ODAT-SE XAFS 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て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法 (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-XAFS とは

偏光全反射蛍光 X 線吸収微細構造解析 (Polarization-dependent Total Reflection Fluorescence X-ray Absorption Fine Structure: PTRF-XAFS) は、X 線吸収スペクトルから得られる吸収原子の対称性や電子状態などを用いて物質構造解析を行う手法で、X 線吸収原子の周りに存在する原子の座標を得ることができます。特に全反射法を適用することにより表面の構造解析に有効な方法です。

X 線スペクトル解析のための第一原理計算ソフトウェアとして FEFF [1,2] が開発されています。原子位置を入力として X 線分光スペクトルの理論予測を行うプログラムで、Fortran で書かれており、標準的な Linux 環境で動作します。odatse-XAFS は、この FEFF を ODAT-SE の順問題ソルバ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 FEFF と組み合わせて使用します。

[1] Ab initio theory and calculation of X-ray spectra, J. J. Rehr, J. J. Kas, M. P. Prange, A. P. Sorini, Y. Takimoto, F. D. Vila, Comptes Rendus Physique 10 (6) 548-559 (2009).

[2] Theoretical Approaches to X-ray Absorption Fine Structure, J. J. Rehr and R. C. Albers, Rev. Mod. Phys. 72, 621 (2000).

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 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。2DMAT / 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-XAFS

- v1.0.0 : 2025-04-11

1.5 主な開発者

odatse-XAFS は以下のメンバーで開発しています.

- odatse-XAFS v1.0.0 -
  - 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)
  - 青山 龍美 (東京大学 物性研究所)
  - 中野 陽斗 (核融合科学研究所)
  - 星 健夫　(核融合科学研究所)

第2章　odatse-XAFS 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
  - 必要な python パッケージ
    - numpy (>= 1.14)
    - pydantic (>= 2.0)
- ODAT-SE version 3.0 以降
- FEFF 8.5light または version 9 以降 (有償版)

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。

     `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. FEFF をインストールする
   - 配布サイトからソースコードを取得し、コンパイルします。
     - コードの取得:

       ```
       $ git clone https://github.com/eucall-software/feff8.5light.git
       $ cd feff8.5light
       $ mkdir build && cd build
       ```

- GCC (gfortran) の場合はコンパイラオプションを指定してビルドする必要があります。

```
$ cmake -DCMAKE_Fortran_FLAGS="-fallow-argument-mismatch -std=legacy" ..
$ make
```

- intel compiler (ifort) の場合は次のようにコンパイラを指定してビルドします。

```
$ cmake -DCMAKE_Fortran_COMPILER=ifort ..
$ make
```

ビルドに成功すると実行形式 `feff85L` が作成されます。`feff85L` を PATH の通ったディレクトリ (環境変数 PATH に列挙された、実行プログラムを探索するディレクトリ) に配置するか、実行時にディレクトリ名をつけて指定します。

3. odatse-XAFS をインストールする

- ソースコードからのインストール

odatse-XAFS 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。

```
$ git clone https://github.com/2DMAT/odatse-XAFS.git
$ cd odatse-XAFS
$ 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-XAFS のライブラリと、実行コマンド `odatse-XAFS` がインストールされます。

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。XAFS 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-XAFS 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-XAFS 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-XAFS 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-XAFS 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-XAFS は、University of Washington の J. J. Rehr らにより開発された FEFF を ODAT-SE フレームワークで利用するためのモジュールです。FEFF は XAFS などの X 線分光スペクトルの理論予測を行うプログラムで、原子位置などのパラメータに対して多重散乱による第一原理計算を行いスペクトルを計算します。これを原子位置から分光スペクトルへの順問題としたとき、実験で得られたスペクトルを再現するように原子位置を推定する逆問題を考えます。ODAT-SE では、逆問題を解くためのアルゴリズムとして、以下の 5 つのアルゴリズムが用意されています。

- minsearch

  Nelder-Mead 法

- mapper_mpi

  与えられたパラメータの探索グリッドを全探索

- bayes

  ベイズ最適化

- exchange

  レプリカ交換モンテカルロ法

- pamc

  ポピュレーションアニーリング法

本チュートリアルでは、最初に順問題プログラム FEFF の実行方法を説明した後、 mapper_mpi による逆問題解析の実行方法について説明します。以下では odatse-XAFS に付属の odatse-XAFS 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実行します。

チュートリアルの最後に、ユーザープログラムの項で、メインプログラムを自分で作成して使う方法を minsearch を例に説明します。

3.1 XAFS 順問題ソルバー

最初に、チュートリアルを行うために必要な FEFF のインストールおよびテストを行います。

3.1.1 ダウンロード・インストール

odatse-XAFS のソースコードをリポジトリから取得します。

```
$ git clone https://github.com/2DMAT/odatse-XAFS.git
$ cd odatse-XAFS
```

FEFF 8.5 light のソースコード一式を配布サイトのリポジトリから取得し、ビルドします。セットアップスクリプトが sample/feff/setup.sh に用意されています。

