ODAT-SE LEED Module Documentation

Release 1.0.0

ISSP, University of Tokyo

Apr 11, 2025

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 What is ODAT-SE ?

ODAT-SE is a framework for applying a search algorithm to a direct problem solver to find the optimal solution. As the standard direct problem solver, the experimental data analysis software for two-dimensional material structure analysis is prepared. The direct problem solver gives the deviation between the experimental data and the calculated data obtained under the given parameters such as atomic positions as a loss function used in the inverse problem. The optimal parameters are estimated by minimizing the loss function using a search algorithm. For further use, the original direct problem solver or the search algorithm can be defined by users. In the current version, for solving a direct problem, ODAT-SE offers the wrapper of the solver for the total-reflection high-energy positron diffraction (TRHEPD), the surface X-ray diffraction (SXRD), and the low-energy electron diffraction (LEED). As algorithms, it offers the Nelder-Mead method, the grid search method, the Bayesian optimization method, the replica exchange Monte Carlo method, and the population annealing Monte Carlo method.

1.2 What is odatse-LEED ?

`SATLEED` is a software package developed by M.A. Van Hove for the analyses of LEED, which calculates the I-V curve from the atomic positions and other parameters, and evaluate the deviations from the I-V curve obtained from the experiments. odatse-LEED is an adaptor library to use SATLEED as a direct problem solver of ODAT-SE. It was originally developed as a component of 2DMAT v2.x, and has been restructured as a separate module to be used with ODAT-SE and SATLEED. For more information on `SATLEED`, see [SATLEED].

1.3 License

This software was developed with the support of "Project for advancement of software usability in materials science" of The Institute for Solid State Physics, The University of Tokyo. We hope that you cite the following reference when you publish the results using ODAT-SE:

"Data-analysis software framework 2DMAT and its application to experimental measurements for two-dimensional material structures", Y. Motoyama, K. Yoshimi, I. Mochizuki, H. Iwamoto, H. Ichinose, and T. Hoshi, Computer Physics Communications 280, 108465 (2022).

Bibtex:

```
@article{MOTOYAMA2022108465,
  title = {Data-analysis software framework 2DMAT and its application to experimental␣
```

(continues on next page)

(continued from previous page)

```
↪measurements for two-dimensional material structures},
  journal = {Computer Physics Communications},
  volume = {280},
  pages = {108465},
  year = {2022},
  issn = {0010-4655},
  doi = {https://doi.org/10.1016/j.cpc.2022.108465},
  url = {https://www.sciencedirect.com/science/article/pii/S0010465522001849},
  author = {Yuichi Motoyama and Kazuyoshi Yoshimi and Izumi Mochizuki and Harumichi␣
↪Iwamoto and Hayato Ichinose and Takeo Hoshi}
}
```

1.4 Version Information

odatse-LEED

- v1.0.0: 2025-04-11
- v1.0-alpha: 2024-11-25

ODAT-SE

- v3.0.0: 2024-11-25

2DMAT

- v2.1.0: 2022-04-08
- v2.0.0: 2022-01-17
- v1.0.1: 2021-04-15
- v1.0.0: 2021-03-12
- v0.1.0: 2021-02-08

1.5 Main developers

ODAT-SE, odatse-LEED, and 2DMAT have been developed by following members.

- ODAT-SE v3.0.0 -, odatse-LEED v1.0-alpha -
  - Y. Motoyama (The Institute for Solid State Physics, The University of Tokyo)
  - K. Yoshimi (The Institute for Solid State Physics, The University of Tokyo)
  - T. Aoyama (The Institute for Solid State Physics, The University of Tokyo)
  - T. Hoshi (National Institute for Fusion Science)
- 2DMAT v2.0.0 -
  - Y. Motoyama (The Institute for Solid State Physics, The University of Tokyo)
  - K. Yoshimi (The Institute for Solid State Physics, The University of Tokyo)
  - H. Iwamoto (Department of Applied Mathematics and Physics, Tottori University)
  - T. Hoshi (Department of Applied Mathematics and Physics, Tottori University)
- 2DMAT v0.1.0 - v1.0.1

