ODAT-SE LEED Module Documentation

リリース *1.0.0*

ISSP, University of Tokyo

2025 年 04 月 11 日

Contents:

第1章　はじめに

1.1 ODAT-SE とは

ODAT-SE は、順問題ソルバーに対して探索アルゴリズムを適用して最適解を探すためのフレームワークです。順問題ソルバーはユーザー自身で定義できるほか、標準的な順問題ソルバーとして 2 次元物質構造解析向け実験データ解析ソフトウェアが用意されています。順問題ソルバーでは、原子位置などをパラメータとし得られたデータと実験データとのずれを損失関数として与えます。探索アルゴリズムによりこの損失関数を最小化する最適なパラメータを推定します。現バージョンでは、順問題ソルバーとして量子ビーム回折実験の全反射高速陽電子回折実験 (Total-Reflection High-Energy Positron Diffraction: TRHEPD), 表面 X 線回折法 (Surface X-ray Diffraction: SXRD), 低速電子線回折法 (Low Energy Electron Diffraction: LEED) に対応しており、探索アルゴリズムは Nelder-Mead 法, グリッド型探索法, ベイズ最適化, レプリカ交換モンテカルロ法, ポピュレーションアニーリングモンテカルロ法が実装されています。

1.2 odatse-LEED とは

SATLEED は M.A. Van Hove 氏により作成された LEED の解析を行うプログラムで、原子位置などから I-V curve を計算し、実験で得られた I-V curve からの誤差を求めます。odatse-LEED は、この SATLEED を ODAT-SE の順問題ソルバーとして利用するためのアダプタライブラリです。2DMAT v2.x の順問題ソルバーの一つとして開発されたコンポーネントを、独立なモジュールとして再構成したものです。ODAT-SE および SATLEED と組み合わせて使用します。SATLEED に関する詳細については [SATLEED] をご覧ください。

1.3 ライセンス

本ソフトウェアのプログラムパッケージおよびソースコード一式は GNU General Public License version 3（GPL v3）に準じて配布されています。

本ソフトウェアは 2020, 2024 年度 東京大学物性研究所 ソフトウェア高度化プロジェクトの支援を受け開発されました。ODAT-SE を引用する際には以下の文献を引用してください。

"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␣
↪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 バージョン履歴

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 主な開発者

ODAT-SE, odatse-LEED, 2DMAT は以下のメンバーで開発しています.

- ODAT-SE v3.0.0 -, odatse-LEED v1.0-alpha -
  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 青山 龍美 (東京大学 物性研究所)
  - 星 健夫　(核融合科学研究所)
- 2DMAT v2.0.0 -

  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 岩本 晴道 (鳥取大学 大学院工学研究科)
  - 星 健夫 (鳥取大学 大学院工学研究科)
- 2DMAT v0.1.0 -
  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 星 健夫 (鳥取大学 大学院工学研究科)

第2章　odatse-LEED のインストール

2.1 実行環境・必要なパッケージ

- python 3.9 以上
- python パッケージ
  - numpy (>= 1.14)
  - pydantic (>= 2.0)
  - fortranformat
  - ODAT-SE (>=3.0)
- SATLEED
  - コンパイルには Fortran コンパイラが必要

2.2 ダウンロード・インストール

1. ODAT-SE をインストールする
   - ソースコードからのインストール

     リポジトリから ODAT-SE のソースファイルを取得します。

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

     pip コマンドを実行してインストールします。

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

     `--user` オプションを付けるとローカル (`$HOME/.local`) にインストールできます。

     注釈

     Python 3.7 未満の環境では以下のようなエラーが発生します。本パッケージでは Python 3.9 以上を利用するようにしてください。

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

python3 -m pip install .[all] を実行するとオプションのパッケージも同時にインストールします。

2. odatse-LEED をインストールする

   odatse-LEED のソースファイルは GitHub リポジトリから取得できます。以下の手順でリポジトリをクローンした後、pip コマンドを実行してインストールします。

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

   --user オプションを付けるとローカル ($HOME/.local) にインストールできます。

   odatse-LEED のライブラリと、実行コマンド odatse-LEED がインストールされます。

3. SATLEED をインストールする

   - SATLEED のソースコードは以下の URL から取得できます。

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

   ファイルを展開し、所定の手続きに従ってコンパイルします。

   - SATLEED は計算したい系の詳細によってソースコードのパラメータを適宜書き換える必要があります。ステップ 2 で取得した odatse-LEED のソースファイルの sample/satleed ディレクトリに、サンプルを実行する場合の書き換え leedsatl.patch とコンパイルを自動で行うスクリプト setup.sh が用意されているので、それを利用しインストールすることもできます。