```
$ cd sample/feff
$ sh ./setup.sh
```

ビルドが成功すると、現在のディレクトリ内に `feff85L` が作成されます。

> **注釈**
>
> intel Fortran コンパイラを利用する場合は、setup.sh の内容を書き換えて実行してください。

> **注釈**
>
> 上記のセットアップスクリプトでは、FEFF が出力する中間ファイルのサイズを減らすため、不要なファイルの出力を抑制するパッチを適用しています。

3.1.2 計算実行

このチュートリアルでは実際に FEFF を使った計算をしてみます。サンプルとなる入力ファイルは odatse-XAFS の `sample/solver` 以下にあります。

```
$ cd sample/solver
```

次に `feff85L` を現在のディレクトリにコピーします。

```
$ cp ../feff/feff85L .
```

`feff85L` を実行します。入力ファイルは `feff.inp` (ファイル名は固定) です。

```
$ ./feff85L
```

以下のログが表示され、多数の出力ファイルが生成されます。

```
Feff 8.50L
Sample_data
Calculating potentials ...
   free atom potential and density for atom type    0
   free atom potential and density for atom type    1
   free atom potential and density for atom type    2
   initial state energy
   overlapped potential and density for unique potential    0
   overlapped potential and density for unique potential    1
   overlapped potential and density for unique potential    2
   muffin tin radii and interstitial parameters
```

(次のページに続く)

(前のページからの続き)

```
iph, rnrm(iph)*bohr, rmt(iph)*bohr, folp(iph)
   0  1.52927E+00  1.20700E+00  1.15000E+00
   1  1.64543E+00  1.30998E+00  1.15000E+00
   2  1.24843E+00  9.73397E-01  1.15000E+00
mu_old=    -0.301
Done with module 1: potentials.
Calculating cross-section and phases...
 0.579139710436025       4.256845921991644E-004  2.895698552180123E-002
         10
   absorption cross section
dx=  5.000000000000000E-002
   phase shifts for unique potential    0
   phase shifts for unique potential    1
   phase shifts for unique potential    2
Done with module 2: cross-section and phases...
Preparing plane wave scattering amplitudes...
Searching for paths...
  WARNING:  rmax > distance to most distant atom.
            Some paths may be missing.
            rmax, ratx   6.00000E+00  0.00000E+00
   Rmax  6.0000  keep and heap limits   0.0000000   0.0000000
   Preparing neighbor table
   Paths found        2   (maxheap, maxscatt       1   1)
Eliminating path degeneracies...
   Plane wave chi amplitude filter   2.50%
   Unique paths      2,  total paths       2
Done with module 4: pathfinder.
Calculating EXAFS parameters...
doing ip =            1
   Curved wave chi amplitude ratio   4.00%
   Discard feff.dat for paths with cw ratio <   2.67%
   path  cw ratio     deg    nleg  reff
     1     0.1000E+03     1.000     2   1.8833
     2     0.3835E+02     1.000     2   2.1978
   2 paths kept,    2 examined.
Done with module 5: F_eff.
Calculating chi...
feffdt, feff.bin to feff.dat conversion Feff 8.50L
Sample_data                                                    Feff 8.50L
POT  Non-SCF, core-hole, AFOLP (folp(0)= 1.150)
Abs   Z=28 Rmt= 1.207 Rnm= 1.529 K  shell
Pot 1 Z=16 Rmt= 1.310 Rnm= 1.645
Pot 2 Z= 8 Rmt= 0.973 Rnm= 1.248
Gam_ch=1.576E+00 H-L exch Vi= 0.000E+00 Vr=-5.000E+00
Mu=-3.013E-01 kf=1.695E+00 Vint=-1.125E+01 Rs_int= 2.140
```

(次のページに続く)

(前のページからの続き)

```
PATH  Rmax= 6.000,  Keep_limit= 0.00, Heap_limit 0.00  Pwcrit= 2.50%
      2 paths to process
   path     filename
      1     feff0001.dat
      2     feff0002.dat
   Use all paths with cw amplitude ratio   4.00%
   S02  1.000  Global sig2  0.00160
Done with module 6: DW + final sum over paths.
```

```
$ ls
atoms.dat     feff0002.dat  fpf0.dat    log1.dat  misc.dat  mod5.inp   pot.bin
chi.dat       feff85L       geom.dat    log2.dat  mod1.inp  mod6.inp   run.log
feff.bin      files.dat     global.dat  log4.dat  mod2.inp  mpse.dat   s02.inp
feff.inp      fort.38       list.dat    log5.dat  mod3.inp  paths.dat  xmu.dat
feff0001.dat  fort.39       log.dat     log6.dat  mod4.inp  phase.bin  xsect.bin
```

3.1.3 計算結果の可視化

出力ファイルのうち、分光スペクトルで参照するデータは chi.dat です。内容は次のようになっています。

