SrRietveld user guide

3.0 Alpha release
25 February, 2009

Wenduo Zhou, Peng Tian, Christopher L. Farrow, Pavol Juh\'as and Simo

Acknowledgements

This program is part of the DiffPy and DANSE open-source projects and is available subject to the conditions and terms laid out below.

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:

1. Wenduo Zhou, Peng Tian, Christopher L. Farrow, P. Juh\'as and S. J. L. Billinge, http://stacks.iop.org/0953-8984/19/335219 To be published.

For more information please visit the project web-page: http://www.diffpy.org/ or email Prof. Simon Billinge at sb2896@columbia.edu

This version of SrRietveld makes use of the FullProf and GSAS Rietveld refinement engine. If you use this program, as well as citing the paper above, you should acknowledge use of FullProf and/or GSAS by citing the paper describing the FullProf and/or GsAS:

1. J. Rodriguez-Carvajal, Recent advances in magnetic structure determination by neutron powder diffraction, Physica B, 192, 55-69 (1993).
2. Larson A C and von Dreele R B 2000 General structure analysis system(GSAS) Los Alamos National Laboratory Report No. LAUR 87-748 unpublished

Table of Contents

1 Introduction

Rietveld refinement is a process of determining accurate atomic structural information about crystalline materials from neutron or x-ray powder diffraction. There are a number of Rietveld computer programs available, both freeware and commercial. What they have in common is that they require extensive interactions with the scientist operating them because of the difficulty in converging fits of the models to the data. They have poor convergence and there is a sharp learning curve for using them. The codes were mostly written beginning in the 1980's and 1990's and are monolithic Fortran programs and it is very difficult to automate and extend them.

The goal of the SrRietveld project is to mitigate these problems and make a Rietveld program that is highly automated, easy to use, and can handle the high data-rates at POWGEN and other SNS diffractometers. There is insufficient time and insufficient resources to write completely new, modular, Rietveld codes within the DANSE project, so the approach in the SrRietveld project was to wrap existing Rietveld legacy codes in Python wrappers, allowing automation and other scripts to be written in Python to control the execution of the existing codes. In the SrRietveld 3.0a release we have implemented the two most popular Rietveld programs, Fullprof and GSAS, as engines running in the Python environment. The design includes a Rietveld API (application programming interface) which allows users to write scripts in Python that can execute different refinement engines. In the current release this means that the same data can be refined with either Fullprof or GSAS, or both. In the future it will allow us to change out the Rietveld engine with any other Rietveld legacy code, or newly written code, and users' scripts will still be operable.

1.1 SrRietveld

SrRietveld is an open-source crystal structure refinement user-interface (UI) for reciprocal-space fitting of powder data, or Rietveld analysis.

SrRietveld is being developed as part of the DANSE software project that is building data analysis and modeling software to make optimal use of neutron diffraction data coming from the Spallation Neutron Source.

SrRietveld is reliable. It will provide the expected features of Rietveld analysis software, such as that provided by FullProf or GSAS, as well as features that simplify typical and advanced usage.

SrRietveld is easy to use. It will provide users with tools to set up Rietveld refinement in relatively trivial time and launch a series of Rietveld refinement automatically or semi-automatically until convergence, allowing complex refinement tasks to be automated and simplified. Meanwhile, SrRietveld provides users tools to analyze the refinement result visually.

In addition to providing users with ease-of-use, we wish to enable new science. In order to satisfy ever-changing user community, extensible, non-proprietary and supported software, such as SrRietveld, will enable innovation and empower users.

SrRietveld is fully extensible. It will work by allowing the user to write scripts in a Python-based scripting language that control and automate the refinement process It is also then straightforward for a developer to build simple gui's on top of the scripts. A library of helper functions is available to simplify common tasks.