- Y. Motoyama (The Institute for Solid State Physics, The University of Tokyo)
- K. Yoshimi (The Institute for Solid State Physics, The University of Tokyo)
- T. Hoshi (Department of Applied Mathematics and Physics, Tottori University)

CHAPTER

TWO

INSTALLATION OF ODATSE-LEED

2.1 Prerequisites

- Python3 (>=3.9)
- The following Python packages are required.
  - numpy >= 1.14
  - pydantic >= 2.0
  - fortranformat
  - ODAT-SE >= 3.0
- SATLEED
  - A Fortran compiler is required for compilation.

2.2 How to download and install

1. Install ODAT-SE
   - From source files:

     Download source files of ODAT-SE from the repository as follows:

     ```
     $ git clone https://github.com/issp-center-dev/ODAT-SE.git
     ```

     Install ODAT-SE using `pip` command:

     ```
     $ cd ODAT-SE
     $ python3 -m pip install .
     ```

     You may add `--user` option to install ODAT-SE locally (in `$HOME/.local`).

     Note

     For Python versions below 3.7, you will get the following error. Please use Python 3.9 or higher for this package.

     ```
     Directory '.' is not installable. File 'setup.py' not found.
     ```

     If you run the following command instead, optional packages will also be installed at the same time.

```
$ python3 -m pip install .[all]
```

2. Install odatse-LEED

   - From source files:

     The source files of odatse-LEED are available from the GitHub repository. After obtaining the source files, install odatse-LEED using `pip` command as follows:

     ```
     $ cd ../
     $ git clone https://github.com/2DMAT/odatse-LEED.git
     $ cd odatse-LEED
     $ python3 -m pip install .
     ```

     You may add `--user` option to install the package locally (in `$HOME/.local`). Then, the library of odatse-LEED and the command `odatse-LEED` wil be installed.

3. Install SATLEED

   - The source archive is available from the following URL.

     ```
     http://www.icts.hkbu.edu.hk/VanHove_files/leed/leedsatl.zip
     ```

     Unpack the source package, and compile the source.

   - `SATLEED` requires modifying source code parameters according to the details of the system you want to calculate. The source files of odatse-LEED include `leedsatl.patch` for modifying the source code and `setup.sh` script for automatic compilation in the `sample/satleed` directory. You can use these files to install SATLEED.

     ```
     $ cd odatse-LEED/sample/satleed
     $ sh ./setup.sh
     ```

     By running `setup.sh`, the executable files `satl1.exe` and `satl2.exe` will be generated in the current directory.

     You may edit `setup.sh` to use other compilers or to apply different compiler options.

2.3 How to run

In ODAT-SE, the analysis is done by using a predefined optimization algorithm and a direct problem solver. There are two ways to do analyses of LEED:

1. Use odatse-LEED program included in this package to perform analyses. The users prepare an input parameter file in TOML format, and run command with it. The type of the inverse problem algorithms can be chosen by the parameter.
2. Write a program for the analysis with odatse-LEED library and ODAT-SE framework. The type of the inverse problem algorithms can be chosen by importing the appropriate module. A flexible use would be possible, for example, to include data generation within the program.

The types of parameters and the instruction to use the library will be given in the subsequent sections.

2.4 How to uninstall

In order to uninstall odatse-LEED and ODAT-SE modules, type the following commands:

```
$ python3 -m pip uninstall odatse-LEED ODAT-SE
```

CHAPTER

THREE

TUTORIALS

odatse-LEED provides a direct problem solver for ODAT-SE that uses SATLEED developed by M. A. Van Hove for the analyses of the low-energy electron diffraction (LEED) data. SATLEED calculates diffraction data for given atomic positions. Regarding this as a direct problem, it corresponds to an inverse problem to find optimal atomic coordinates that reproduce the diffraction data obtained from the experiment. ODAT-SE provides a framework to solve these inverse problems.

In this tutorial, we will instruct how to use odatse-LEED to find optimal configuration by the grid search (mapper). We use odatse-LEED program included in odatse-LEED with input files in TOML format. Next, we will explain how to write your own main program for analyses.