```
# Sample_data                                                    Feff 8.50L
# POT  Non-SCF, core-hole, AFOLP (folp(0)= 1.150)
# Abs   Z=28 Rmt= 1.207 Rnm= 1.529 K  shell
# Pot 1 Z=16 Rmt= 1.310 Rnm= 1.645
# Pot 2 Z= 8 Rmt= 0.973 Rnm= 1.248
# Gam_ch=1.576E+00 H-L exch Vi= 0.000E+00 Vr=-5.000E+00
# Mu=-3.013E-01 kf=1.695E+00 Vint=-1.125E+01 Rs_int= 2.140
# PATH  Rmax= 6.000,  Keep_limit= 0.00, Heap_limit 0.00  Pwcrit= 2.50%
#  S02=1.000                                          Global_sig2= 0.00160
#  Curved wave amplitude ratio filter   4.000%
#     file         sig2 tot  cw amp ratio   deg  nlegs   reff  inp sig2
#           1       0.00160    100.00      1.00     2   1.8833
#           2       0.00160     38.35      1.00     2   2.1978
#     2/   2 paths used
#  ---------------------------------------------------------------------
#       k          chi          mag           phase @#
     0.0500    5.362812E-02  2.330458E-01  2.321993E-01
     0.1000    5.425108E-02  2.327328E-01  2.352690E-01
     0.1500    5.608660E-02  2.318076E-01  2.443784E-01
     0.2000    5.790322E-02  2.308804E-01  2.534995E-01
     0.2500    6.083927E-02  2.293763E-01  2.684505E-01
     0.3000    6.372689E-02  2.278648E-01  2.834501E-01
     0.3500    6.758157E-02  2.258418E-01  3.038991E-01
     0.4000    7.135477E-02  2.237971E-01  3.245020E-01
```

(次のページに続く)

(前のページからの続き)

```
    0.4500    7.589064E-02  2.213458E-01  3.499599E-01
    0.5000    8.032632E-02  2.188382E-01  3.758443E-01
    0.5500    8.527557E-02  2.160851E-01  4.056746E-01
    0.6000    9.014607E-02  2.132012E-01  4.365566E-01
    0.6500    9.528167E-02  2.103058E-01  4.701976E-01
    0.7000    1.003518E-01  2.071479E-01  5.057289E-01
    0.7500    1.057128E-01  2.043406E-01  5.437356E-01

(以下略)
```

で始まるコメント行には計算時のモデルなどの情報が記載されます。それに続いて閾値 ($k = 0$) から始まる $k, \chi(k), |\chi(k)|$, 位相の情報が書き出されます。以下に $\chi(k)$ をプロットした図を示します。

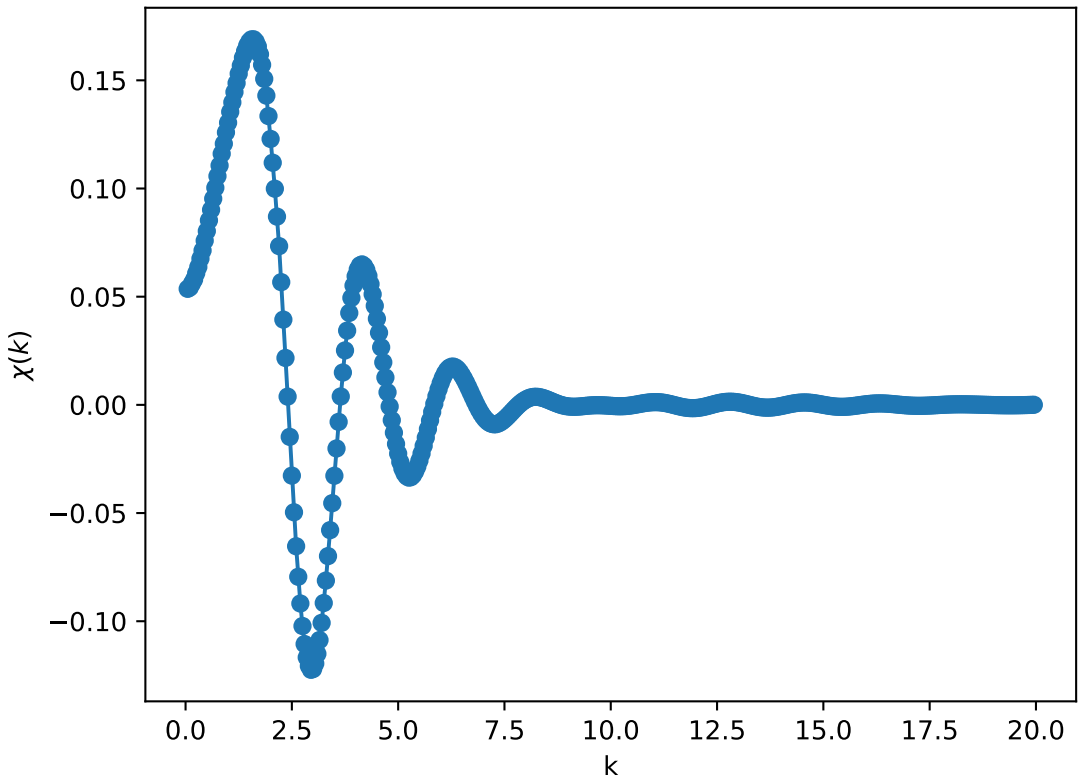

図 1: XAFS スペクトルの計算例

3.2 グリッド型探索

ここでは、グリッド型探索を行い、分光スペクトル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。探索グリッドは入力パラメータをもとに等間隔のメッシュとして生成します。

3.2.1 サンプルファイルの場所

サンプルファイルは sample/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。

- mock_data.txt, template.txt

  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

- ref_ColorMap.txt

  計算が正しく実行されたか確認するためのファイル (本チュートリアルを行うことで得られる ColorMap.txt の回答)。

- input.toml

  メインプログラムの入力ファイル

- 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。

3.2.2 参照ファイルの説明

template.txt は FEFF の入力ファイルのテンプレートです。ここでは、S 原子の座標 x, y, z のうち、計算量を減らすために z = -1.60 に固定して x, y の 2 次元空間の探索を行います。以下にファイルの内容を示しますが、 @x, @y が動かすパラメータに相当します。

リファレンスとなる測定データを模したデータは mock_data.txt です。偏光方向を 3 通りにとったスペクトルデータが格納されています。