The major features supported by SrRietveld currently listed below:

1. supporting both FullProf and GSAS as refinement engines. Exchanging underlying refinement engine is seamless to users
2. supporting automatic/semi-automatic launched Le-Bail and Rietveld refinement on single measurement toward final convergence
    1. applied to neutron time-of-flight (multiple banks), neutron constant wavelength, X-ray diffraction data;
    2. allowing user to trace each parameter's value at each refinement step for further analysis;
    3. allowing user to plot patterns of each refinement step after refinement is finished for further analysis.
3. supporting sequential fitting on series of measurements from in-situ diffraction experiment,

    applied to neutron time-of-flight (multiple banks), neutron constant wavelength, X-ray diffraction data;
    1. providing user with fast access to refined parameters and refinement result (residuals);
    2. allowing user to access the refinement detail of each single measurement for further analysis;
4. supporting diffraction pattern calculation
    1. applied to neutron time-of-flight (multiple banks), neutron constant wavelength, X-ray diffraction data;

1.1.1 Design principles

SrRietveld has been designed to provide users with an easy-to-use yet powerful interface for fitting structure models to powder diffraction data. It makes use of an object oriented, component based architecture, which makes it highly extensible and maintainable.

SrRietveld is written in the Python programming language. Python features a relaxed and friendly syntax, supports "write once, run anywhere" portability, and has extensive libraries and modules for virtually every task. Software codes written in a variety of programming languages can be bound into Python, which allows them to be used together. Python is becoming a popular choice in the scientific computation community.

SrRietveld is built upon RietveldAPI. It is a library of python scripts to set up, manage and analyze Rietveld refinements.

General users can use pre-built scripts to quickly and easily carry out refinements that are offered as part of the SrRietveld application. In the future, these will be controlled through a graphical user interface, though currently they are executed from the command line.

Power users can build their customized Rietveld refinements by writing high-level scripts using SrRietveld and rietveldAPI library modules.

In 3.0 Alpha release, SrRietveld provides the users with a series of command line scripts to set up, launch and analyze Rietveld refinements.

1.1.2 Capabilities

SrRietveld will allow users to perform powerful refinement tasks with little effort. Python has a friendly and powerful scripting interface that enables SrRietveld to interact with other Python modules such as matplotlib for plotting and pymol for structure visualization. The scripting interface to SrRietveld allows one to easily string together a series of related Rietveld refinements, analyze the refinement results and optionally feed the results into other Python-enabled programs. SrRietveld will soon be enabled to run distributed on a computer cluster in order to deal with the fast data throughput of POWGEN3. Once-difficult refinement strategies, such as parametric optimization and global optimization are also planned for a future release. Users can also look forward to automatic Rietveld refinement that requires a minimal amount of prior knowledge about a crystal structure.

SrRietveld can manage multiple fits at once. Each fit can have multiple experimental data sets and structure models. Fits in a sequence can call upon other fits for their starting parameters, and configured fits can be queued to run while the user is away. All the initial, final, and intermediate data are stored in a platform independent project file that can be loaded on any computer. All management tasks, such as fit creation, configuration, modification, and visualization, can be done through the graphical interface.

SrRietveld uses the matplotlib (http://matplotlib.sourceforge.net) Python package for 2D plotting of data and results. Matplotlib has a friendly interface so the user can quickly and easily view the results of a fitting. SrRietveld lets users plot data from a series of fits and plot it against selected meta-data (temperature, doping, etc.).

1.2 Fullprof and GSAS

The FullProf and GSAS program has been mainly developed for Rietveld analysis (structure profile refinement) of neutron (constant wavelength, time of flight, nuclear and magnetic scattering) or X-ray powder diffraction data collected at constant or variable step in scattering angle 2theta. The programs can be also used as a Profile Matching (or pattern decomposition using Le Bail method) tool, without the knowledge of the structure. Single crystal refinement can also be performed alone or in combination with powder data. Time of flight (TOF) neutron data analysis is also available. Energy dispersive X-ray data can also be treated but only for profile matching.

The Fullprof engine and GSAS engine, called PyFullProf and PyGSAS respectively, can be used either directly from the Python command line, or as part of larger and more complex software applications.

