Documentation for `retroactipy`

Project in *Practical python programming for Big Data and the scientist*

Petter Helgesson

Contents

1 Introduction

1.1 Nuclear reaction cross sections and resonance parameters

All nuclides existing in nature are associated with certain probabilities for certain nuclear reactions to occur, which are quantified by *cross sections*. Easily put, the cross section describes the effective cross section of the nuclide that an incident particle of a certain energy "sees", times the probability for the considered reaction to occur. For many applications, incident neutrons are of the main interest, and this work is limited to incident neutrons.

Due to quantum mechanical effects, cross sections often vary very rapidly with energy at lower neutron energies (below $\lesssim 100\,\mathrm{keV}$), and in this energy range the cross sections are well described by *resonance parameters*. Resonance parameters may provide a more compact storage than the very dense energy grid which is otherwise necessary, and can also provide some understanding of the physics. They are also used "directly" for some applications.

1.2 The retroactive method

When determining resonances, one uses experiments that determine quantities quite closely related to cross sections, and then fit resonance parameters to this. For many older experiments, only fitted resonance parameters are reported. They may be provided with uncertainty estimates of the individual parameters, but not with information on correlations between these parameters.

Therefore, the following idea, called the retroactive method, is outlined in the manual of the resonance parameter fitting code SAMMY [1]:

- Reconstruct pseudo-experimental cross section data from resonance parameters (can also be done with SAMMY)
- Assign experimental uncertainty information to this data, guesstimated based on knowledge of the experiment – this include both statistical and systematic uncertainties!
- Redo the fit, and include correlations this time.

To my knowledge, there is no available and correct tool for this method.

1.3 Aim

A data processing tool which implements the basic ideas of the retroactive method using SAMMY. In this project, I limit the work to a case with a very simple guesstimate of the experimental covariances – this should be left easily generalizable such that the tool can be developed to be truly useful.

2 What `retroactipy` does

The tool takes an Evaluated Nuclear Data Format (ENDF, [2]) file, which includes the definition of a nuclide. For this nuclide, it finds resonance parameters from the ENDF file, and then uses the program SAMMY to reconstruct pseudo-experimental cross section data from these resonance parameters. The tool then adds uncertainty information to the data (this is done in a general way such that it later can be further developed), and then SAMMY is used again, iteratively, to fit the resonance parameters. The output is ENDF formatted files with the resulting resonance parameters and covariances.

Thus:

- Input: ENDF file including resonance parameters
- Output: ENDF files with new collection of resonance parameters and covariance information.

The tool is limited to nuclides with only the (n,el) and (n,γ) channels open at low energies. It is straightforward to generalize to include fission (but nothing else such as (n,α) due to format limitations).

2.1 Why useful?

For most nuclides, it would, in my opinion, be possible to get a better guess on the covariance matrix for the resonance parameters than the currently existing information used in uncertainty estimates for applications of neutron transport, *e.g.*, nuclear power, irradiation safety, or medicine (slow moving neutrons in resonance range – not so interesting for weapons :D). This can therefore improve the reliability of the estimates of uncertainties of computed quantities related to these applications.

3 How to run `retroactipy`

The `Main` class takes the path of an ENDF file and sets up a fictive experiment given the resonance parameters contained in this file. Then, `self.fit(out_paths)` can be called to finalize the retroactive method by fitting parameters the fictive experiment. The resulting `ResonanceFile` instance is stored as `self.resonance_out` and the resulting files are stored at the entries of `out_paths` (list of length 2).

3.1 Example

```
>> import retroactipy as ret
>> main = ret.Main('Ni059.endfb71')
>> main.experiment.plot(c = 'k') # Plot the fictive experiment
>> main.fit(['out.mf2', 'out.mf32'])
>> self.resonance_out.path
   '/home/petter/pythonWork/course/retroactipy/out.mf2'
>> self.resonance_out.cov.path
   '/home/petter/pythonWork/course/retroactipy/out.mf32'
>> main.resonance_out.plot(c = 'r') # Plot cross sections from resulting parameters
>>
```

The resulting ENDF formatted resonance and resonance covariance files are stored at `out.mf2` and `out.mf32`, respectively.

3.2 Modifying the method

The initialization of a `Main` object stores the fictive experiment as `self.experiment` and creates a `SammyRunner` instance `self.sr`. These attributes may be modified to change assumptions about the experiment or settings in running SAMMY. For example, `self.sr.cleanup = False` will result in keeping all temporary files from the SAMMY runs (can be a few hundred MB). Some more details are found in the descriptions of the `PseudoExperiment` and `SammyRunner` classes.

4 Classes

Only the most imortant attributes and methods are mentioned below.

4.1 `Main`

"God-ish class" which can be used as a user interface. The initialization takes an ENDF file, sets up resonance file (`self.resonance_file`), creates a `SammyRunner` instance (`self.sr`) and a `PseudoExperiment` instance (`self.experiment`) using `self.resonance_file`. `self.sr` or `self.experiment` may be modified (to change settings). Then, `self.fit(out_paths)` can be called to finalize the retroactive method by fitting parameters to `self.experiment` using `self.resonance_file` as starting guess. The resulting `ResonanceFile` instance is stored as `self.resonance_out` and the resulting files are stored at the entries of `out_paths` (list of length 2).