```
* This feff.inp file generated by ATOMS, version 2.50
* ATOMS written by and copyright (c) Bruce Ravel, 1992-1999

* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       total mu =      725.4 cm^-1, delta mu =      610.0 cm^-1
*       specific gravity = 12.006, cluster contains   55 atoms.
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       mcmaster corrections:  0.00020 ang^2 and  0.770E-07 ang^4
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *

TITLE   Sample_data

EDGE      K
S02       1.0

*         pot    xsph  fms   paths genfmt ff2chi
CONTROL   1      1      1      1      1      1
PRINT     0      0      0      0      0      0
```

(次のページに続く)

(前のページからの続き)

```
*         r_scf   [ l_scf  n_scf  ca ]
*SCF       6.05142     0     15     0.1


EXAFS    20
RPATH    6

*         kmax  [ delta_k  delta_e ]
*XANES     4.0     0.07     0.5
*         r_fms     [ l_fms ]
*FMS       6.05142    0
*
*RPATH     0.10000
*         emin  emax  resolution
*LDOS      -20    20   0.1


POTENTIALS
*   ipot   z [ label   l_scmt  l_fms  stoichiometry ]
       0   28    Ni
       1   16    S
       2    8     O
NLEG         2

*CRITERIA     4.00     2.50

*DEBYE        300.00   340.00

* CORRECTION 4.50 0.5
* RMULTIPLIER 1.00
* ION 0 0.2
* ION 1 0.2

*         ixc  [ Vr  Vi ]
EXCHANGE  0    -5    0
SIG2  0.0016
POLARIZATION   @Ex @Ey @Ez

ATOMS
0.0000 0.0000 0.0000 0 Ni
@x @y -1.6000 1 S
1.1400 1.2800 0.9700 2 O
```

3.2.3 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。パラメータの詳細についてはマニュアルの入出力の章を参照してください。以下は、サンプルファイルにある input.toml の内容です。

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

[solver]
name = "feff"

[solver.config]
feff_exec_file  = "feff85L"
feff_output_file = "chi.dat"
#remove_work_dir = true
#use_tmpdir = true

[solver.param]
string_list = ["@x", "@y"]
polarization_list = ["@Ex", "@Ey", "@Ez"]
polarization = [ [0,1,0], [1,0,0], [0,0,1] ]
calculated_first_k = 3.6
calculated_last_k = 10

[solver.reference]
path_epsilon = "mock_data.txt"

[algorithm]
name = "mapper"
label_list = ["x_S", "y_S"]

[algorithm.param]
min_list = [-2.0, -2.0]
max_list = [ 2.0,  2.0]
num_list = [21, 21]
```

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、今の場合は template.txt 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。 feff 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 feff85L に関するオプションを指定します。

- feff_exec_file は FEFF の実行ファイルのパスを指定します。
- feff_output_file は FEFF の出力ファイルのうち、XAFS スペクトルデータを格納するファイルを指定します。
- remove_work_dir は各 iteration ごとに FEFF の出力ファイルを削除するオプションです。
- use_tmpdir は FEFF の出力ファイルの書き出し先を /tmp にするオプションです。

[solver.param] セクションでは FEFF の入力に関するオプションを指定します。

- string_list は、 template.txt で読み込む、動かしたい変数の名前のリストです。
- polarization_list は、 template.txt で読み込む、偏光ベクトルの要素の名前のリストです。
- polarization は、偏光ベクトルのセットを指定します。
- calculated_first_k, calculated_last_k は、計算結果と測定データを比較する際の波数 k の範囲 (下端・上端) を指定します。