1.3 Availability

SrRietveld, FullProf and GSAS are open source. SrRietveld is distributed under a BSD license. They run on Linux, and all major Unix systems. The source code is freely available. For more information please contact Professor Simon Billinge (sb2896@columbia.edu) or consult the web-page http://www.diffpy.org. News of updates and releases will be posted at this website and on the diffpy-users group at http://groups.google.com/group/diffpy-users.

1.4 Installation

SrRietveld will use several Diffpy libraries, including:

1. diffpy.srrietveld
2. diffpy.pyfullprof
3. diffpy.pygsas
4. diffpy.rietveldapi
5. diffpy.refinementapi
6. diffpy.Structure

Unix/Linux/MAC

In 3.0 Alpha release, SrRietveld will be deployed to the analysis computer of SNS' POWGEN.

The enthusiastic testers can download the diffpy libraries from DANSE's repository at: svn@danse.us/diffraction

Required third party packages:

`python` http://www.python.org Python interpreter, minimum version 2.3

`python-dev`
http://www.python.org development files for Python

`numpy` http://numpy.scipy.org Numerical Python, minimum version 0.9.8

`python-matplotlib`
http://matplotlib.sourceforge.net plotting library for Python, minimum version 0.90

Windows

1.5 Community

There are several Google groups for support and development of SrRietveld and other DiffPy software. Visit the links below for message archives or instructions on subscription and posting.

srrietveld-dev http://groups.google.com/group/srrietveld-dev
Please share your comments, desires and suggestions with the developers by posting to the Diffpy developers' forum, or by emailing Simon Billinge (sb2896@columbia.edu). Since we are in the alpha-testing phase there are no user community pages. In future the discussion will be merged to diffpy-users and diffpy-dev.

diffpy-users http://groups.google.com/group/diffpy-users
Help on usage of SrRietveld, Fullprof and other DiffPy packages. This group should become a knowledge base of SrRietveld user tips, tricks and troubleshooting. Feel free to ask your question here.

diffpy-dev http://groups.google.com/group/diffpy-dev
For discussions about development and changes of SrRietveld, Fullprof and DiffPy library in general.

2 Quick start (Tutorial)

In this chapter the SrRietveld layout will be briefly described, followed by a couple of simple examples as tutorial.

Users familiar with the basics can proceed to ⟨undefined⟩ [Examples and tutorials], page ⟨undefined⟩, or use ⟨undefined⟩ [SrRietveld reference sheets], page ⟨undefined⟩. All the files used in examples are available in the tutorial subdirectory relative to this document.

2.1 SrRietveld layout

SrRietveld is designed to have both graphic user interface and command line scripts for users to choose.

2.1.1 SrRietveld Graphic User Interface

SrRietveld GUI is still in development. It will be ready in future release.

2.1.2 SrRietveld command line scripts

SrRietveld can also be invoked in terminal console commands. Power users can use these commands with flexibility.

The SrRietveld input files are in XML format, whose syntax is illustrated in this user manual. For convenience, SrRietveld command line script can generate some template XML files for users.

There are three types of input files:

1. Instrument file: defining the parameters related to the instrument and provided by instrument scientists. This file is necessary.
2. Crystal structure model file: defining the crystal structure model to be refined against. This file is also necessary.
3. Refinement strategy file: defining the sequence to refine parameters. This file is optional. If this file is not given, SrRietveld will perform the refinement by using a default refinement strategy.

2.2 Tutorial

The purpose of this example is to familiarize the novice user with the SrRietveld basics.

The goal is to refine a couple of hundreds of measurements on NaCl by IDD11 in APS, Argonne National Lab.

The data files can be copied from SNS analysis machine at: /SNS/users/wdzhou/SrRietveld-Examples/NaCl.