3.1 Optimization by Grid search

In this section, we will explain how to perform a grid-type search to analyze atomic coordinates from diffraction data. The grid type search is compatible with MPI. It is necessary to prepare the data file MeshData.txt that defines the search grid in advance.

3.1.1 Location of the sample files

The sample files are located in sample/mapper. The following files are stored in the folder:

- base directory

  Directory that contains reference files to proceed with calculations in the main program. The reference files include: exp.d, rfac.d, tleed4.i, and tleed5.i.

- input.toml

  Input file of the main program.

- MeshData.txt

  Data file for the search grid.

- ref_ColorMap.txt

  Reference file of the output to check if the calculation is performed correctly.

- prepare.sh, do.sh

  Script prepared for bulk calculation of this tutorial.

Below, we will describe these files and then show the actual calculation results.

3.1.2 Reference files

`tleed4.i`, `tleed5.i`, and `rfac.d` are the parameter files for satleed. The atomic coordinates to be optimized should be replaced by the keywords such as `opt000` or `opt001` in `tleed5.i`. `exp.d` is the reference experimental data.

The search grid is given by `MeshData.txt`. In this tutorial, the content of `MeshData.txt` is as follows:

```
1 -0.490000 0.777500
2 -0.490000 0.977500
3 -0.490000 1.177500
4 -0.490000 1.377500
5 -0.490000 1.577500
...
```

The first column denotes the sequential id, and the second and subsequent columns denote the parameter values for `opt000` and `opt001` in `tleed5.i`.

Note

A script file `make_meshdata.py` is prepared to generate `MeshData.txt`. Run the script as follows:

```
$ python3 make_meshdata.py > MeshData.txt
```

3.1.3 Input file

This section describes the input file for the main program, `input.toml`. The details of `input.toml` can be found in the input file section of the manual. The following is the content of `input.toml` in the sample file.

```
[base]
dimension = 2
output_dir = "output"

[solver]
name = "leed"
[solver.config]
path_to_first_solver = "../satleed/satl1.exe"
path_to_second_solver = "../satleed/satl2.exe"
[solver.param]
string_list = ["opt000", "opt001"]
[solver.reference]
path_to_base_dir = "./base"
rfactor = "satleed"

[algorithm]
name = "mapper"
label_list = ["z1", "z2"]
[algorithm.param]
mesh_path = "./MeshData.txt"

[runner]
ignore_error = true
```

First, `[base]` section is explained.

- `dimension` is the number of variables to be optimized. In this case, it is `2` since we are optimizing two variables as described in `tleed5.i`.
- `output_dir` is the name of directory for the outputs. If it is omitted, the results are written in the directory in which the program is executed.

`[solver]` section specifies the solver to be used inside the main program and its settings.

- `name` is the name of the solver you want to use. In this tutorial it is `leed`.

The solver can be configured in the subsections `[solver.config]`, `[solver.param]`, and `[solver.reference]`.

`[solver.config]` section specifies options for `satl1.exe` and `satl2.exe` that are called from the main program.

- `path_to_first_solver` is the command name of `satl1.exe`. It is specified as a path to the executable file, or searched from the PATH environment variable.
- `path_to_second_solver` is the command name of `satl2.exe`. It is specified as a path to the executable file, or searched from the PATH environment variable.

`[solver.param]` section specifies the parameters for the solver.

- `string_list` specifies the list of embedded keywords in the input files.

`[solver.reference]` section specifies the location of the experimental data and the range to read.

- `path_to_base_dir` specifies the path to the directory for the reference data.
- `rfactor` specifies the definition of the R-factor. The default value is `rpe` (Pendry's R-factor). In this example, `satleed` is specified that uses the output of SATLEED.

`[algorithm]` section specifies the algorithm to use and its settings.

- `name` is the name of the algorithm you want to use. In this tutorial we will use `mapper` since we will be using grid-search method.
- `label_list` is a list of label names to be attached to the output of `opt000` and `opt001`.