[solver.reference] セクションでは、リファレンスとなる測定データに関するオプションを指定します。

- path_epsilon は実験データが置いてあるパスを指定します。

[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、変数の値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、探索するパラメータの範囲や初期値を指定します。

- min_list, max_list, num_list は、探索グリッドの範囲と分割数を指定します。

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出力の章をご覧ください。

3.2.4 計算実行

最初にサンプルファイルが置いてあるフォルダへ移動します (以下、本ソフトウェアをダウンロードしたディレクトリ直下にいることを仮定します).

```
$ cd sample/mapper
```

順問題の時と同様に feff85L をコピーします。

```
$ cp ../feff/feff85L .
```

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

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

実行すると、output ディレクトリ内に各ランクのフォルダが作成され、その中にグリッドの id がついたサブフォルダ LogXXXX_00000000 (XXXX がグリッドの id) が作成されます。また、以下の様な出力が標準出力に書き出されます。

```
name            : mapper
label_list      : ['x_S', 'y_S']
param.min_list  : [-2, -2]
param.max_list  : [2, 2]
param.num_list  : [21, 21]
Iteration : 1/441
@x = -2.00000000
@y = -2.00000000
R-factor = 19.739646449543752 Polarization [0.0, 1.0, 0.0] R-factor1 = 2.
↪23082630928769  Polarization [1.0, 0.0, 0.0] R-factor2 = 3.745102742186708 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 53.243010297156864
Iteration : 2/441
@x = -1.80000000
@y = -2.00000000
R-factor = 15.870615265918195 Polarization [0.0, 1.0, 0.0] R-factor1 = 2.
↪465225144249503  Polarization [1.0, 0.0, 0.0] R-factor2 = 3.7116841611214517 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 41.43493649238363
Iteration : 3/441
@x = -1.60000000
@y = -2.00000000
R-factor = 12.4966032440396 Polarization [0.0, 1.0, 0.0] R-factor1 = 3.
↪4464214082242046  Polarization [1.0, 0.0, 0.0] R-factor2 = 2.6218600524063693 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 31.421528271488228
Iteration : 4/441
@x = -1.40000000
@y = -2.00000000
R-factor = 11.698213396270965 Polarization [0.0, 1.0, 0.0] R-factor1 = 3.
↪4791684719050933  Polarization [1.0, 0.0, 0.0] R-factor2 = 1.6240174174998872 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 29.991454299407913
Iteration : 5/441
@x = -1.20000000
@y = -2.00000000
R-factor = 14.299726412681139 Polarization [0.0, 1.0, 0.0] R-factor1 = 2.
↪2280314879817467  Polarization [1.0, 0.0, 0.0] R-factor2 = 1.5332463231108493 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 39.13790142695082
Iteration : 6/441
@x = -1.00000000
@y = -2.00000000
R-factor = 21.44097816422594 Polarization [0.0, 1.0, 0.0] R-factor1 = 3.
↪7563622860968673  Polarization [1.0, 0.0, 0.0] R-factor2 = 1.810765574876649 ␣
```

(次のページに続く)

(前のページからの続き)

```
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 58.7558066317043
Iteration : 7/441
@x = -0.80000000
@y = -2.00000000
R-factor = 28.455902096414444 Polarization [0.0, 1.0, 0.0] R-factor1 = 6.
↪512305703044855  Polarization [1.0, 0.0, 0.0] R-factor2 = 2.004528093101423 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 76.85087249309706
...
```

@x, @y 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 output/ColorMap.txt に出力されます。今回の場合は

```
-2.000000 -2.000000 19.739646
-1.800000 -2.000000 15.870615
-1.600000 -2.000000 12.496603
-1.400000 -2.000000 11.698213
-1.200000 -2.000000 14.299726
-1.000000 -2.000000 21.440978
-0.800000 -2.000000 28.455902
...
```

のように得られます。1, 2 列目に @x, @y の値が、3 列目に R-factor が記載されます。

なお、一括計算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-XAFS 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
```

3.2.5 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。以下のコマンドを入力すると 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 -o ColorMapFig.png
```

