SrMise 0.3a2 User Guide

1 Introduction

SrMise is a program for automated peak extraction from the pair distribution function (PDF) designed to leverage the Akaike Information Criterion (AIC) for information theoretic multimodeling. SrMise provides a command-line interface to many features, but is primarily intended as a library for use with scripting or other programs.

If you use this program to do productive scientific research that leads to publication, we ask that you acknowledge use of the program by citing the following paper in your publication:

> L. Granlund, S.J.B. Billinge, P.D. Duxbury, Automated Peak Extraction from Atomic Pair Distribution Functions, Acta Crystallographica A. Submitted (2014)

2 License

SrMise Copyright 2014, Board of Trustees of Michigan State University

For more information please email Luke Granlund at luke.r.granlund@gmail.com

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDER "AS IS". COPYRIGHT HOLDER EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, TITLE, FITNESS, ADEQUACY OR SUITABILITY FOR A PARTICULAR PURPOSE, AND ANY WARRANTIES OF FREEDOM FROM INFRINGEMENT OF ANY DOMESTIC OR

FOREIGN PATENT, COPYRIGHTS, TRADE SECRETS OR OTHER PROPRIETARY RIGHTS OF ANY PARTY. IN NO EVENT SHALL COPYRIGHT HOLDER BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE OR RELATING TO THIS AGREEMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

3 Installation

SrMise requires Python 2.6-2.7 (`http://www.python.org`) and is available at `http://www.diffpy.org`.

To install SrMise simply use the easy_install executable provided by setuptools (`http://pypi.python.org/pypi/setuptools`):
easy_install diffpy.srmise-version.tar.gz (from file) or
easy_install diffpy.srmise (from PyPI).

A more modern alternative to setuptools is the pip package (`https://pypi.python.org/pypi/pip`). The fastest way to install pip is downloading and running `https://bootstrap.pypa.io/get-pip.py`. Basic installation of SrMise is as above, using *pip* rather than *easy_install*. Advanced installation options are described in the documentation for setuptools and pip.

SrMise also requires matplotlib (`http://www.matplotlib.org`), numpy/scipy (`http://www.scipy.org`), and PDFgui (`http://www.diffpy.org`). If connected to the internet, *easy_install* or *pip* will attempt to automatically download and install these dependencies.

Windows note: It is highly recommended that Windows users obtain the installable binaries for these dependencies before installing SrMise, as compiling them on that platform tends to be laborious. Details on Windows installation are included in install.txt.

4 Quick Start

Once installed, SrMise is available from the command-line by running *srmise*. This command-line interface exposes many common features, but does not expose access to advanced tools such as multimodel analysis. Full details about the command-line tool may be find by running *s*rmise --help.

The only required input is a file containing a PDF, but a typical run might appear as below:
srmise somepdf.gr --range 1.5 10 --bcrystal .084 --dg 0.3 --save somepdf.srmise --plot
This specifies the region over which to perform extraction, defines a specific crystal baseline to use, specifies that each data point in the PDF should be assumed to have a y-axis uncertainty of 0.3 (in the units of the input file), saves the results to a file, and plots the extracted peaks before exiting. The uncertainty in the PDF produced by some data reduction tools can be unreliably small for certain kinds of detectors, so specifying a physically reasonable uncertainty with *--dg number* may be necessary.

Somewhat nicer plots may be made from existing SrMise files with a second command-line utility. Basic usage is *srmiseplot somefile.srmise --show*.

5 Approach to Peak Extraction

SrMise uses a clustering method as a framework to find peak-like regions within the PDF. Initially each cluster is used to provisionally identify at most one peak within it. This function is typically a Gaussian divided by radius r. As the algorithm progresses these clusters are carefully combined using a recursive call to find obscured peaks in model residuals. The eventual result is a global cluster containing a model of many peaks. A greedy pruning subroutine attempts to remove the least justified peaks, as identified with the Akaike Information Criterion. Before completion termination ripples are applied to the modeled peaks during a modified pruning step to reduce the likelihood of extracting physically spurious peaks. Finally, the PDF baseline, which must be estimated or known before extraction, is fit along with extracted peaks.