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

   setup.sh を実行すると、現在のディレクトリに satl1.exe と satl2.exe が作成されます。

   コンパイラやコンパイルオプションを変更する場合は setup.sh を編集してください。

2.3 実行方法

ODAT-SE では順問題ソルバと逆問題解析アルゴリズムを組み合わせて解析を行います。LEED の解析を行うには次の 2 通りの方法があります。

1. このパッケージに含まれる odatse-LEED プログラムを利用して解析を行います。ユーザは、プログラムの入力となるパラメータファイルを TOML 形式で作成し、プログラムの引数に指定してコマンドを実行します。逆問題解析のアルゴリズムはパラメータで選択できます。

2. odatse-LEED ライブラリと ODAT-SE フレームワークを用いてプログラムを作成し、解析を行います。逆問題解析アルゴリズムは import するモジュールで選択します。プログラム中に入力データの生成を組み込むなど、柔軟な使い方ができます。

パラメータの種類やライブラリの利用方法については以降の章で説明します。

2.4 アンインストール

odatse-LEED モジュールおよび ODAT-SE モジュールをアンインストールするには、以下のコマンドを実行します。

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

第3章　チュートリアル

順問題ソルバーとして用意されている odatse-LEED は M.A. Van Hove 氏により作成された低速電子線回折 (LEED) の解析ソフトウェア SATLEED を ODAT-SE で利用するモジュールとして作成されています。SATLEED では、与えられた原子座標に対して回折データをシミュレーションで計算します。これを順問題と見なしたとき、回折データが実験で与えられたときにそれを再現する原子座標を求める問題は逆問題に相当します。ODAT-SE はこの逆問題を解くフレームワークを提供します。

このチュートリアルでは、全探索法 (mapper) を用いてもっともらしい原子座標を推定する問題を例として odatse-LEED の使い方を説明します。以下では odatse-LEED に付属の odatse-LEED プログラムを利用し、TOML 形式のパラメータファイルを入力として解析を行います。次に、ユーザープログラムの項では、メインプログラムを自分で作成して使う方法を説明します。

3.1 グリッド型探索

ここでは、グリッド型探索を行い、回折データから原子座標を解析する方法について説明します。グリッド型探索は MPI に対応しています。探索グリッドを与えるデータ MeshData.txt を事前に準備する必要があります。

3.1.1 サンプルファイルの場所

サンプルファイルは sample/mapper にあります。フォルダには以下のファイルが格納されています。

- base ディレクトリ

  メインプログラムでの計算を進めるための参照ファイルを格納するディレクトリ。参照ファイルは exp.d, rfac.d, tleed4.i, tleed5.i です。

- input.toml

  メインプログラムの入力ファイル

- MeshData.txt

  探索グリッドのデータ

- ref_ColorMap.txt

  計算結果の参照値。計算が正しく実行されたか確認するための output/ColorMap の参照データ。

- prepare.sh , do.sh

  本チュートリアルを一括計算するために準備されたスクリプト

以下、これらのファイルについて説明したあと、実際の計算結果を紹介します。

3.1.2 参照ファイルの説明

tleed4.i, tleed5.i, rfac.d は satleed のパラメータファイルです。最適化する原子座標は tleed5.i の中で opt000, opt001 のようにキーワードとして埋め込みます。exp.d は参照する実験データです。

実際に探索するグリッドは MeshData.txt で与えます。サンプルでは MeshData.txt の中身は以下のようになっています。

```
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
...
```

1 列目が通し番号、2 列目以降は base/tleed5.i に入る opt000 , opt001 の値が順に指定されています。

注釈

MeshData.txt を作成するスクリプト make_meshdata.py が用意されています。

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

を実行するとメッシュデータが作成されます。

3.1.3 入力ファイルの説明

ここでは、メインプログラム用の入力ファイル input.toml について説明します。input.toml の詳細については入力ファイルに記載されています。以下は、サンプルファイルにある input.toml の中身になります。

```
[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"
```

(次のページに続く)

(前のページからの続き)