作成された図を見ると、(±1.2, ±0.8) 付近に最小値があることがわかります。

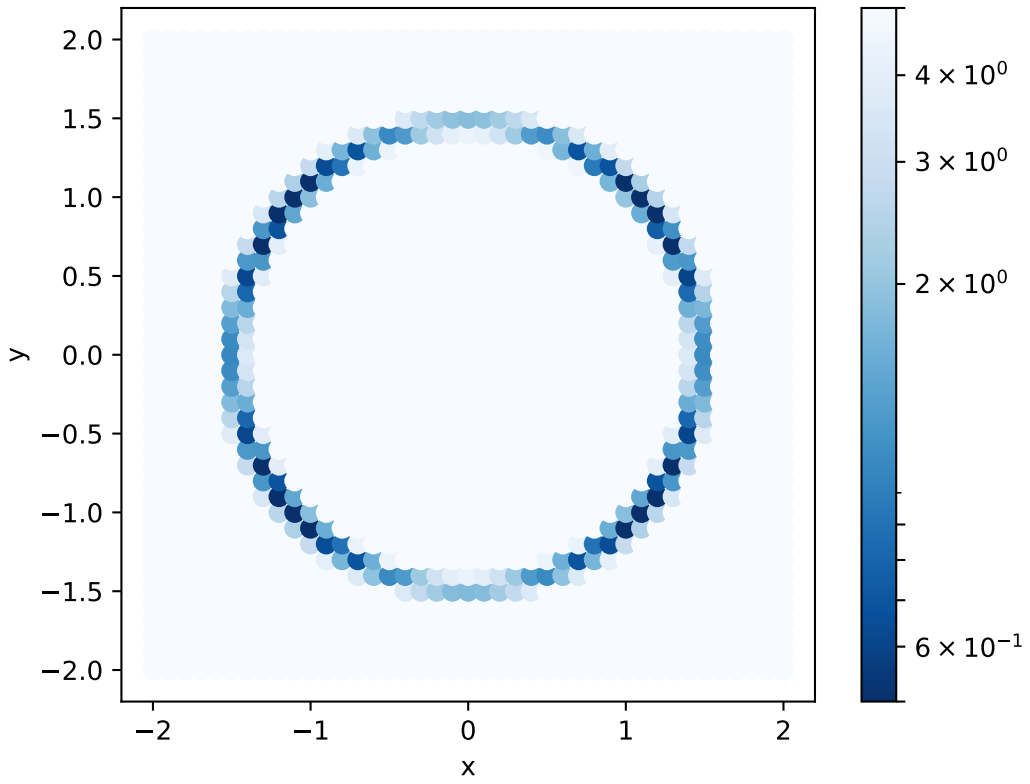

図 2: S 原子の x, y 座標に対する R-factor のプロット。z = -1.60 に固定。

3.3 ユーザープログラムによる解析

ここでは、odatse-XAFS モジュールを用いたユーザ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。

3.3.1 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

- mock_data.txt, template.txt

  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

- ref.txt

  本チュートリアルで求めたい回答が記載されたファイル

- 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。

3.3.2 プログラムの説明

simple.py は odatse-XAFS 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 numpy as np

import odatse
import odatse.algorithm.min_search
from odatse.extra.XAFS 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.XAFS

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.XAFS.Solver クラス

  odatse-XAFS 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 のインスタンスを渡して作成します。

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。

3.3.3 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 は前述のグリッド探索法による最適化で用いたのとほぼ同じファイルを利用できます。algorithm.param セクションには探索範囲 min_list, max_list と初期値 initial_list を指定します。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。

その他、入力ファイルのテンプレートや測定データファイルは前述の tutorial と同様です。

3.3.4 計算実行

最初にサンプルファイルが置いてあるフォルダへ移動します (以下、本ソフトウェアをダウンロードしたディレクトリ直下にいることを仮定します).

```
$ cd sample/user_program
```

feff85L をコピーします。

```
$ cp ../feff/feff85L .
```

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。

```
name            : minsearch
label_list      : ['x_S', 'y_S', 'z_S']
param.min_list  : [-2.0, -2.0, -2.0]
param.max_list  : [2.0, 2.0, 2.0]
param.initial_list: [1.12, 0.96, -1.57]
value_01 = 1.12000000
value_02 = 0.96000000
value_03 = -1.57000000
R-factor = 0.7762817472030608 Polarization [0.0, 1.0, 0.0] R-factor1 = 0.
↪2621870579175361 6  Polarization [1.0, 0.0, 0.0] R-factor2 = 0.3520528886264926 ␣
```

(次のページに続く)

(前のページからの続き)

```
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 1.7146052950651536
value_01 = 1.12000000
value_02 = 0.96000000
value_03 = -1.57000000
R-factor = 0.7762817472030608 Polarization [0.0, 1.0, 0.0] R-factor1 = 0.
↪26218705791753616  Polarization [1.0, 0.0, 0.0] R-factor2 = 0.3520528886264926 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 1.7146052950651536
value_01 = 1.37000000
value_02 = 0.96000000
value_03 = -1.57000000
R-factor = 7.006992151846735 Polarization [0.0, 1.0, 0.0] R-factor1 = 2.
↪097374935360426  Polarization [1.0, 0.0, 0.0] R-factor2 = 3.7994697622892972 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 15.124131757890481
value_01 = 1.12000000
value_02 = 1.21000000
value_03 = -1.57000000
R-factor = 5.73016510319226 Polarization [0.0, 1.0, 0.0] R-factor1 = 3.
↪5024285153919115  Polarization [1.0, 0.0, 0.0] R-factor2 = 2.0041674778416843 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 11.683899316343183
value_01 = 1.12000000
value_02 = 0.96000000
value_03 = -1.32000000
R-factor = 56.31862514558586 Polarization [0.0, 1.0, 0.0] R-factor1 = 6.
↪4008790163862015  Polarization [1.0, 0.0, 0.0] R-factor2 = 9.109651695414557 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 153.44534472495684
value_01 = 1.28666667
value_02 = 1.12666667
value_03 = -1.82000000
R-factor = 32.91890925644038 Polarization [0.0, 1.0, 0.0] R-factor1 = 2.
↪2301127998431753  Polarization [1.0, 0.0, 0.0] R-factor2 = 5.050496886392638 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 91.47611808308534
value_01 = 1.24500000
value_02 = 1.08500000
value_03 = -1.69500000
R-factor = 14.218592101199897 Polarization [0.0, 1.0, 0.0] R-factor1 = 3.
↪3863023885318193  Polarization [1.0, 0.0, 0.0] R-factor2 = 5.0060964449177945 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 34.26337747015008
eval: x=[ 1.12  0.96 -1.57], fun=0.7762817472030608
value_01 = 1.16166667
value_02 = 1.00166667
value_03 = -1.44500000
R-factor = 7.048842595863635 Polarization [0.0, 1.0, 0.0] R-factor1 = 1.
↪224939069135288  Polarization [1.0, 0.0, 0.0] R-factor2 = 1.8540984688270858 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 18.06749024962853
value_01 = 1.18250000
```

