ODAT-SE XAFS 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-XAFS ?

Polarization-dependent Total Reflection Fluorescence X-ray Absorption Fine Structure (PTRF-XAFS) is a method to analyze material structures by the X-ray absorption spectra that reveal symmetries or electronic states of atoms. Especially, by using the total reflection, it is efficient for the analysis of surface structure.

For the analysis of X-ray spectra, a first-principle calculation software, FEFF [1,2], has been developed that provides theoretical prediction of X-ray spectroscopy from the information of atomic positions. It is implemented by Fortran and runs on standard Linux platforms. odatse-XAFS is an adaptor library to use FEFF 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 FEFF.

[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 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 2DMAT / 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 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-XAFS

- v1.0.0: 2025-04-11

1.5 Main developers

odatse-XAFS, and FEFF solver module for 2DMAT have been developed by following members.

- odat-XAFS v1.0.0 -
  - Y. Motoyama (The Institute for Solid State Physics, The University of Tokyo)
  - K. Yoshimi (The Institute for Solid State Physics, The University of Tokyo)
  - T. Aoyama (The Institute for Solid State Physics, The University of Tokyo)
  - A. Nakano (National Institute for Fusion Science)
  - T. Hoshi (National Institute for Fusion Science)

CHAPTER

TWO

INSTALLATION OF ODATSE-XAFS

2.1 Prerequisites

- Python3 (>=3.9)
  - The following Python packages are required.
    - numpy >= 1.14
    - pydantic >= 2.0
  - ODAT-SE version 3.0 and later
  - FEFF 8.5light, or version 9 and later

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`).

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

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

2. Install FEFF
   - Download the source package from the distribution site, and compile the source files.
     - Obtain the source package and create a build directory:

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

     - For GCC (gfortran), compiler options are required as follows:

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

- For intel compiler (ifort), compiler executable is specified as follows:

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

If the compiliation is successful, an executable file `feff85L` will be generated. Put `feff85L` in a directory listed in the PATH environment variable, or specify the paths to these commands at run time.

3. Install odatse-XAFS

- From source files:

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

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

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

Then, the library of odatse-XAFS and the command `odatse-XAFS` wil be installed.

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 XAFS:

1. Use odatse-XAFS 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-XAFS 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-XAFS and ODAT-SE modules, type the following commands:

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

CHAPTER

THREE

TUTORIALS

`odatse-XAFS` is a direct problem solver module for the ODAT-SE framework to use FEFF developed by J. J. Rehr of University of Washington. FEFF evaluates theoretical prediction of X-ray spectroscopy data such as XAFS by the first-principle calculations using multiple scattering for given atomic positions and other parameters. Consider this to be a direct problem from atomic positions to X-ray spectrum, it turns to an inverse problem to find the atomic position from a given spectrum. ODAT-SE provides the following five algorithms to solve the inverse problem.

- `minsearch`

  Nelder-Mead method.

- `mapper_mpi`

  Searching the entire search grid for a given parameter.

- `bayes`

  Bayesian optimization.

- `exchange`

  Sampling by the replica exchange Monte Carlo method.

- `pamc`

  Sampling by the population annealing Monte Carlo method.

In this tutorial, we will first introduce how to run the direct problem solver `FEFF`. Then we will instruct how to run `mapper_mpi` for solving inverse problems. Hereinafter, we use `odatse-XAFS` program included in odatse-XAFS with input files in TOML format.

At the end of the tutorial, we will explain how to write your own main program for analyses.

3.1 XAFS Solver

In this section, we will explain how to install and test FEFF.

3.1.1 Download and Install

First, you need to obtain the source package of odatse-XAFS from the repository.

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

Next, you need to download the source files of FEFF from the repository, and build it. A setup script is provided for this proocess.

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

When it is successful, `feff85L` will be created in the current directory.

> **Note**
>
> If you use intel Fortran compiler, you need to edit setup.sh before running the script.

> **Note**
>
> In the abve setup script, a patch to FEFF is applied that suppresses output of several unused files to reduce the amount of temporal files.

3.1.2 Calculation execution

In this tutorial, we will actually do the calculation using FEFF. The sample input files are located in `sample/solver` of odatse-XAFS.

```
$ cd sample/solver
```

Next, copy `feff85L` to the current directory.