Inside, you will find

1. i_rapdf_idd11_cal.xml: the instrument XML file calibrated by Ni data;
2. s_nacl.xml: the structure of NaCl obtained from literature;
3. backgroundtemp.txt: a background file;
4. NaCl_Inof_All.txt: 2 column file listing the data file name and their corresponding temperature;
5. Directory Data/ containing 321 data sets measured at 321 different temperatures.

2.2.1 Pattern Calculation

The first step is to calculate the pattern according to the proposed structure model, to make sure that the model is not obviously incorrect.

The proposed structure of NaCl is the structure at room temperature. Therefore, by referring to NaCl_Info_All.txt, we choose Data/NaCl_ramp01_10f_200ms_1059.chi as the pattern to compare.

Command:

```
> srpc -i i_rapdf_idd11_cal.xml -s s_nacl.xml -d Data/NaCl_ramp01_10f_200ms_1059.chi
 --compare=Data/NaCl_ramp01_10f_200ms_1059.chi -b backgroundtemp.dat --
scale=1.0E-6
```

The calculated pattern and the real experimental pattern will be plotted against each other. If the magnitude of them are not close enough, the scale factor can be tuned by inputting an appropriate value with option –scale.

The –scale factor should not be large, or the calculated pattern may be overflow.

2.2.2 Refinement on a single measurement

It is better to do Rietveld refinement on a single diffraction measurement, before starting the sequential refinement.

Because the starting structure model is at room temperature, we choose NaCl_ramp01_10f_200ms_1059.chi, which was measured at room temperature.

Command:

```
> srr -i i_rapdf_idd11_cal.xml -s s_nacl.xml -d Data/NaCl_ramp01_10f_200ms_1059.chi
 -b backgrountemp.dat -v refine_1059.proj
```

The refinement will be launched with the convergence strategy defined in SrRietveld. And the refinement result will be saved in file refine_1059.proj The refinement engine in default is Fullprof.

With additional option,

```
--engine=gsas or
-e gsas
```

2.2.3 Pattern plotting

User can view diffraction pattern of the final result (last round) as

```
> srpp -o refine_1059.proj
```

or he/she can view the diffraction pattern of at the third step

```
> srpp -o refine_1059.proj -r 3
```

2.2.4 View refinement result (single measurement)

User can look up a parameter, such as related to name "chi2" or "biso"

```
> srvr -p refine_1059.proj --parameter=chi2
> srvr -p refine_1059.proj --parameter=bios
> srvr -p refine_1059.proj --parameter=bios --atom=Na
```

These will show the parameters' values at the last step with uncertainties.

If the user is interested to see how one specific parameters changes along refinement, he/she can add an option

```
--step=all
```

2.2.5 Save configuration from saved project file

After a refinement is completed and saved to a SrRietveld project file, user can export the refined crystal structure to a SrRietveld structure file in XML format, which can be used as an input for future refinement.

```
srvr -p refine_1059.proj -s nacl259.xml
```

User can also export the refinement strategy file, such that she/he can modify to the refinement strategy, including sequence of refinement guides, to refine the structure model again for better convergence.

```
srvr -p refine_1059.proj -r refinestrategy_userdefine.xml
```

The user can use any text editor or specific XML editor to modify the XML containing refinement strategy.

For example, the user can do further refinement based on the previous refinement result:

```
> srr -i i_rapdf_idd11_cal.xml -s s_nacl259.xml -d Data/NaCl_ramp01_10f_200ms_1059.chi
 -b backgrountemp.dat -v refine_1059.proj
```

or if with other refinement strategy

```
> srr -i i_rapdf_idd11_cal.xml -s s_nacl259.xml -d Data/NaCl_ramp01_10f_200ms_1059.chi
 -b backgrountemp.dat -v refine_1059b.proj -r refinestrategy_userdefine.xml
```

2.2.6 Sequential fitting on a series of measurements

SrRietveld supports sequential refinement on a set of measurements on same material but in different environment, for example, temperature or pressure.

In this tutorial, it is a temperature series.