`[algorithm.param]` section specifies the parameters for the search algorithm.

- `mesh_path` is the file name of the search grid.

`[runner]` section specifies the settings on how to execute the solver.

- `ignore_error` specifies how to treat the errors occurred within the execution of the solver. If it is set to true, the result is treated as NaN and the execution of the program is continued.

For details on other parameters that can be specified in the input file, please see the Input File section of the manual.

3.1.4 Calculation execution

Before running the sample program, you need to run `setup.sh` in the `sample/satleed` directory to compile SATLEED and generate `satl1.exe` and `satl2.exe`.

Move to the directory in which sample files are located:

```
$ cd sample/mapper
```

Then, run the main program. The computation time will take only a few minutes on a normal PC.

```
$ mpiexec -np 4 odatse-LEED input.toml | tee log.txt
```

Here, the calculation using MPI parallel with 4 processes will be done. When executed, a folder for each MPI rank will be created in `output`, and the results of the calculations are stored there. The standard output will be shown like as follows.

```
name            : mapper
label_list      : ['z1', 'z2']
param.mesh_path : ./MeshData.txt
Iteration : 1/121
Iteration : 2/121
Iteration : 3/121
Iteration : 4/121
...
```

z1 and z2 are the candidate parameters for each mesh and R-factor is the function value at that point. Finally, the R-factor calculated at all the points on the grid will be written to ColorMap.txt. In this case, the following results will be obtained.

```
-0.490000 0.777500 0.861000
-0.490000 0.977500 1.004700
-0.490000 1.177500 0.909900
-0.490000 1.377500 0.896600
-0.490000 1.577500 1.009500
-0.490000 1.777500 0.779100
-0.490000 1.977500 0.944200
-0.490000 2.177500 0.966500
-0.490000 2.377500 0.867000
-0.490000 2.577500 0.907000
-0.490000 2.777500 0.924100
-0.390000 0.777500 0.801900
-0.390000 0.977500 0.793900
...
```

The first and second columns contain the values of opt000 and opt001, and the third column contains the R-factor.

Note that do.sh is available as a script for batch calculation. In do.sh, the difference between ColorMap.dat and ref_ColorMap.dat is also examined. Here is what it does, without further explanation.

```
#!/bin/sh

sh prepare.sh

time mpiexec -np 4 odatse-LEED input.toml

echo diff output/ColorMap.txt ref_ColorMap.txt
res=0
diff output/ColorMap.txt ref_ColorMap.txt || res=$?
if [ $res -eq 0 ]; then
  echo TEST PASS
  true
else
  echo TEST FAILED: ColorMap.txt and ref_ColorMap.txt differ
  false
fi
```

3.1.5 Visualization of calculation results

By plotting `ColorMap.txt`, we can estimate the region where the value of `R-factor` becomes small. In this case, the following command will create a two-dimensional parameter space diagram `ColorMapFig.png`.

```
$ python3 plot_colormap_2d.py
```

Looking at the generated figure, we can see that it has the minimum value around (0.0, 1.75).