Look at the example in Sec. 3.1.

- Attributes
  - `endf_file`: `EndfFile` object
  - `resonance_file`: a `ResonanceFile` instance constructed from `endf_file`
  - `experiment`: a `PseudoExperiment` instance
  - `sr`: a `SammyRunner` instance
  - `resonance_out`: a `ResonanceFile` instance (after `self.fit()` has been called)
- Methods
  - `fit(self, out_paths)`:

4.2 `EndfFile`

Class describing `EndfFile` objects. Only contains a path, which is error checked at initialization. The class could be expanded with attributes and methods that are common for all ENDF formatted files.

4.3 `CompleteEndfFile(EndfFile)`

Class describing the subset of ENDF files which are complete.

- Methods
  - `create_resonances(self, path, pathcov)`: creates (and returns) `ResonanceFile` instance and produces the corresponding file(s) containing the subsection of the ENDF file with resonance parameters and (if available) covariances.
  - `create_nuclide(self, path)`: parses head of the file, returns `Nuclide` instance

4.4 `ResonanceFile(EndfFile)`

Class describing ENDF formatted files which contain the resonance part only (`MF = 2`, `MT = 151`), the file can be created by method in the `CompleteEndfFile` class.

4.5 `ResonanceCovFile(EndfFile)`

Class describing ENDF formatted files which contain the resonance covariance part only (`MF = 32`, `MT = 151`), the file can be created by method in thev `CompleteEndfFile` class. More methods may be added (quite useless at the moment).

4.6 `Nuclide`

Class describing the basic features of a nuclide.

4.7 `CrossSection`

Class describing a cross section. If owned by an experiment, it is provided with uncertainty information in the array `statistical_unc` and in the dictionary `unc` and knowledge of its owner.

4.8 `PseudoExperiment`

Class describing a set of pseudo-experimental cross section data. This includes `CrossSection` objects including uncertainty information, and can include some more information about the imagined experiment.

- Attributes
  - `cross_sections`: list of `CrossSection` instances. These contain uncertainty information.
  - `thickness`: sample thickness

4.9 `EndfLineParser`

Class which is initialized using a line from an ENDF file, and then can be used to call for certain parts of that line, using square brackets (thorough `__getitem__()`). Indices of normally formatted lines (or oddly formatted using `self.breaks`) can be used, or certain keywords. Lists combining these two, and slices, are also allowed.

Possible keywords:

- 'mf' or 'file' for mf number
- 'mt' or 'tape' for mt number
- Anything starting with 'mat' for material number
- Anything starting with 'num' or 'nbr' for 'record number' (ENDF jargon)

4.10 `SammyRunner`

Class collecting methods and settings (attributes) for running SAMMY. Structure-wise uninteresting, but handles the most important methods.

4.10.1 Attributes

- `executable`: SAMMY executable
- `data_format = 'twenty'`: Format used in reading writing data
- `temperature = 0.`: Default experiment temperature
- `default_thickness = 0.003`: Sample thickness if not in experiment
- `include_correlations = True` : If systematic uncertainties should be treated
- `cleanup = True`: Should temporary files be removed?
- `convergence_dict`: `dict` with convergence criteria used by `fit()`

4.10.2 `reconstruct()`

Reconstructs the cross sections using parameters in `resonance_file` object at the provided energies, and using the "settings" of the `SammyRunner` object. As of now, the reconstructions are done at SAMMYs default energies and interpolated, this should be changed and should enable Doppler broadening (`self.temperature > 0.`). Fission shall be added.

4.10.3 `fit()`

Produces SAMMY input and runs SAMMY "without prior", returns a new ResonanceFile instance with resonance parameters fit to the pseudo-experiment. The fit is so far simplified such that "true GLSQ" is used for (n,tot) and then Bayesian updating is used for (n,gamma). This would be equivalent to GLSQ all the way if there were no non-linearities and no correlations between the different reactions

5 Fictitious user case

"Bob" is a licensed SAMMY user. He wants to estimate the uncertainties in certain safety parameters resulting from a simulation of a thermal nuclear reactor, which will be very sensitive to the resonance parameters of the nuclides involved in the simulation. He is aware that the lack of correlations in the uncertainty information for the resonance parameters in the ENDF files he plans to use may yield strongly underestimated propagated uncertainties. Therefore, he is happy that he can apply `retroactipy` to the nuclides of interest! After doing so, the new ENDF files including the newly estimated covariances can be made available for others to use.

References

[1] N. Larson, Updated User's Guide for SAMMY: Multilevel R-matrix fits to neutron data using Bayes' Equations, ORNL (October 2008).

[2] A. Trkov, M. Herman, D. Brown, et al., ENDF-6 formats manual, Tech. Rep. BNL-990365-2009, Brookhaven National Laboratory (2011).