SAMMY Modernization Efforts

D. Wiarda

A. Holcomb

G. Arbanas

J. Brown

Oak Ridge National Laboratory

TPR, Feb. 2021

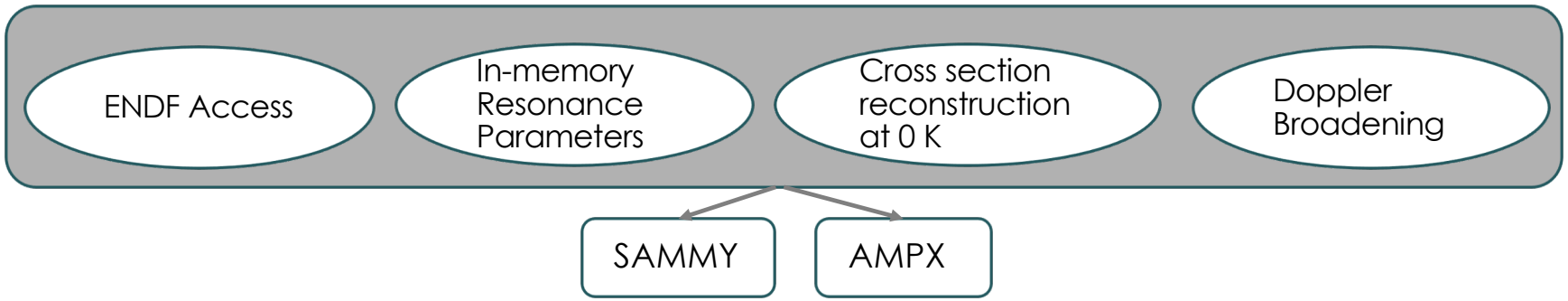
Summary

- SAMMY Modernization and Maintenance goals
- In-memory resources
  - Resonance parameters
  - Covariance information
  - Energy grids
- Move to more modern memory management
- SAMMY modularization and API
- Outlook

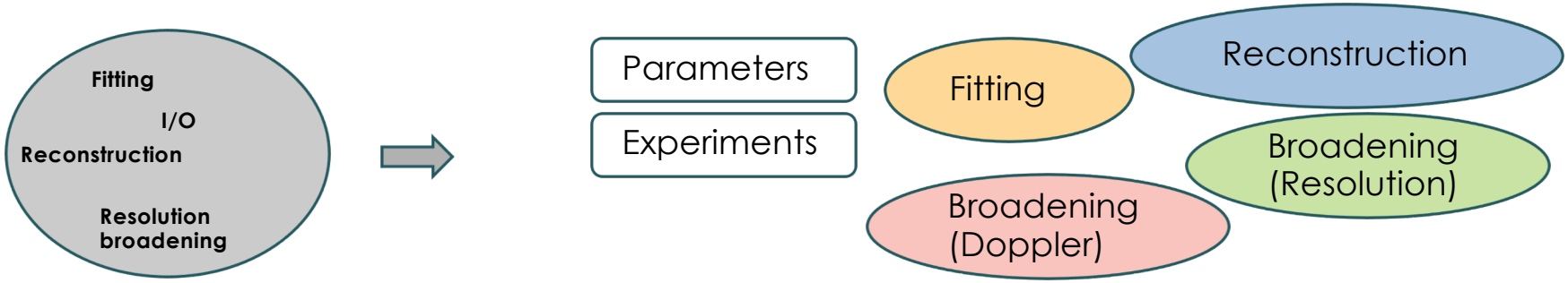
SAMMY Modernization and Maintenance goals