(次のページに続く)

(前のページからの続き)

```
value_02 = 1.02250000
value_03 = -1.50750000
R-factor = 0.4469433592171206 Polarization [0.0, 1.0, 0.0] R-factor1 = 0.
↪2535379790399123  Polarization [1.0, 0.0, 0.0] R-factor2 = 0.20069518356682897 ␣
↪Polarization [0.0, 0.0, 1.0] R-factor3 = 0.8865969150446205
eval: x=[ 1.1825  1.0225 -1.5075], fun=0.4469433592171206
...
```

value_01, value_02, value_03 に各ステップでの候補パラメータと、その時の R-factor が出力されます。また各ステップでの計算結果は output/0/LogXXXX_YYYY (XXXX, YYYY はステップ数) のフォルダに出力されます。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 = 0.20606977805890725
x_S = 1.125299780290939
y_S = 0.9597181918334485
z_S = -1.596967599355829
```

となります。リファレンス ref.txt が再現されていることが分かります。

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-XAFS モジュールは FEFF を用いて原子位置 x からスペクトルを計算し、実験で得られた XAFS スペクトルからの誤差を $f(x)$ 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、ターゲットとする参照データと、FEFF の入力ファイルのテンプレートです。出力データは、FEFF の feff85L が出力する結果およびログファイルです。以下の節で内容を示します。

4.1 入力パラメータ

solver セクションとセクション中のサブセクション config, param, reference を利用します。

4.1.1 [solver] セクション

- dimension

  形式: 整数型

  説明: パラメータの次元数を指定します。solver セクションまたは base セクションに記述する必要があり、両方指定した場合は solver セクションの値が優先されます。 string_list の要素数と一致している必要があります。

4.1.2 [solver.config] セクション

- feff_exec_file

  形式: string 型 (default: "feff8.5L")

  説明: FEFF ソルバーへのパスを指定します。

- feff_input_file

  形式: string 型 (default: "feff.inp")

  説明: FEFF ソルバーの入力ファイルのファイル名を指定します。 feff85L の場合は feff.inp 固定です。

- feff_output_file

  形式: string 型 (default: "chi.dat")

説明: FEFF の出力ファイルのうち、X 線吸収スペクトルデータのファイル名を指定します。 `feff85L` の場合は `chi.dat` 固定です。

- `feff_template_file`

  形式: string 型 (default: "template.txt")

  説明: FEFF 入力ファイルのテンプレートのファイル名を指定します。

- `remove_work_dir`

  形式: bool 型 (default: false)

  説明: FEFF の出力ファイルを削除するかどうかを指定します。

- `use_tmpdir`

  形式: bool 型 (default: false)

  説明: FEFF の出力ファイルの書き出し先として /tmp (または TMPDIR 環境変数で指定するディレクトリ) 内の一時的なディレクトリを使用するかどうかを指定します。

4.1.3 [solver.param] セクション

- `string_list`

  形式: string 型のリスト (default: ["value_01", "value_02"])

  説明: ソルバーの入力ファイルを作成するための参照用テンプレートファイルで利用するプレースホルダーのリスト。リストの要素数は `dimension` の値と一致する必要があります。これらの文字列が探索中のパラメータの値に置換されます。

- `polarization_list`

  形式: string 型の 3 要素のリスト

  説明: ソルバーの入力ファイル用のテンプレートファイル内で偏光ベクトルの成分のプレースホルダーとして使用する文字列のリスト。これらの文字列が偏光ベクトルの各要素に置換されます。

- `polarization`

  形式: 3 要素からなる float 型のリストのリスト

  説明: 偏光ベクトルのリストを指定します。

- `calculated_first_k`

  形式: float 型

  説明: FEFF による計算値と測定データとの比較を行う波数の範囲について下端を指定します。

- `calculated_last_k`

  形式: float 型

  説明: FEFF による計算値と測定データとの比較を行う波数の範囲について上端を指定します。

- k_range

  形式: float 型のリスト

  説明: FEFF による計算値と測定データとの比較を行う波数の範囲を [下端, 上端] の形式のリストとして指定します。 calculated_first_k, calculated_last_k による指定とは排他的で、どちらかの形式で範囲を指定する必要があります。

4.1.4 [solver.reference] セクション

- path_epsilon

  形式: string 型

  説明: 実験データファイルへのパスを指定します。

4.2 ソルバー用補助ファイル

4.2.1 入力テンプレートファイル

入力テンプレートファイル template.txt は surf.exe の入力ファイルを作成するためのテンプレートです。動かすパラメータ (求めたい原子座標などの値) を value_* などの文字列に置き換えます。使用する文字列は入力ファイルの [solver.param] セクションにある string_list で指定します。偏光ベクトルの要素についても同様に polarization_* などの文字列に置き換え、使用する文字列を [solver.param] セクションの polarization_list に指定します。

以下、テンプレートの例を記載します。