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

[runner]
ignore_error = true
```

最初に [base] セクションについて説明します。

- dimension は最適化したい変数の個数です。今の場合は base/tleed5.i で説明したように 2 つの変数の最適化を行うので、 2 を指定します。
- output_dir は出力先のディレクトリ名です。省略した場合はプログラムを実行したディレクトリになります。

[solver] セクションではメインプログラムの内部で使用するソルバーとその設定を指定します。

- name は使用したいソルバーの名前です。 leed に固定されています。

ソルバーの設定は、サブセクションの [solver.config], [solver.param], [solver.reference] で行います。

[solver.config] セクションではメインプログラム内部で呼び出す satl1.exe, satl2.exe についてのオプションを指定します。

- path_to_first_solver は satl1.exe のコマンド名です。パスを指定するか、コマンド名を PATH 環境変数から探索します。
- path_to_second_solver は satl2.exe のコマンド名です。パスを指定するか、コマンド名を PATH 環境変数から探索します。

[solver.param] セクションではパラメータについての指定を行います。

- string_list は埋め込みキーワードのリストを指定します。

[solver.reference] セクションでは参照データについての指定を行います。

- path_to_base_dir は参照データが置いてあるディレクトリ名を指定します。
- rfactor は R-factor の定義を指定します。デフォルトは rpe (Pendry R-factor) です。この例では satleed を指定して、SATLEED が計算した R-factor の値を使用します。

[algorithm] セクションでは、使用するアルゴリズムとその設定をします。

- name は使用したいアルゴリズムの名前です。このチュートリアルではグリッド探索による解析を行うので、 mapper を指定します。
- label_list は、 opt000, opt001 を出力する際につけるラベル名のリストです。

[algorithm.param] セクションでは探索アルゴリズムに関するパラメータを指定します。

- mesh_path は探索グリッドを記述したファイルを指定します。

[runner] セクションでは外部プログラム実行についての指定を行います。

- ignore_error を true に指定した場合、外部プログラムの実行と値の評価に関するエラーは NaN として扱い、計算を続行します。

その他、入力ファイルで指定可能なパラメータの詳細については入力ファイルの章をご覧ください。

3.1.4 計算実行

あらかじめ sample/satleed ディレクトリ内で setup.sh を実行して SATLEED をコンパイルし、 satl1.exe と satl2.exe を作成しておきます。

サンプルファイルが置いてあるフォルダへ移動します。

```
$ cd sample/mapper
```

メインプログラムを実行します。次のコマンドではプロセス数 4 の MPI 並列を用いた計算を行っています。計算は通常の PC で数分程度で終わります。

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

実行すると、 output ディレクトリとその下に各ランクのフォルダが作成され、計算結果が出力されます。また、以下の様なログが標準出力に表示されます。

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

z1, z2 に各メッシュでの候補パラメータと、その時の R-factor が出力されます。最終的にグリッド上の全ての点で計算された R-factor は、 ColorMap.txt に出力されます。今回の場合は

```
-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
...
```

のように得られます。1 列目、2 列目に opt000, opt001 の値が、3 列目に R-factor が記載されます。なお、メインプログラムを実行するスクリプトとして do.sh を用意しています。do.sh では ColorMap.dat と ref_ColorMap.dat の差分も比較しています。以下、説明は割愛しますが、その中身を掲載します。

```
#!/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 計算結果の可視化

ColorMap.txt を図示することで、 R-factor の小さいパラメータがどこにあるかを推定することができます。今回の場合は、以下のコマンドを実行すると 2 次元パラメータ空間の図 ColorMapFig.png が作成されます。

```
$ python3 plot_colormap_2d.py
```

作成された図を見ると、(0.0, 1.75) 付近に最小値を持っていることがわかります。

3.2 ユーザープログラムによる解析

ここでは、odatse-LEED モジュールを用いたユーザープログラムを作成し、解析を行う方法を説明します。逆問題アルゴリズムは例として Nelder-Mead 法を用います。