6 Scripting

Several scripting examples are included with the distribution. These include extracting a single model from experimental PDFs for C_{60} fullerenes in an FCC lattice, crystalline $SrTiO_3$, and caffeine. In addition, the basics of AIC-based multimodeling is demonstrated on the fullerene. These examples have been chosen to demonstrate a wide range of SrMise options, and use settings that are not necessarily appropriate to a careful scientific study. In particular, PDFs such as caffeine with very few distinct peaks are generally not suitable for analysis with SrMise.

6.1 Peak Extraction

If using SrMise for peak extraction from a PDF, the script should import and instantiate the appropriate class, for example

```
from diffpy.srmise.mise import PDFPeakExtraction
```

```
ppe = PDFPeakExtraction()
```

There are two ways to load a PDF. To read from an existing SrMise file, simply use

```
ppe.read("somefile.srmise")
```

but to read from a PDF instead use

```
ppe.loadpdf("somepdf.gr").
```

The loadpdf function can read any PDF readable by PDFgui.

Next, extraction variables for the fit should be defined. These include the range of extraction, the PDF baseline, etc. Variables from an existing SrMise file are maintained, and some may be determined from a PDF file. Those that have not been defined are given a default value before extraction. One pattern for defining extraction variables is

```
kwds = {}
kwds["rng"] = [1., 8.]
kwds["nyquist"] = True
...
ppe.setvars(**kwds)
```

The primary extraction options are below. The complete list is found in the PDFPeakExtraction.setvars docstring.

The PDF baseline is effectively subtracted from the data before extraction, and then fit along with the extracted peaks at the end. Thus, the quality of extracted peaks are conditioned on the quality of the baseline, but this can be difficult to estimate. Physically, the origin of the PDF baseline is small-angle scattering that is lost, so a high-quality baseline estimate generally requires at least some *a priori* information about the material's structure. Moreover, interparticle correlation, etc. may mean some contribution to the PDF should be excluded before peak extraction, and SrMise treats this as part of the baseline as well. At present, SrMise has only rudimentary support for estimating baselines, and the user should generally expect to expend some effort to find a reasonable one.

The `diffpy.srmise.mise.baselines` module implements baselines in the following way. Each type of baseline has a corresponding class which inherits from `baseline.base.BaselineFunction`, and these define the form of the baseline (e.g. polynomial, spherical nanoparticle) but not the numerical parameters necessary for a calculation. However, these classes implement the `actualize()` method, which returns a `baselines.base.Baseline` instance that encapsulates both the form of baseline as well as the numerical parameters necessary to actually perform calculations.

Thus, there are two standard ways to assign a baseline. First, one can pass an instance of a class which inherits from `BaselineFunction` as an extraction parameter and implements parameter estimation. For example,

```
from diffpy.srmise.mise.baselines import Polynomial
blf = Polynomial(degree=1)
ppe.setvars(baseline=blf)
```

Variable	Summary	Default
baseline	The PDF baseline. See below.	Crystalline. See below.
cres	Clustering resolution.	Nyquist rate if $Q_{\max} > 0$. Otherwise 4 times the mean sample spacing.
dg	The uncertainty in G(r) to use during extraction. See below.	Uncertainty reported by PDF, otherwise 5% of difference from greatest to least value.
error_method	Class from the modelevaluators module.	AIC
initial_peaks	A Peaks instance. All peaks in this instance are present at the start of extraction.	An empty Peaks instance.
nyquist	Boolean indicating whether to use Nyquist sampling.	True of $Q_{\max} > 0$, otherwise False.
pf	Sequence of PeakFunctionBase subclass instances. At present, only the first element in the sequence is ever used.	[GaussianOverR(0.7)], a Gaussian divided by r with max fwhm width of 0.7 in units of x-axis (usually Å).
qmax	Maximum momentum transfer.	The $Q_{\max}$ value determined algorithmically from data, and if that fails the value that reported in the original PDF. Otherwise, 0.
rng	Two-element sequence of range on the horizontal axis over which to perform extraction.	The first and last points in data.

