Recent Developments in the R-Matrix Code SAMMY

D. Wiarda, G. Arbanas, J. Brown, M. Pigni, J. McDonnell, C. Chapman

Oak Ridge National Laboratory

TPR, Dec. 2024

Summary

- Sequential fits of different experiments
- Input reading streamlining
- Summary

Desired: Sequential fit for different experiments

Covariance matrix

Resonance parameters (RR and/or URR) Propagated through all steps	**Correlations** set to 0 after each step
Correlations set to 0 after each step	**Experimental Correction** (Normalization, Resolution, etc)

- Sequentially adjust resonance parameters only
- Allow different experimental set-ups and reactions for each step. (Adjusted sequentially only if the same experiment is used in a subsequent fit).
- Ignore correlation between experiment and resonance parameters.

  (This makes it easier on the user. If needed it can be added and used for Bayesian adjustment algorithms, but likely restrictions on the use will have to be applied)

Needed for: Sequential fit for different experiments

A way to access/store parameter and covariance information for experimental corrections.

Solution:

- New classes
  - For each type of experimental correction (broadening, normalization, resolution functions, …)

Number of parameters
Parameters
Adjustment flags

- New h5 covariance file for experimental corrections and their covariance information
- Upon reading the two covariance files (from previous fit and experimental corrections (subset or all):
  - Update the general covariance information, dropping information as needed. Update indices as needed
- All types of parameters are allowed, including pup'ed (adjusted and propagated uncertainty) parameters

Sequential fit of different experiments

- New options in the input or parameter file (ExpCorrData is user defined name of the h5 file to save the experimental correction parameters):
  AdjustableObject to be saved = ExpCorrData
  AdjustableObject to be restored = ExpCorrData
  AdjustableObject to be dropped
  Each taking a list of experimental corrections to be used (same header as in original input or parameter file definitions)

- Adjustable parameters can now be marked as adjustable and/or pup'ed in the input or parameter file.

- Parameters in either of the covariance files take precedence over input and parameter file, except if explicitly dropped via
  AdjustableObject to be dropped
  or listed in
  AdjustableObject to be restored

- **Available in the public facing SAMMY repository**

On the path to a nicer SAMMY API

Ideally an API to SAMMY should be as simple as:

1. Set up -> read input, parameter and covariance data
   - Currently done in every intermediate step
2. Calculate the theoretical data (including corrections)
   - Already separated in a dedicated call – but uses some global parameters
3. Adjustment step (or possible external fitting routine like Minuit).
   - Separated in dedicated call – but includes final and intermediate print-outs
4. Set updated parameters
   - Currently interwoven with output and writing of new input and covariance files
5. Repeat step 2 -4 as needed
6. Output final results and covariance files
   - See above, as it is interwoven with step 4

On the path to a nicer SAMMY API: Input

- As reported last year, we moved some input handling into a C++ class - this year many more parameters have been moved to this class.
  - Advantages:
    - Consolidate the input parameters in one place
    - Easier way to add new user options:
      Examples: Inverse channel and new SESH options
    - Ensure that the input is treated consistently
      Example: The same options are now available in angodf as in SAMMY
- Input for the URR is still mostly left as is, except for alphanumerical Card 3 input.
- Down the road input values (alphanumeric input) can be set/read via an external interface (python).

  All alphanumeric input options have a distinct ID in SAMMY. The current input class now already handles all the user input via these IDs, but not all can be accessed via convenient function names.

On the path to a nicer SAMMY API: Input (cont.)

Previously the input and parameter files were read several times:

- On startup to decide which type of run and options are desired
- On each internal run (one for each adjustment step, and one for the last output):
  - Once to determine the sizes and numbers of parameters
  - Once to actually read the data

Now

- Input and parameter file are read only once at the beginning of the run
  - This is possible due to the new input class and the new classes for the experimental corrections
- Global parameters related to input are set once at the beginning

To do:

- A new input and parameter file is still written after each intermediate step
- Global parameter are still used for many tasks.

On the path to a nicer SAMMY API: To do

- Consolidate the reading of the data files, so it is only read once.
  - In progress
- Ensure the covariance file is not read on intermediate runs, as the data are in memory
  - In progress
- Update the handling of the URR, so that URR and RR can be adjusted together
- Eliminate the need for global parameters

Conclusion

- SAMMY is available from https://code.ornl.gov/RNSD/SAMMY Versions will be tagged as needed.
- Support for easy switching of experimental corrections in sequential fits
- Improved program flow.

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