3.2.1 サンプルファイルの場所

サンプルファイルは sample/user_program にあります。フォルダには以下のファイルが格納されています。

- simple.py

  メインプログラム。パラメータを input.toml ファイルから読み込んで解析を行う。

- input.toml

  simple.py で利用する入力パラメータファイル

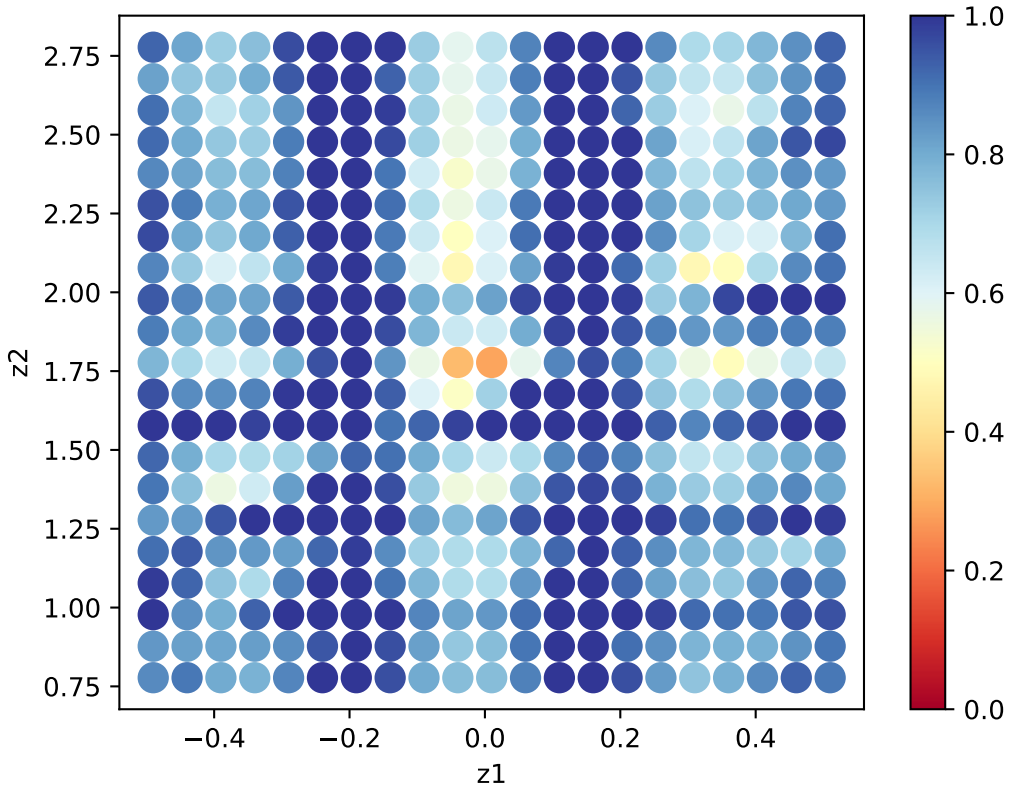

図 1: 2 次元パラメータ空間上での R-factor の値。(21 x 21 のメッシュで計算)

- base/

  メインプログラムでの計算を進めるための参照ファイルを格納するディレクトリ。参照ファイルは exp.d, rfac.d, tleed4.i, tleed5.i .

- ref.txt

  本チュートリアルで得られる結果の比較データ

- prepare.sh, do.sh

  本チュートリアルを一括計算するために準備されたスクリプト

以下、これらのファイルについて説明したのち、実際の計算結果を紹介します。

3.2.2 プログラムの説明

simple.py は odatse-LEED モジュールを用いて解析を行うシンプルなプログラムです。プログラムの全体を以下に示します。

```
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()
```

プログラムではまず、必要なモジュールを import します。

- ODAT-SE のメインモジュール odatse
- 今回利用する逆問題解析アルゴリズム odatse.algorithm.min_search
- 順問題ソルバーモジュール odatse.extra.LEED

次に、解析で利用するクラスのインスタンスを作成します。

- odatse.Info クラス

  パラメータを格納するクラスです。 from_file クラスメソッドに TOML ファイルのパスを渡して作成することができます。

- odatse.extra.LEED.Solver クラス

  odatse-LEED モジュールの順問題ソルバーです。Info クラスのインスタンスを渡して作成します。

- odatse.Runner クラス

  順問題ソルバーと逆問題解析アルゴリズムを繋ぐクラスです。Solver クラスのインスタンスおよび Info クラスのパラメータを渡して作成します。

- odatse.algorithm.min_search.Algorithm クラス

  逆問題解析アルゴリズムのクラスです。ここでは Nelder-Mead 法による最適化アルゴリズムのクラスモジュール min_search を利用します。Runner のインスタンスをわたして作成します。

Solver, Runner, Algorithm の順にインスタンスを作成した後、Algorithm クラスの main() メソッドを呼んで解析を行います。

上記のプログラムでは、入力パラメータを TOML 形式のファイルから読み込む形ですが、パラメータを dict 形式で渡すこともできます。

3.2.3 入力ファイルの説明

メインプログラム用の入力ファイル input.toml に、順問題ソルバーおよび逆問題解析アルゴリズムのパラメータを指定します。 base および solver セクションの内容は前述のグリッド型探索の場合と同じです。