Before starting the refinement, the user should have

1. calibrated instrument file: i_rapdf_idd11_cal.xml
2. proposed structure model file: s_nacl.xml, or s_nacl_X.xml that is exported from the result of previous refinement, for example refine_1059.proj or refine_1059b.proj
3. data-environment parameter value list file: NaCl_Info_All.txt

```
> srsr -i i_radpf_idd11_cal.xml -s s_nacl_02.xml -l Data -f NaCl_Info_All.txt -
b backgroundtemp.dat
 -n T --refinedirection=down --start=295.0 -v seqrefine_nacl.proj
```

By this command, SrRietveld will refine all the measurements starting from 295.0 K down to the lowest temperature measured int the experiment, while all the data are stored in directory Data. The refinement engine in default is Fullprof.

With additional option,

```
--engine=gsas or
-e gsas
```

SrRietveld will use GSAS as refinement engine.

SrRietveld's sequential refinement will save the refinement results in several files and directories.

In the example just shown, SrRietveld will

1. save all the refinement history and parameter to a single file, seqrefine_nacl.proj. It will be a large file, but with all information

2. generate a directory named, seqrefine_nacl

3. generate a light-weight XML file in directory seqrefine_nacl. In this XML file, only the information of chi2, Rwp, and crystal structures are saved.

4. the project files of refinement on every single measurement. These files will be stored in directory seqrefine_nacl.

2.2.7 View refinement results of sequential refinements

The user will use a different script: srrietveld-viewsequentialrefineresult.py (srvsr) to look up the parameters from the result of sequential fitting.

```
> srvsr -o seqrefine_nacl --parameter=chi2 -p : chi^2 in all the refinements
> srvsr -o seqrefine_nacl --parameter=biso -p : Biso of all the atoms in all the refin
> srvsr -o seqrefine_nacl --parameter=biso --atom=Cl -p : Biso of atom Cl in all the r
```

The option -p is used to plot the result.

3 Executable commands reference manual

3.1 srrietveld-patterncalculation.py (srpc)

srrietveld-patterncalculation.py (also named srpc) lets the users to simulate a pattern with a given structure model and a given instrument setup.

3.2 srrietveld-saveconfig.py (srsv)

srrietveld-saveconfig.py (also named srsv) is able to save various configurations from SrRietveld saved project files, FullProf pcr files, or GSAS iparm files to standard SrRietveld XML configuration files to set up new refinement.

The format of the XML files will be introduced in next chapter.

3.3 srrietveld-refine.py (srr)

srrietveld-refine.py (aka. srr) will do a multiple-step refinement, including Le-Bail and Rietveld, if selected, on a crystal structure model given by user toward convergence. The refinement is configured according to the instrument parameters which is input by user. The refinement can be launched either automatically or semi-automatically according to user's specified refinement strategy. The final result and all the results in internal steps are saved to a project file designated by user.

3.4 srrietveld-viewrefinementresult.py (srvr)

srrietveld-viewrefinementresult.py (also named srvr) offers users with the capability to read parameters' values from SrRietveld project file saved by srrietveld-refine.py (srr).

3.5 srrietveld-sequentialrefne.py (srsr)

srrietveld-sequentialrefine.py (also named srsr) offers users with the capability to refine multiple related measurements with automatic launched refinement till convergence for every measurement.

The refinement result will be saved to files in multiple format for further review.

3.6 srrietveld-viewsequentialrefineresult.py (srvsr)

srrietveld-viewsequentialrefineresult.py (also named srvsr) offers users with the capability to read parameters' values from the projects files saved by srrietveld-sequentialrefine.py (srsr).

3.7 srrietveld-plotpattern.py

srrietveld-plotpattern.py (aka. srpp) is able to plot the diffraction pattern, including the calculated pattern, the observed pattern, the difference between of these two patterns, and the reflections, at any refinement step during the refinements, from a project file saved by srrietveld-refine.py.

4 Work flow of refining data measured by specific instruments

The work flow to use SrRietveld to refine the data measured at specific diffractometers may not be the same. In this chapter, how to use SrRietveld with several specific instruments will be taken care.