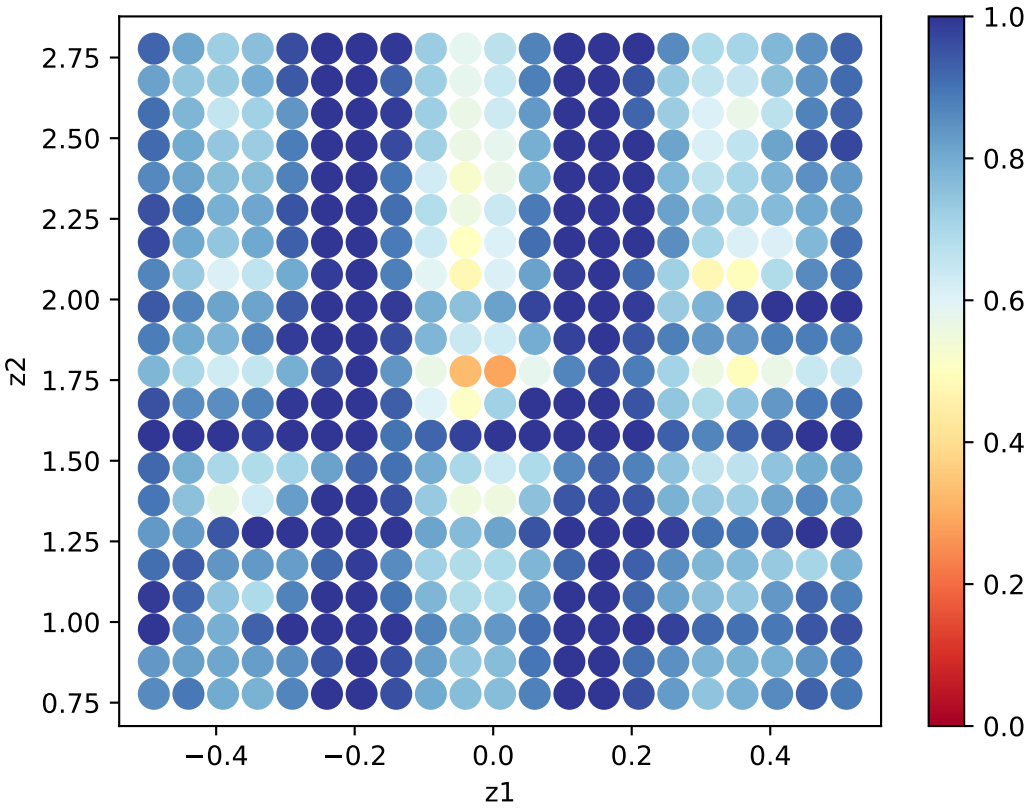

Fig. 3.1: `R-factor` on a two-dimensional parameter space (calculated on 21x21 mesh).

3.2 Analyses by user programs

In this tutorial, we will write a user program using odatse-LEED module and perform analyese. As an example, we adopt Nelder-Mead method for the inverse problem algorithm.

3.2.1 Location of the sample files

The sample files are located in `sample/user_program`. The following files are stored in the folder.

- `simple.py`

  Source file of the main program. This program reads `input.toml` for the parameters.

- `input.toml`

Input file of the main program.

- `base`

  Directory that contains reference files to proceed with calculations in the main program. The reference files include: `exp.d`, `rfac.d`, `tleed4.i`, and `tleed5.i`.

- `ref.txt`

  A reference file for the result of the calculation.

- `prepare.sh` , `do.sh`

  Script prepared for doing all calculation of this tutorial

The following sections describe these files and then show the actual calculation results.

3.2.2 Description of main program

`simple.py` is a simple program for the analyses using odatse-LEED module. The entire source file is shown as follows:

```
import odatse
import odatse.algorithm.min_search
from odatse.extra.LEED import Solver

info = odatse.Info.from_file("input.toml")

solver = Solver(info)
runner = odatse.Runner(solver, info)
alg = odatse.algorithm.min_search.Algorithm(info, runner)
alg.main()
```

At the beginning of the program, the required modules are imported as listed below.

- `odatse` for the main module of ODAT-SE.
- `odatse.algorithm.min_search` for the module of the inverse problem algorithm used in this tutorial.
- `odatse.extra.LEED` for the direct problem solver module.

Next, the instances of the classes are created.

- `odatse.Info` class

  This class is for storing the parameters. It is created by calling a class method `from_file` with a path to TOML file as an argument.

- `odatse.extra.LEED.Solver` class

  This class is for the direct problem solver of the odatse-LEED module. It is created by passing an instance of Info class.

- `odatse.Runner` class

  This class is for connecting the direct problem solver and the inverse problem algorithm. It is created by passing an instance of Solver class and an instance of Info class.

- `odatse.algorithm.min_search.Algorithm` class

  This class is for the inverse problem algorithm. In this tutorial, we use `min_search` module that implements the optimization by Nelder-Mead method. It is created by passing an instance of Runner class.

After creating the instances of Solver, Runner, and Algorithm in this order, we invoke `main()` method of the Algorithm class to start analyses.

