memory AMPX resonance parameters are used throughout SAMMY.
  - This is the reason that SAMMY needs to be linked to SCALE
- SAMMY specific wrappers give access to fit flags and the order of resonances in SAMMY.
  They are used throughout the code.
- Reading resonance parameter from parameter files is completely in C++.
  - Writing awaits consolidation of scratch file writing.
- AMPX and SAMMY reconstruction and grid creation are not yet combined. This is planned for a future update.

Covariance data


AMPX Covariance class (used for parameter and data covariance)
SAMMY wrapper Keep track of number of varied parameters)

- All covariance information is stored in-memory, data as well as parameter.
- All scratch files related to covariance information, except for the external SAMMY covariance files, have been eliminated.
- Current values of the varied parameters are also held in an in-memory class.

While all varied parameters and their current values are saved in a C++ in-memory class, SAMMY keeps track of the total number, the propagated uncertainty parameters (PUPs) and the types of parameters (resonance information, resolution broadening, ...) in global variables.

These numbers can (and on occasion do) get out of sync with the real numbers. While SAMMY stops if this happens, the user cannot remedy the problem short of not varying certain parameter. Usually, PUPs are involved.

These numbers will be consolidated into the above framework to ensure that they always keep in sync.

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.

The remainder of the talk will be about planned efforts: Consolidated input/output between SAMMY, AMPX and the wider data community. As such, we value input on the final formats, but would like to reuse elements from GNDS.

Theoretical and experimental data

- Currently SAMMY uses ODF files (an old ORELA style data format) both to read and store experimental and final adjusted data.
  - For user input and output they can be converted to ASCII listing, but internally ODF files are used.
- We would like to not use these any longer – therefore this will be the next enhancement to tackle.
- These are 1-D x-y style data or 2-D double differential style data. **Low level GNDS data containers are ideal to store these data.**
- The GNDS structure on top of these containers allows to indicate the type of cross section.
- Covariance information on these data can already be stored in GNDS (bin-width might have to be the same as the energy values).
- The use of appropriate style element will allow to save the resonance parameters as well as the reconstructed data (Fudge already does something similar) for easy inter-code comparison.

Resonance parameters

- Basic resonance parameters will be read/written into GNDS format. This is needed for the exchange of resonance evaluations with the data community.
- New format extensions will be tested with SAMMY and AMPX. This will allow testing in transport codes.
- In addition:
  We would like to use the GNDS resonance structure for normal SAMMY user input. But we need to determine where flags for adjusted and propagated uncertainty parameters (PUPs) will go.
  - Conversion from and to current input and par file structure will be provided.
  - New features will only be available in the new GNDS style input.

The hope is that we as a community develop tools that will allow to compare and visualize the parameters much easier. This of course includes flags and values pertinent to the type of resonance parameters and calculations.

Resolution broadening and experimental effects

- In order to replace the current SAMMY input with respect to parameter information we additionally need (all of which may have covariance information):
  - Doppler broadening parameters
  - Resolution broadening, for example: parameters for ORELA or RPI beam conditions
  - Sample information.
  - Linking between parameters (for example: all gamma widths of a spin-group should be adjusted together).
  - …
- We haven't decided yet what the best format is for these parameters.

Covariance information

We would like to store the covariance information from SAMMY exclusively into the GNDS format. Of course, conversion to current formats will be provided.

The format already allows to give the parameter covariance information for a wide variety of data containers. This will allow to provide covariance information for resonance parameters, doppler-broadening, resolution broadening, sample information, etc.

However, currently GNDS is ascii only, **for covariance information a binary option might be necessary** to preserve the precision.

Final outputs from a SAMMY evaluation

The final end-product of an evaluation will continue to be an ENDF file (GNDS or ENDF formatted) containing the resonance parameter and associated covariance.

The use of GNDS style input files for SAMMY, does not mean that these are ENDF libraries to be sent to the data centers, but rather a convenient input layer with re-usable data containers that are used in the data community.

It would be nice to capture as much as possible of the experimental data (possibly a link to the new EXFOR format (WPEC Subgroup 50), which are used in the evaluation) and some of the information that makes the evaluation easily reproducible, in connection with WPEC Subgroup 49.

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.

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