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

- Write code once and reuse

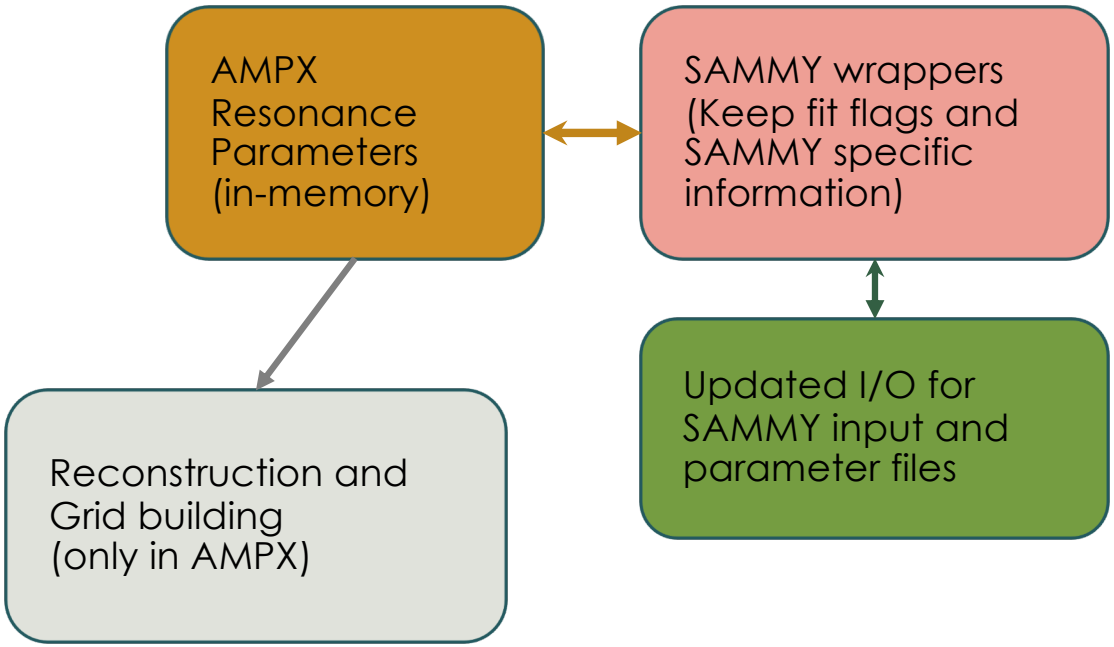


- Transform SAMMY into a modular code, with independent modules with clear interfaces.



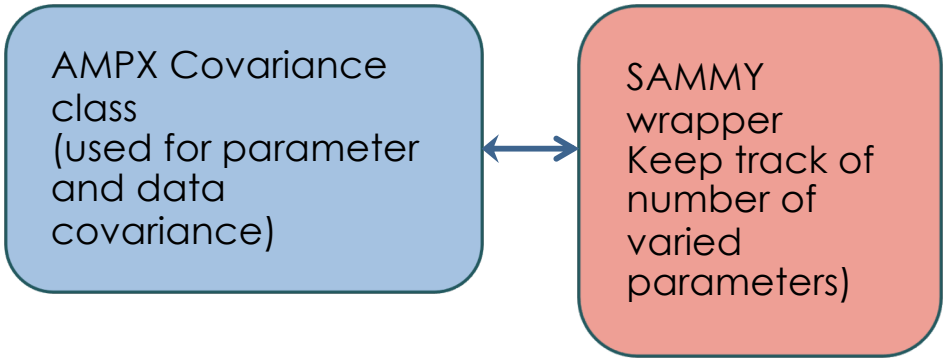
- Add new features, which is now easier as only the desired module needs to be changed.

Resonance parameters



- 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



- 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.

Energy grids in SAMMY

Energy grid C++ class

E_1	Exp. Data 1	Theory 1	$\frac{\partial \sigma}{\partial p_1}$	…
…	…	…	…	…
E_N	Exp. Data 2	Theory 2	$\frac{\partial \sigma}{\partial p_1}$	…

implemented

In progress

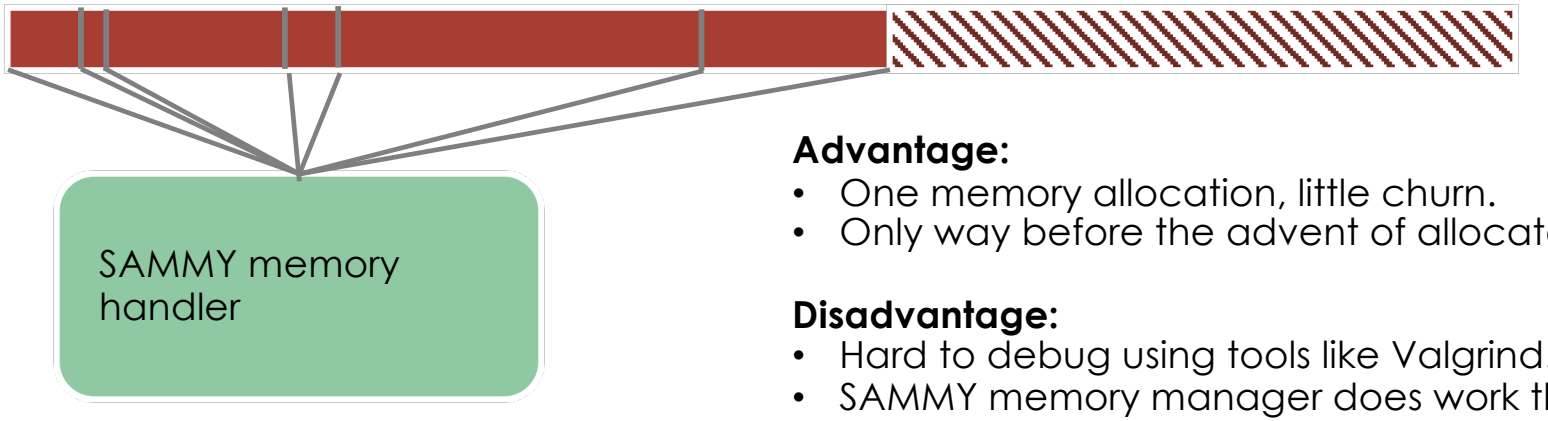
The energy grid was written and retrieved from scratch files. These scratch files have been eliminated