```
* This feff.inp file generated by ATOMS, version 2.50
* ATOMS written by and copyright (c) Bruce Ravel, 1992-1999

* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       total mu =      725.4 cm^-1, delta mu =      610.0 cm^-1
*       specific gravity = 12.006, cluster contains   55 atoms.
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       mcmaster corrections:  0.00020 ang^2 and  0.770E-07 ang^4
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *

TITLE   Sample_data

EDGE      K
S02       1.0

*         pot    xsph  fms   paths genfmt ff2chi
CONTROL   1      1      1      1      1      1
PRINT     0      0      0      0      0      0

*         r_scf   [ l_scf  n_scf  ca ]
```

(次のページに続く)

(前のページからの続き)

```
*SCF       6.05142     0     15     0.1


EXAFS    20
RPATH    6

*         kmax  [ delta_k  delta_e ]
*XANES     4.0     0.07     0.5
*         r_fms     [ l_fms ]
*FMS       6.05142    0
*
*RPATH     0.10000
*         emin  emax  resolution
*LDOS      -20    20   0.1


POTENTIALS
*   ipot   z [ label   l_scmt  l_fms  stoichiometry ]
      0   28    Ni
      1   16    S
      2    8     O
NLEG         2

*CRITERIA     4.00     2.50

*DEBYE        300.00   340.00

* CORRECTION 4.50 0.5
* RMULTIPLIER 1.00
* ION 0 0.2
* ION 1 0.2

*         ixc  [ Vr  Vi ]
EXCHANGE  0    -5    0
SIG2  0.0016
POLARIZATION   polarization_01 polarization_02 polarization_03

ATOMS
0.0000 0.0000 0.0000 0 Ni
value_01 value_02 value_03 1 S
1.1400 1.2800 0.9700 2 O
```

この場合、 value_01, value_02, value_03 が動かすパラメータ、 polarization_01, polarization_02, polarization_03 が偏光ベクトルの要素になります。FEFF 入力ファイルの書式については FEFF のマニュアルを参照してください。

4.2.2 ターゲット参照ファイル

ターゲットとする測定データが格納されたファイルを [solver.reference] セクションの path_epsilon に指定します。ファイルの書式は、第一列に波数、第二列以降はスペクトル強度と不確かさの組が偏光方向ごとに格納されています。最初の 2 行はヘッダです。以下、ファイルの例を示します。

```
k    c(k)_E[001]
(Ni09sum000-004k    e(k)_E[001]             Ni11sum000-004k_sum140521_E1    e(k)_E[1-
↪10]            Ni13dd_sum000-004k_d140617_t    e(k)_E[110]
3.5           -0.02335000             0.006999908                    -0.04765000
        0.007511923                     -0.04365000             0.007200607
3.55          -0.01203000             0.009141367                    -0.03033000
        0.010077591                     -0.03322000             0.009255752
3.6           -0.00198000             0.008535745                    -0.02501000
        0.008242841                     -0.02414000             0.007907668
...
```

4.2.3 出力ファイル

odatse-XAFS では、 feff85L により出力されるファイルが、ランクの番号が記載されたフォルダ下にある Log%%%%%_##### フォルダ以下の call_01, call_02, call_03 ディレクトリに一式出力されます。(use_tmpdir が True の場合は、/tmp または TMPDIR 環境変数で指定する一時ディレクトリ内に書き出されます。) %%%%% はアルゴリズムの反復回数 step (例：モンテカルロステップ数)、##### はアルゴリズムにおけるグループの番号 set (例：モンテカルロにおけるレプリカ番号)、call_01... は偏光方向ごとの個別のラベルです。

大規模計算ではこれらのフォルダの数が多くなり、計算機のストレージの制限に抵触する場合があります。そのような場合には、 solver.config.remove_work_dir パラメータを true にして、計算が終了した作業フォルダを削除してください。

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、沢頭孟氏 (鳥取大学)、高草木達氏 (北海道大学) 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-XAFS 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-XAFS 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-XAFS 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)