In the above program, the input parameters are read from a file in TOML format. It is also possible to take the parameters in the form of dictionary.

3.2.3 Input files

The input file for the main program, `input.toml`, contains parameters for the direct problem solver and the inverse problem algorithm. The contents of the `base` and `solver` sections are the same as those in the previous example.

The parameters for the Nelder-Mead method should be specified in the `algorithm.param` section, while `algorithm.name` for the type of algorithm is ignored.

- `min_list` and `max_list` specifies the search region in the form of the lower and upper bounds of the parameters as lists.
- `initial_list` and `initial_scale_list` specify the initial simplex. For the two-parameter case, when `initial_list = [z1, z2]` and `initial_scale_list = [dz1, dz2]` are specified, a triangle is taken as an initial simplex that has the vertices at `[z1, z2]`, `[z1+dz1, z2]`, and `[z1, z2+dz2]`.

3.2.4 Calculation execution

Before running the sample program, you need to run `setup.sh` in the `sample/satleed` directory to compile SATLEED and generate `satl1.exe` and `satl2.exe`.

Move to the directory in which sample files are located:

```
$ cd sample/user_program
```

Then, run the main program. The computation time will take only a few minutes on a normal PC.

```
$ python3 simple.py | tee log.txt
```

The standard output will be shown like as follows.

```
label_list      : ['z1', 'z2']
param.min_list  : [-0.5, 0.75]
param.max_list  : [0.5, 2.75]
param.initial_list: [-0.2, 1.75]
minimize.initial_scale_list: [0.02, 0.02]
eval: x=[-0.18  1.75], fun=0.9422
eval: x=[-0.14  1.72], fun=0.8607
eval: x=[-0.12  1.745], fun=0.7262
eval: x=[-0.03  1.6975], fun=0.4055
eval: x=[-0.01  1.7225], fun=0.3186
eval: x=[-0.01  1.7225], fun=0.3186
eval: x=[-0.045 1.71875], fun=0.2953
eval: x=[-0.025 1.74375], fun=0.2157
...
```

`z1` and `z2` are the candidate parameters at each step, and `R-factor` is the function value at that point. The final estimated parameters will be written to `output/res.dat`. In the current case, the following result will be obtained:

```
fx = 0.2101
z1 = -0.0199101293087008 4
z2 = 1.7509844067692764
```

You can see that we will get the same values as the correct answer data in `ref.txt`.

Note that `do.sh` is available as a script for batch calculation.

CHAPTER

FOUR

INPUT AND OUTPUT

odatse-LEED module is a Solver which calculates the Rocking curve from atomic positions etc., using SATLEED, and returns the deviation from the experimental Rocking curve as $f(x)$.

In this section, the input parameters, the input data, and the output data are explained. The input parameters are taken from the solver entry of the Info class. The parameters are specified in [solver] section when they are given from a TOML file. If the parameters are given in the dictionay format, they should be prepared as a nested dict under the solver key. In the following, the parameter items are described in the TOML format.

The input data consist of target reference data, and the bulk structure data. The output data consist of files containing the result of optimization. Their contents will be shown in this section.

4.1 Input parameters

Input parameters are specified in the solver section and the subsections (config, param, and reference).

- name

  Format: string

  Description: The name of the direct problem solver. Optional.

- dimension

  Format: integer

  Description: The number of parameters. If it is not specified, the dimension is taken from the [base] section.

4.1.1 [config] section

- path_to_first_solver

  Format: string

  Description: Path to the solver satl1.exe. The default value is satl1.exe.

- path_to_second_solver

  Format: string

  Description: Path to the solver satl2.exe. The default value is satl2.exe.

- sigma_file_path

  Format: string

  Description: Path to the experimental data file. The default value is exp.d.

- `remove_work_dir`

  Format: boolean

  Description: The output files of the solver program are stored in a directory `LogXXXX_YYYY` for each search point, in which `XXXX` denotes the step count and `YYYY` denotes the identifier such as replica number. When `remove_work_dir` is set to true, the directory is removed after the evaluation at the search point. The default value is false.