逆問題解析アルゴリズムについては、Nelder-Mead 法のパラメータを algorithm.param の項目に指定します。なお、アルゴリズムの種類を指定する algorithm.name パラメータの値は無視されます。

- min_list, max_list は探索領域の指定で、領域の下端と上端を変数についてのリストの形式で与えます。
- initial_list と initial_scale_list は初期シンプレックスを指定するパラメータです。2 変数の例では、initial_list = [z1, z2], initial_scale_list = [dz1, dz2] を指定した場合、[z1, z2], [z1+dz1, z2], [z1, z2+dz2] を頂点とする三角形を初期シンプレックスにとります。

3.2.4 計算実行

あらかじめ sample/satleed ディレクトリ内で setup.sh を実行して SATLEED をコンパイルし、satl1.exe と satl2.exe を作成しておきます。

サンプルファイルが置いてあるフォルダへ移動します。

```
$ cd sample/user_program
```

メインプログラムを実行します。計算時間は通常の PC で数分程度で終わります。

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

実行すると、以下の様な出力がされます。

```
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
```

(次のページに続く)

(前のページからの続き)

```
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
...
```

x=[z1, z2] に各ステップでの候補パラメータと、その時の fun=R-factor が出力されます。最終的に推定されたパラメータは、 output/res.dat に出力されます。今の場合、

```
fx = 0.2101
z1 = -0.01991012930870084
z2 = 1.7509844067692764
```

となります。リファレンス ref.txt が再現されているか確かめてください。

なお、一連の計算を行う do.sh スクリプトが用意されています。

第4章　入出力

odatse-LEED モジュールは SATLEED を用いて原子位置などから Rocking curve を計算し、実験で得られた Rocking curve からの誤差を $f(x)$ として返す Solver です。

この章では、入力パラメータおよび入力データと出力データについて説明します。入力パラメータは Info クラスの solver の項目が該当します。TOML ファイルを入力として与える場合は [solver] セクションに記述します。dict 形式でパラメータを作成する場合は solver キー以下に入れ子の dict 形式でデータを用意します。以下では、TOML 形式でパラメータ項目を説明します。

入力データは、ターゲットとなる参照データとバルク構造データです。出力データは最適解の結果を格納したファイルです。以下の節で内容を示します。

4.1 入力パラメータ

solver セクションおよびサブセクション config, param, reference を利用します。

- name

  形式: string 型

  説明: 順問題ソルバーの名称を指定する。省略可。

- dimension

  形式: int 型

  説明: パラメータの次元を指定する。指定しない場合は [base] セクションの値が使われる。

4.1.1 [config] セクション

- path_to_first_solver

  形式: string 型

  説明: ソルバー satl1.exe へのパス。デフォルト値は satl1.exe 。

- path_to_second_solver

  形式: string 型

  説明: ソルバー satl2.exe へのパス。デフォルト値は satl2.exe 。

- sigma_file_path

  形式: string 型

  説明: 実験データを格納したファイルへのパス。デフォルト値は exp.d 。

- remove_work_dir

  形式: bool 型

  説明: 各探索点でのソルバーの出力ファイルは LogXXXX_YYYY というディレクトリに書き出される。XXXX はステップ数、YYYY はレプリカ等の識別子である。 remove_work_dir が True の場合、探索点の評価が終わった後にこのディレクトリを消去する。デフォルト値は False。

- use_tmpdir

  形式: bool 型

  説明: 中間ファイルを書き出すディレクトリを /tmp または環境変数 TMPDIR で指定されるディレクトリ内の一時ディレクトリに指定する。デフォルト値は False。

4.1.2 [param] セクション

- string_list

  形式: string 型のリスト

  説明: 入力ファイルのテンプレートに埋め込まれたキーワードのリスト

4.1.3 [reference] セクション

- path_to_base_dir

  形式: string 型

  説明: ソルバーを実行するための入力ファイル exp.d, rfac.d, tleed4.i, tleed5.i が格納されたディレクトリへのパス

- rfactor

  形式: string 型または辞書型

  説明: R-factor のタイプ。複数のタイプの荷重平均を指定することもでき、その場合はタイプと相対ウェイトの辞書型で指定する。(但し複数のタイプを指定できるのは r1 〜 rpe のみ)