The result of the estimation routine will be used to construct an appropriate `BaselineFunction` instance. If the class implements no estimation routine, or the routine fails, the baseline is identically 0 and peak extraction will likely produce nonsense.

More commonly, one calls `actualize()` with appropriate parameters instead. For example, the baseline $3r^3 + r + 2$ we be obtained with

```
from diffpy.srmise.mise.baselines import Polynomial
blf = Polynomial(degree=3)
bl = blf.actualize([3, 0, 1, 2])
ppe.setvars(baseline=bl)
```

If no baseline is set by the user, SrMise attempts to estimate a crystalline baseline, namely a degree one polynomial with origin fixed at 0.

A summary of the available baselines is given below. At this time baseline estimation exists only for low-degree polynomials, and the method is not robust. If the approximate form of the baseline is known, an alternate way to find the baseline quickly is to supply a guess and then perform extraction assuming unphysically large δg. This will extract only the few most prominent peaks, and hopefully allow the baseline freedom to adjust. If the trial is saved the fit baseline will be available for future use.

Baseline	**Summary**	**Estimation**
Arbitrary	Arbitrary user-supplied function for baseline values and/or estimation. Does not support serialization.	Optional.
FromSequence	Baseline from arbitrary list of coordinates. Can be evaluated everywhere in domain via unsmoothed cubic spline interpolation. No free parameters.	No.
NanoSpherical	Baseline from nanoparticle form factor for sphere of uniform density (Guinier et al. 1955). Parameters are scale factor and radius.	No.
Polynomial	An arbitrary polynomial of degree n. If $n < 0$ there are no free parameters and baseline is identically 0. If $n \geq 0$ there are $n + 1$ parameters, the coefficients of the polynomial.	$n \leq 1$

Setting the assumed uncertainty δg for peak extraction is important. This can be given as a positive number, or as a sequence of positive numbers of appropriate length. Unfortunately, the reported PDF uncertainty from some data reduction is not trustworthy, in which case physically plausible guess must be made. As a rule of thumb, a typical PDF has an uncertainty of “a few percent.” Keep in mind that even if the statistical uncertainty is small, the data may simply not support enough parameters to fit well. In that case errors from model misspecification will dominate the residual and SrMise may take a long time to obtain a rather poor result. In such a case, increasing the assumed uncertainty should at least result in a usable fit, and alert the investigator that the extracted peaks are likely a composite of many underlying peaks. One advantage of using the Akaike Information Criterion, however, is that it does not assume the “true model” is available to the investigator, and so some misspecification is well tolerated.

In setting the assumed uncertainty, it may be convenient to perform calculations using the raw data. The appropriate values and their uncertainties (if given in the original file) are accessible with `ppe.x`, `ppe.y`, `ppe.dx`, and `ppe.dy`.

Finally we can perform peak extraction and saved the results.

```
ppe.extract()
ppe.write("foobar.srmise")
```

To save a more human-readable version of the results,

```
ppe.writepwa("foobar.pwa")
```

records a file with a summary of extraction variables and the extracted peaks. In addition, most high-level objects in SrMise produce a human-readable summary when used with the Python print keyword. A no-frills plot of the PDF and extracted peaks is produced using `ppe.plot()`.

SrMise optionally prints a lot of information to stdout during extraction, which can be adjusted. The default amount of output is equivalent to

```
import diffpy.srmise.mise.miselog as ml
ml.setlevel("info")
```

Supported levels are "debug", "info", "warning", "error", "critical". In addition, one may set up a separate log file with its own level. For example,

```
ml.addfilelog("foobar.log")
ml.setfilelevel("debug").
```

6.2 Multimodeling

In SrMise multimodeling is handled with the `diffpy.srmise.mise.multimodelselection` module. Best practices when multimodeling peak extraction from the PDF are still being established. This module is functional but in a rough state, and may change drastically in future versions of SrMise. Consequently, this section is more of a sketch of multimodeling than a complete guide. The user is encouraged to look over the example of multimodeling included with the distribution, as well as the technical details in Granlund et al. (2014).