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

Execute `feff85L`. The input file is `feff.inp`. (The name of the input file is fixed.)

```
$ ./feff85L
```

Then, the following log messages will be displayed, and a number of output files are generated.

```
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
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
```

(continues on next page)

(continued from previous page)

```
   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
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 Visualization of calculation result

Among the output files, we will refer `chi.dat` for the spectrum data, whose contents are as follows:

```
# 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
     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

...
```

The lines starting with # are comments containing the information of calculation conditions and models. Then, the lines follow that contain the wave number k starting from the threshold ($k = 0$), $\chi(k)$, $|\chi(k)|$, and the phase. The figure shows $\chi(k)$ as a function of k.

3.2 Optimization by Grid search

In this section, we will explain how to perform a grid-type search to analyze atomic coordinates from spectrum data. The grid type search is compatible with MPI. The search grid is generated from the input parameters as an evenly spaced mesh.

3.2.1 Location of the sample files

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

- `mock_data.txt`, `template.txt`

  Reference file to proceed with calculations in the main program.

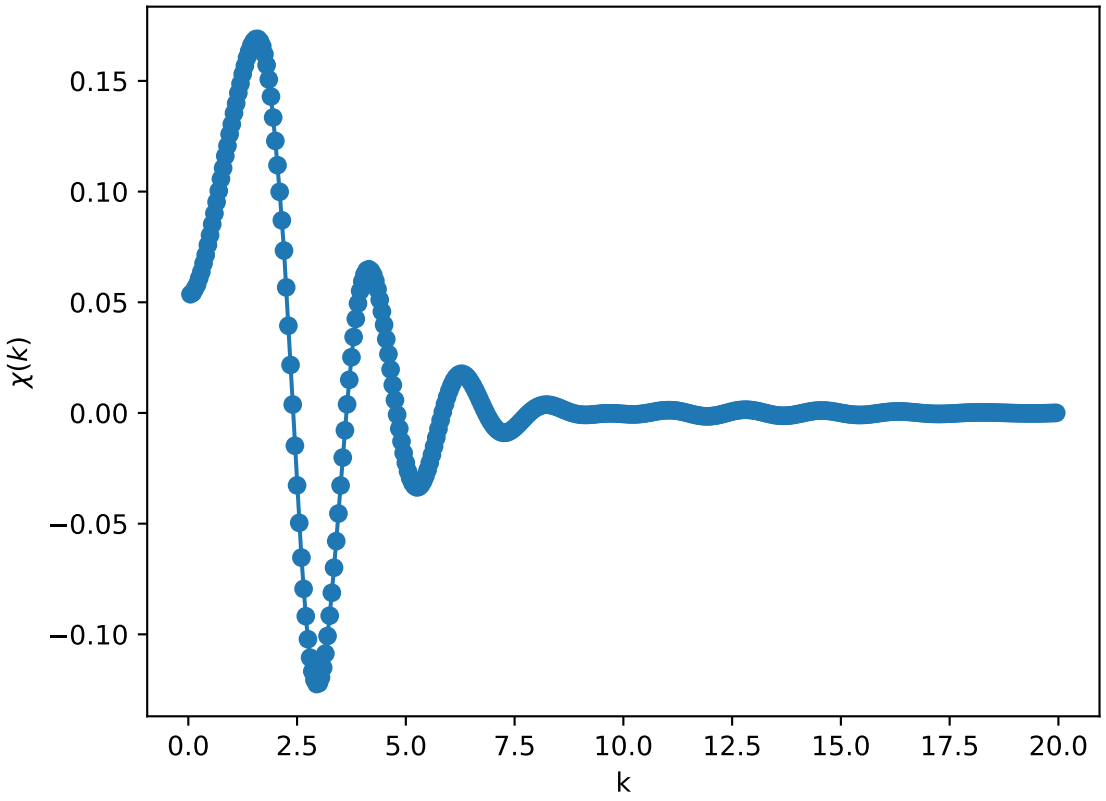

Fig. 3.1: An example of XAFS spectrum calculation using FEFF.

- `ref_ColorMap.txt`

  A file to check if the calculation was performed correctly (the answer to `ColorMap.txt` obtained by doing this tutorial).

- `input.toml`

  Input file of the main program.

- `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.

3.2.2 Reference files

`template.txt` is a template of the input file for FEFF. In this tutorial, to reduce the computational cost, we will perform the two-parameter search for the coordinates x, y of a sulfur atom, with z fixed to z = -1.60. The content of the file is shown below in which `@x` and `@y` correspond to the parameters to be varied.

The reference data that imitates experiments is stored in the file `mock_data.txt` that contains the spectrum data for three different directions of polarization.

```
* 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 ]
*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
```

(continues on next page)

(continued from previous page)

```
*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 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 = "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
```

(continues on next page)

(continued from previous page)

```
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]
```

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 `template.txt`.
- `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 `feff`.

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

`[solver.config]` section specifies options for `feff85L` called from the main program.

- `feff_exec_file` specifies the path to the FEFF executable.
- `feff_output_file` specifies the file among the output files of FEFF that contains the XAFS spectrum data.
- `remove_work_dir` specifies whether the work directory for the output of FEFF should be removed every after the calculation.
- `use_tmpdir` specifies whether the output files of FEFF should be written in /tmp.

`[solver.param]` section specifies options for the input file of FEFF.

- `string_list` is a list of variable names embedded in `template.txt`.
- `polarization_list` is a list of placeholders for the polarization vector embedded in `template.txt`.
- `polarization` is a list of polarization vectors.
- `calculated_first_k`, `calculated_last_k` are the lower and upper ends of the wave number for which the calculated values and the experimental data are to be compared.

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

- `path_epsilon` specifies the path where the experimental data is located.

`[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 `@x` and `@y`.

`[algorithm.param]` section specifies the options to the search algorithm.

- `min_list`, `max_list`, `num_list` are the range of search grid and the number of grid points.

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

3.2.4 Calculation execution

First, move to the folder where the sample files are located. (We assume that you are directly under the directory where you downloaded this software.)