- `use_tmpdir`

  Format: boolean

  Description: If it is true, the intermediate files are stored in a directory within /tmp or within the directory specified by the environment variable TMPDIR. The default value is false.

4.1.2 [param] section

- `string_list`

  Format: list of strings

  Description: List of keywords embedded in the input template files.

4.1.3 [reference] section

- `path_to_base_dir`

  Format: string

  Description: Path to the directory which stores `exp.d`, `rfac.d`, `tleed4.i`, `tleed5.i`.

- `rfactor`

  Format: string or dictionary

  Description: Types of R-factor. The weighted average of multiple types of R-factors can be specified by a dictionary that relates the R-factor types and the relative weights. (For the latter, only the R-factor types r1 .. rpe are available.)

  The available types are described in the following section. The default value is `rpe` (Pendry R-factor).

- `rescale`

  Format: boolean

  Description: If it is set to true, I_t is rescaled in the R-factor calculation. The default value is false.

- `smoothing factor`

  Format: float

  Description: The smoothing parameter for I-V curve data. If it is set to 0.0, the smoothing is not applied. The default value is 0.0.

- `vi_value`

  Format: float

  Description: The imaginary part of the inner potential needed to calculate the Pendry R-factor. The parameter value can be explicitly specified by `vi_value`, or it is obtained from the input file `tleed5.i` for SATLEED.

4.2 Types of R-factor

The types of R-factors available for the `rfactor` parameter are listed as follows.

The notation is summarized: $\sum_b$ denotes the sum over the beams. $\sum_b{}' = \sum_b w_b$ denotes the sum over that beams that takes account of the relative weight of energy ranges $w_b = \int dE / \sum_b \int dE$. When `rescale` is set to true, $c = \int I_e dE / \int I_t dE$, otherwise $c = 1$.

- rsq

$$R = \sqrt{\frac{\sum_b \int (I_e - I_t)^2 \, dE}{\sum_b \int I_e^2 dE}}$$

- rsq_modified

$$R = \sqrt{\sum_b{}' \frac{\int (I_e - I_t)^2 \, dE}{\int I_e^2 dE}}$$

- r1

$$R = \sum_b{}' \frac{\int |I_e - cI_t| \, dE}{\int I_e dE}$$

- r2

$$R = \sum_b{}' \frac{\int (I_e - cI_t)^2 \, dE}{\int I_e^2 dE}$$

- rp1

$$R = \sum_b{}' \frac{\int |I'_e - cI'_t| \, dE}{\int |I'_e| \, dE}$$

- rp2

$$R = \sum_b{}' \frac{\int (I'_e - cI'_t)^2 \, dE}{\int (I'_e)^2 dE}$$

- rpp1

$$R = \sum_b{}' \frac{\int |I''_e - cI''_t| \, dE}{\int |I''_e| \, dE}$$

- rpp2

$$R = \sum_b{}' \frac{\int (I''_e - cI''_t)^2 \, dE}{\int (I''_e)^2 dE}$$

- rrzj (reduced Zanazzi-Jona R-factor)

$$R = \sum_b{}' \frac{1}{0.027 \int I_e dE} \int \frac{|I''_e - cI''_t| \cdot |I'_e - cI'_t|}{|I'_e| + \max |I'_e|} dE$$

- rmzj (modified Zanazzi-Jona R-factor)

$$R = \sum_b{}' \frac{1}{\int I_e'' dE} \int \frac{|I_e'' - cI_t''| \cdot |I_e' - cI_t'|}{|I_e'| + c \cdot \max|I_t'|} dE$$

- rpe (Pendry R-factor)

$$R = \sum_b{}' \frac{\int (Y_e - Y_t)^2 \, dE}{\int {Y_e}^2 + {Y_t}^2 dE}, \quad Y = \frac{L}{1 + V_{0i}^2 L^2}, \quad L = \tilde{I}'/\tilde{I}.$$

  In the Pendry R-factor, $\tilde{I} = I_t + \kappa\langle I \rangle$, where κ is the parameter denoted by `PERSH`, and it is set to 0.05. $\langle I \rangle$ is the average over the peaks of I. V_{0i} denotes the imaginary part of the inner potential.