The first step is performing peak extraction many times to create these models. This is much like the previous section, except that the assumed uncertainty δg is varied over a range of physically plausible values. Even if the true uncertainty in $G(r)$ is known this is useful for generating a population of models, because any single model produced at a given assumed uncertainty is not guaranteed to be optimal.

To prepare for peak extraction, one creates a `PDFPeakExtraction` instance (which I will call `ppe`) and sets parameter variables as in the previous section. Then associate it with the multimodel tool as follows

```
from diffpy.srmise.mise import MultimodelSelection
ms = MultimodelSelection()
ms.setppe(ppe)
```

Next, define a range of uncertainties to assume while performing extraction. These can be any sequence of valid inputs to the `dg` extraction variable. However, when analyzing results at a later stage it is assumed that the uncertainties for any single trial can be transformed to that of another trial simply by scaling. For example, either of the following is acceptable:

```
dgs = numpy.linspace(1,5,9)
dgs = [s*ppe.dy for s in numpy.linspace(1,5,9)].
```

The former sets up 9 trials, with all points in the first trial having uncertainty one, all in the second 1.5, etc. The latter also sets up 9 trials, the first with whatever uncertainty the PDF reported (which may differ for each point), the second scaling those by 1.5, and so on. Run the trials and save the results to a file.

```
ms.run(dgs)
ms.save("multifoobar.dat")
```

Before getting to the heart of analysis, calculate and save the AIC-values. This is evaluated for every model at all the uncertainties provided, usually the same ones that were assumed in the first place. To ensure the AIC are comparable, all models must be evaluated over identical data. In the present version this is done by resampling the data once at a specified sampling rate dr, usually the Nyquist rate.

```
rate = numpy.pi/ppe.qmax
ms.makeaics(dgs, rate, filename="multifoobarAIC.dat")
```

These files may be loaded using

```
ms.load("multifoobar.dat")
ms.loadaics("multifoobarAIC.dat").
```

Following this, we classify the models using the same data so that those which are very nearly identical are in the same class. This is necessary to generate plausible "Akaike probabilities", which (subject to considerable technical details, see Granlund et al. (2014)) are the likelihood that a given model describes the data better than any of the other models under consideration.

```
(r, g, dr, dg) = ms.ppe.resampledata(rate)
ms.classify(r, tolerance=0.1)
```

This function calculates a different between two peaks subject to a particular tolerance. For reference, a difference of 0 means identical, while a difference of 1 is that of a peak to a line that is identically 0. The method used to classify models is likely to change in future version of SrMise, but in testing thus far the final results are robust to even fairly large changes in tolerance, say from 0.05 to 0.2.

A summary of other useful `MultimodelSelection` methods and members follows. See the docstrings for details. In this table `model` is an integer, the index to the model generated by the *i*th trial, and `dG` is an element of `ms.dgs`, the uncertainty used when comparing models.

Attribute	**Summary**
aics[dG][model]	Maps given dG and model to calculated AIC.
aicprobs[dG][model]	Maps given dG and model to calculated Akaike probability.
bestmodels()	Returns all models which have greatest Akaike probability for at least one dG.
getmodel(dG)	Returns best model at given dG. Can get different models with corder and morder keywords.
plot3dclassprobs()	Create 3D figure of Akaike probability vs. class vs. dG. Numerous keywords for customization. Returns dictionary referencing the figure, axes, etc.
setcurrent(model)	Set the PDFPeakExtraction instance to result of given model.

6.3 Plotting

SrMise has limited plotting abilities, but the `diffpy.srmise.applications.plot` module provides some additional tools for displaying and comparing the results of peak extraction. In fact, the command-line *srmiseplot* utility is an alias which exposes the basic functionality of this module. For scripting, basic usage is

```
from diffpy.srmise.applications.plot import makeplot
```

`makeplot(ppe)` or
`makeplot(ms)`.
The latter will plot the current model along with an inset summarizing the positions of all extracted peaks for all trials. In addition, you can pass a sequence of numbers as an additional parameter to compare peaks located at those positions to the positions of peaks in the current model.