4.1 POWGEN

POWGEN has not been online yet. Please visit back this section in future.

4.2 RaPDF

In order to refine the data made from RAPDF, some prerequisites should be met:

1. A directory containing: Sample CHI files with file names given in routine, which SrRietveld understands. By the names, SrRietveld can figure out the environment parameters values (such as temperature); The proposed structure models for the sample;
2. A CHI file for standard (Si, Ni, or etc) calibration data with file name given in routine, which SrRietveld understands. By the name, SrRietveld can figure out what material of this standard is (Ni, Si, and etc) and at what temperature, this standard is measured.
3. Essential parameters of RaPDF instrument, including energy and peak profile type (optional, default is Thomas-Cox-Hastings Pseudo Voigt).

The usual workflow to process RaPDF data is as

1. Reduce raw data to CHI files by using FIT2d; Giving the proper file names to these files; Putting the data sets measured from the same samples to same directory.
2. Ask SrRietveld to generate a RaPDF instrument file; Correct the energy (wavelength) in instrument file. Alternatively, tell SrRietveld the energy;
3. Use WinPlotr or other application to view calibration data to figure out whether excluded regions should be added and how large the refinement range is;
4. Optionally, use WinPlotr to view calibration data to select the background points manually and saved to a file. Use WinPlotr to view all the diffraction data to figure out whether excluded regions should be added and how large the refinement range is;
5. Optionally, use Winplotr to select the backgrounds of all the data files;
6. Create structure XML file from a template obtained from SrRietveld if user does not have CIF file;
7. Get a calibrated instrument file from instrument scientist, or use SrRietveld to do the calibration by the CHI file of calibration data
8. Determine the excluded regions and refinement range;
9. Record the serial number given by SrRietveld about the calibration result;
10. Run SrRietveld: by giving a) directory of sample files b) calibration serial number obtained in step 8; c) background files if he does not want automatic selection d) range of data files to refine (all files or a subset) e) direction of refinement (for example, from high-T to low-T or the opposite direction) f) excluded regions g) refinement range (in 2-theta unit)

4.3 NPDF

The implementation of guidance of NPDF will be completed soon.

5 SrRietveld reference sheets

5.1 SrRietveld XML File Format

SrRietveld uses XML format files as the standard input, due to Rietveld refinement's complicated setup.

Most of XML input files are hidden from general users. Knowing how to edit these XML file can provide users more power to control the refinement performed in SrRietveld

5.1.1 Model XML File

The XML file to describe a crystal model is straightforward. We even expect the general user to know how to edit it.

The model XML file have the format as this example.

<?xml version='1.0' encoding='UTF-8'?> <MODEL>

<PHASE Phase Attributes>

<ATOM Atom Attributes>

...

<ATOM Atom Attributes>

</PHASE>

...

<PHASE Phase Attributes>

<ATOM Atom Attributes>

...

<ATOM Atom Attributes>

</PHASE>

</MODEL>

XML Tree

Phase Attribute

Type, Name, SpaceGroup, a, b, c,

Atom attribute

Element, Tag, x, y, z, occ, biso (or b11, b22, b33, b12, b13, b23)

Example

<?xml version='1.0' encoding='UTF-8'?>

<MODEL>

__<PHASE Type='User' Name='FeSi' SpaceGroup='P 21 3' a='4.474656' b='4.474656' c='4.474656' alpha='90.0' beta='90.0' gamma='90.0'>

____<ATOM Element='Si' Tag='Si' x='-0.15760' y='-0.15760' z='-0.15760' biso='-0.16290' occ='4.00000'/>

____<ATOM Element='Fe' Tag='Fe' x='0.13652' y='0.13652' z='0.13652' biso='-0.12682' occ='4.00000'/>

__</PHASE>

</MODEL>

5.2 List of SrRietveld parameters

The following is the list of SrRietveld variables, including their default values in parentheses, and a brief description and a note, where appropriate.

(To be completed soon)

Index