- SAMMY uses two energy grids:
  - Experimental data grid
  - Auxiliary grid used for resolution broadening
- Energy points for both grids now use the C++ class
- Switch to use derivatives in this class everywhere is ongoing, but they are used in all fitting routines.

Implicit data covariance

- Uncertainties and covariance information on implicit, i.e. not varied, parameters can contribute to the final data covariance and the value of the varied parameters.
- In SAMMY these can be either PUPs or user supplied sensitivities for external parameters with respect to the experimental data.
- PUPs covariance and sensitivities are now stored internally (similarly to varied parameters), eliminating the need for scratch files.
- External parameter information is read and checked on input and stored in in-memory C++ structures. These classes are used in all fitting routines, eliminating the need for scratch files.

Container array



Advantage:
- One memory allocation, little churn.
- Only way before the advent of allocate

Disadvantage:
- Hard to debug using tools like Valgrind.
- SAMMY memory manager does work the operating system and compiler can do
- Array is fixed and needs to be dimensioned for largest desired problems.

Unfortunately, we can't just search and replace all instances of the instances of access to the SAMMY memory manger with allocate/deallocate.
There are a lot of overstepped bounds, re-use of arrays and other access violations. Therefore, this is a highly manual process.

Removal of the container array

- A lot of the difficulties in removing the container array was due to the inconsistencies in reading and writing experimental data, parameters and covariance information to scratch files.
- Since these parameters have now been moved to in-memory storage, removal of the container array was possible.
- Historical errors were discovered and fixed in the process.

The container array has been eliminated from SAMMY; modern debugging methods are now possible. Number of parameters and experimental data are now limited by the machine on which SAMMY is run and not by the size of the initial container array.

API to SAMMY for external access - in progress

sammy_init
read normal input

sammy_get_theory
allow external change of parameters

sammy_get_covariance_data
get adjusted parameters (to allow external changes)

sammy_get_exp_grid
get experimental and theory data

- We are in the process of adding an initial API to SAMMY, accessible from C++ and Fortran
- This allows external changes to the adjusted and resonance parameters during a SAMMY run.
- We started to modularize SAMMY's internal fitting routines, so they can be used outside of SAMMY.
- Allows coupling to external fitting routines, see Jesse Brown's talk.

Please note that this API is for initial testing, therefore function names are subject to change. The implementation of the *Model* interface in our *fitAPI* package internally uses these functions and has a more user-friendly function structure.

Fixed production defects

Removal of the container array and consolidation of the covariance information uncovered several historical errors:

- If using two resolution functions, an array was dimensioned too small. This led to the wrong value for the resolution broadened cross section.
- In order to allow for a faster conversion between angular cross section given in the laboratory frame and the center of mass frame, SAMMY uses pre-calculated conversion factors, that are calculated for the exit channels. If there are more exit channels with distinct Q-values than isotopes, the arrays were dimensioned incorrectly.
- If all gamma widths of a given spin-group are adjusted as one parameter instead of individual parameters, a user could easily input conflicting information, leading to overstepped array bounds. The input is now corrected, and the user is informed about the action taken.
- Fixed a production error where omitting spin groups from the fit did not work correctly if using reduced width in the parameter file written by SAMMY.

SAMMY 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 and writing GNDS formatted ENDF files.
- Switching to these routines will bring GNDS support to SAMMY

Future updates to SAMMY

- Continue the modularization and consolidation of duplicate code
- Switch to AMPX ENDF reading/writing routines
- Use the same code to reconstruct cross section data at 0 K in SAMMY and AMPX.
- Introduce imaginary component to *R*-matrix channel radii to parameterize eliminated *direct* capture channels in Reich-Moore approximation
- Simultaneous fitting of multiple targets and compound systems
- New parameter fitting methods would quantify the effects of assuming linear models and normal PDFs.
- Any new GNDS formats will be implemented in SAMMY, as needed, and then proposed to the EG-GNDS group.

SAMMY 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.
- Stable versions will be tagged and distributed as before