  指定可能なタイプは後述する。デフォルト値は rpe (Pendry R-factor)。

- rescale

  形式: bool 型

  説明: True の場合、R-factor の計算において I_t のリスケールを行う。デフォルト値は False。

- smoothing_factor

  形式: float 型

  説明: I-V 曲線のスムージングを行う場合のパラメータ。0.0 の場合はスムージングを行わない。デフォルト値は 0.0。

- `vi_value`

  形式: float 型

  説明: Pendry R-factor を計算する際に必要な inner potential の虚部の値。 `vi_value` で明示的に指定した場合はこの値を利用する。指定しない場合は SATLEED の入力ファイル `tleed5.i` から取得する。

4.2 R-factor のタイプ

`rfactor` パラメータに指定可能なタイプは次のとおりです。

以下では、 $\sum_b$ はビームについての和を示し、 $\sum_b{}' = \sum_b w_b$ は各ビームのエネルギー範囲についての相対ウェイト $w_b = \int dE / \sum_b \int dE$ を考慮した和です。また、 `rescale` が True の場合は $c = \int I_e dE / \int I_t dE$ 、False の場合は $c = 1$ です。

- 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)

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

  Pendry R-factor では、 $\tilde{I} = I_t + \kappa\langle I \rangle$ です。 κ は PERSH で指定されるパラメータで 0.05 が使われています。 $\langle I \rangle$ は I のピークに関する平均です。また、 V_{0i} は inner potential の虚部を示します。

  ※留意: rfac.d の WR(10) パラメータが 1 の場合 (satl2.exe で Pendry R-factor を計算する場合)、出力される I-V 曲線データ iv 1〜 iv N は $\tilde{I}$ が出力されます。そのため、R-factor のタイプに satleed を指定する場合以外は rfac.d の WR(10) パラメータは 0 にする必要があります。

- satleed

  satl2.exe の出力 (search.s) から R-factor の値を取得します。

4.3 ソルバー用補助ファイル

4.3.1 ターゲット参照ファイル

ターゲットにするデータが格納されたファイル。 [reference] セクションの path_to_base_dir 中にある tleed4.i および tleed5.i を編集します。

最適化したいパラメータ値をキーワードに置き換えます。キーワードはパラメータの種別によって異なり、座標値や変位の場合は opt で始まる文字列、内部エネルギーの場合は IP で始まる文字列、Debye 温度の場合は debye で始まる文字列です。キーワードは入力ファイルの solver.param.string_list に列挙します。string_list の要素数・並び順は、最適化する値を入れる変数のリストの個数・順番と一致させる必要があります。

文字列は入力ファイルの書式に従ってフォーマットされた数値に置き換えられるので、書式に合わせてテンプレート中に埋め込む必要があります。以下の例では、 opt000, opt001 に対応する数値 (例えば 0.23) は Fortran の (F7.4) (全体の幅 7 桁、小数点以下 4 桁) に従って 崮 0.2300 のように整形されて置き換えられます。

なお、IFLAG, LSFLAG を 0 にしない場合は satleed 側での最適化も行われます。

以下、ファイル例を記載します。

```
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
```

(次のページに続く)

(前のページからの続き)

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

4.4 出力ファイル

leed では、 output_dir に指定する出力ディレクトリ内のランクの番号が記載されたフォルダ下に、評価点ごとのサブフォルダ LogXXXX_YYYY が作成され、その中に計算時に出力されるファイル一式が出力されます。XXXX はステップ数、 YYYY はレプリカ番号等の識別子です。

第5章　謝辞

本ソフトウェアは、東京大学物性研究所 ソフトウェア開発・高度化プロジェクト (2020, 2021, 2024 年度) の支援を受け開発されました。

順問題ソルバーの実装にあたり、R. Ahmed 氏 (九州大学)、和田健氏 (KEK)、白澤徹郎氏 (AIST) にお世話になりました。この場を借りて感謝します。

第6章　お問い合わせ

odatse-LEED に関するお問い合わせはこちらにお寄せください。

- バグ報告

  odatse-LEED のバグ関連の報告は GitHub の Issues で受け付けています。

  バグを早期に解決するため、報告時には次のガイドラインに従ってください。

  - 使用している ODAT-SE および odatse-LEED のバージョンを指定してください。
  - インストールに問題がある場合には、使用しているオペレーティングシステムとコンパイラの情報についてお知らせください。
  - 実行に問題が生じた場合は、実行に使用した入力ファイルとその出力を記載してください。

- その他

  研究に関連するトピックなど GitHub の Issues で相談しづらいことを問い合わせる際には、以下の連絡先にコンタクトをしてください。

  E-mail: `2dmat-dev__at__issp.u-tokyo.ac.jp` (_at_を@に変更してください)

関連図書

[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). https://doi.org/10.1016/0167-5729(93)90011-D