7 Extensibility

SrMise is structured to support some extensibility, but has been tested and used almost exclusively with the pair distribution function. All extensible features should be considered experimental.

Custom peak and baseline functions may be created by inheritance from the diffpy.srmise.mise.peaks.base.PeakFunction and diffpy.srmise.mise.baselines.base.BaselineFunction classes, respectively. See the doc-strings of those classes for more information.

The core functionality of SrMise is found in the diffpy.srmise.mise.peakextraction module. Peak extraction from functions other than the PDF can be built from this (cf. the pdfpeakextraction module). At present only AIC and AIC$_c$ are supported for model evaluation. Applications with statistical features that do match the assumptions of at least one of these should not use SrMise. See Granlund et al. (2014) for discussion of using the Akaike Information Criterion for peak extraction on pair distribution functions. A comprehensive overview of the AIC is found in Burnham & Anderson (2002).

8 A Note About Interpretation

As with all peak extraction utilities, and model fitting in general, “correctness” is not guaranteed by a fit that looks good, or even one that meets many statistical criteria for quality. SrMise is a tool to expand the reach and efficiency of peak extraction from pair distribution functions in the intermediate regime of overlapping peaks in tandem with Akaike Information Criterion-based multimodeling. It is not a replacement for expert subject knowledge. The user is exhorted to exercise his or her best judgment. Details about best practices and theoretical limitations of this approach are discussed in Granlund et al. (2014)

9 Known Issues

- All peaks are assumed to be positive, i.e. above the baseline. Extraction of negative peaks is not supported, nor is support planned.
- Automatic determination of $Q_{\text{max}}$is experimental and known to fail on some data. Perform a sanity check whenever using this function.
- In the present version the PDF saved within a SrMise file may differ from the original values found in a non-SrMise source in the least significant digits. Therefore, peaks extracted from the original source vs. a SrMise file previously created from that source may differ in unpredictable ways. One work-around is to always load directly from the original source, ignoring the PDF saved in the SrMise file. Since saving a SrMise file loaded from a different one (or itself) preserves the PDF, a better option may be to create such a file from the original source once and then never use the original source file again.
- Resampling introduces small numerical differences which prevents recovering exactly the original data when resampled back at the original rate. This may change peak extraction results in unpredictable ways. Wherever possible preserve the original data and always perform resampling with respect to it.
- Fitting data with narrow features, relative to the minimum width implied by Q_{max}, is unphysical and causes termination ripples to be calculated inaccurately. Extracted peaks in this case will probably be unresponsive to changes in their parameters. In less extreme cases accuracy will still be reduced. If necessary, increase the "supersample" and "extension" variables of a TerminationRipples instance.
- Akaike probabilities are calculated assuming the AIC values are comparable. In cases where the data are differently sampled (normally because pruning could not remove enough parameters to return to Nyquist sampling) the probabilities are, strictly speaking, invalid. If the differences in sampling are minor, however, this can be tolerated to some extent, particularly where the models evaluated on the differently-sampled data would have very little weight anyway.
- Model evaluators besides AIC and AIC_c are not functional at present, as their use is occasionally assumed elsewhere in the present version of SrMise. Peak extraction from pair distribution functions should not use AIC_c for fundamental reasons.

References

Burnham, K. P. & Anderson, D. R. (2002), *Model Selection and Multimodel Inference*, Springer, New York, NY.

Granlund, L., Billinge, S. J. L. & Duxbury, P. M. (2014), ‘Automated Peak Extraction from Atomic Pair Distribution Functions’, *Acta Crystallographica Section A* (Submitted).

Guinier, A., Fournet, G., Walker, C. B. & Yudowitch, K. L. (1955), *Small-angle Scattering from X-rays*, John Wiley & Sons, Inc., New York.