  N.B. when RPE (the Pendry R-factor) is chosen in rfac.d for the R-factor (i.e. WR(10) is set to 1), the output of I-V curves contains $\tilde{I}$ instead of I. Therefore, WR(10) must be set to 0 except when `satleed` is chosen for `rfactor` type.

- satleed

  The value of R-factor is taken from the output (search.s) of satl2.exe.

4.3 Reference file for Solver

4.3.1 Target reference file

The file containing the data to be targeted to fit. Edit `tleed4.i` and `tleed5.i` in `path_to_base_dir` in the [`reference`] section.

The parameter values to be optimized are replaced by the keywords that depend on the types of parameters: The coordinates or the deviations of the coordinates are denoted by strings starting from `opt`. The inner potential is denoted by strings starting from `IP`. The Debye temperatures are denoted by strings starting from `debye`. The keywords are listed in the parameter `solver.param.string_list` in the input file. The number and the order of the keywords should be identical to those of the list of variables to optimize.

The keywords should be embedded in the templates according to the format of the parameters. In the following example, the parameter values corresponding to `opt000` and `opt001` (e.g. 0.23) are formatted by the format style `(F7.4)` (7 letters in total with 4 digits below the floating point) such as `␣0.2300`.

Note that if IFLAG and LSFLAG are not set to 0, the satleed side is also optimized.

An example file is shown below.

```
 1  0  0                          IPR ISTART LRFLAG
 1 10  0.02  0.2                  NSYM  NSYMS ASTEP VSTEP
 5  1  2  2                       NT0  NSET LSMAX LLCUT
 5                                NINSET
 1.0000 0.0000                  1      PQEX
 1.0000 2.0000                  2      PQEX
 1.0000 1.0000                  3      PQEX
 2.0000 2.0000                  4      PQEX
 2.0000 0.0000                  5      PQEX
 3                                 NDIM
opt000 0.0000 0.0000  0           DISP(1,j)  j=1,3
 0.0000opt001 0.0000  0           DISP(2,j)  j=1,3
 0.0000 0.0000 0.0000  1           DISP(3,j)  j=1,3
 0.0000 0.0000 0.0000  0           DISP(4,j)  j=1,3
```

(continues on next page)

(continued from previous page)

```
0.0000  0                          DVOPT  LSFLAG
3  0  0                            MFLAG NGRID NIV
...
```

4.4 Output file

In `leed`, the output files are stored in the subfolder `LogXXXX_YYYY` for each parameter value that is created in the folder with the rank number within the `output_dir`. Here, `XXXX` denotes the step count, and `YYYY` denotes the identifier such as the replica number.

CHAPTER

FIVE

ACKNOWLEDGEMENTS

The development of odatse-LEED was supported by “Project for advancement of software usability in materials science” (FY2020, 2021, 2024) of The Institute for Solid State Physics, The University of Tokyo.

For the implementation of the forward problem solver, we thank R. Ahmed (Kyushu Univ.), K. Wada(KEK), and T. Shirasawa(AIST).

CHAPTER

SIX

CONTACT

- Bug Reports

  Please report all problems and bugs on the github Issues page.

  To resolve bugs early, follow these guidelines when reporting:

  1. Please specify the version of ODAT-SE and odatse-LEED you are using.
  2. If there are problems for installation, please inform us about your operating system and the compiler.
  3. If a problem occurs during execution, enter the input file used for execution and its output.

  Thank you for your cooperation.

- Others

  If you have any questions about your research that are difficult to consult at Issues on GitHub, please send an e-mail to the following address:

  E-mail: `2dmat-dev__at__issp.u-tokyo.ac.jp` (replace _at_ by @)

BIBLIOGRAPHY

[SATLEED] M.A. Van Hove, W. Moritz, H. Over, P.J. Rous, A. Wander, A. Barbieri, N. Materer, U. Starke, G.A. Somorjai, Automated determination of complex surface structures by LEED, Surface Science Reports, Volume 19, 191-229 (1993).