```
$ cd sample/mapper
```

Copy `feff85L` to the current directory.

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

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

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

Here, the calculation using MPI parallel with 4 processes will be done. When executed, a folder for each rank will be created, and a subfolder `LogXXXX_YYYY` (where `XXXX` and `YYYY` are the grid id and the sequence number, respectively) will be created under it. The standard output will look like as follows.

```
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.
```

(continues on next page)

(continued from previous page)

```
↪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  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` and `@y` 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.

```
-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
...
```

The first and second columns contain the values of `@x` and `@y`, respectively, and the third column contains the `R-factor`.

Note that `do.sh` is available as a script for batch calculation. In `do.sh`, `res.txt` and `ref.txt` are also compared for the check. Here is what it does, without further explanation.

```
#!/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 Visualization of calculation results

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

```
$ python3 plot_colormap_2d.py -o ColorMapFig.png
```

Looking at the generated figure, we can see that it has the minimum value around (± 1.2, ± 0.8).

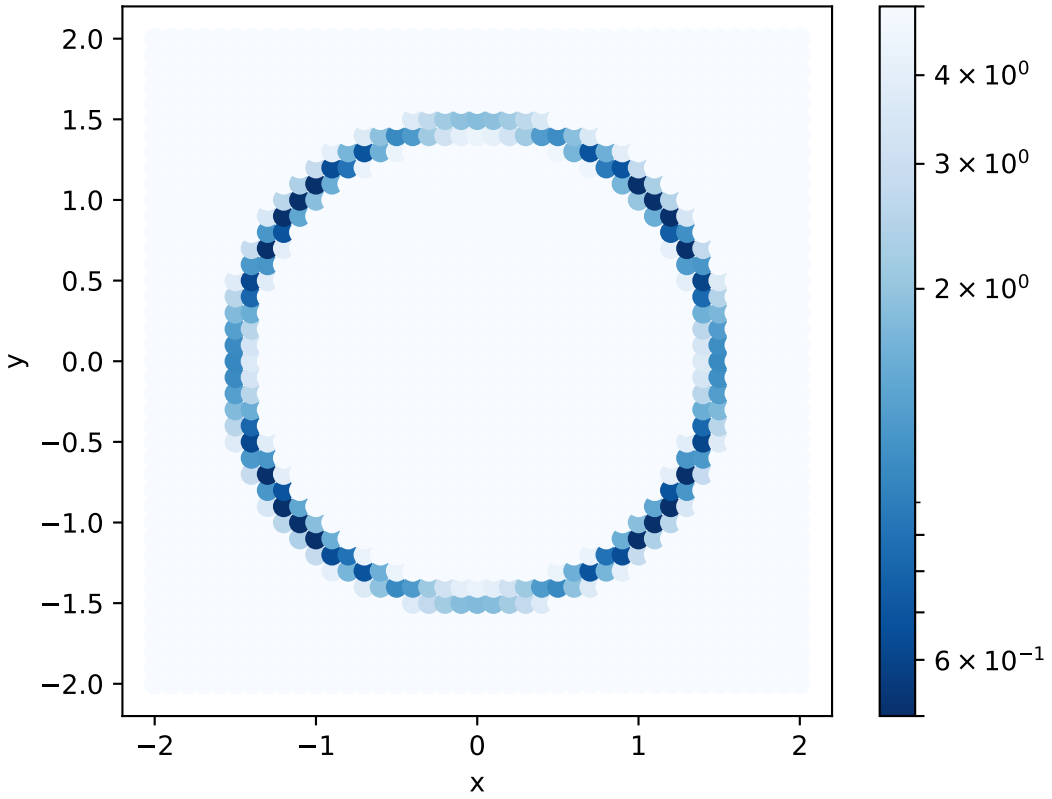

Fig. 3.2: Color map of R-factor with respect to x and y coordinates of S atom at z=-1.60.

3.3 Analyses by user programs

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

3.3.1 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.

- `mock_data.txt`, `template.txt`

  Reference file to proceed with calculations in the main program.

- `ref.txt`

  A file containing the answers you want to seek in this tutorial.

- `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.

3.3.2 Description of main program

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

```
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()
```

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.XAFS` 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.XAFS.Solver` class

  This class is for the direct problem solver of the odatse-XAFS 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.

3.3.3 Input files

The input file `input.toml` for the main program is almost the same as that used in the previous tutorial for grid search. In the `algorithm.param` section, the search region `min_list` and `max_list`, and the initial value `initial_list` are specified. `algorithm.name` parameter for specifying the algorithm type is ignored.

The template file and the reference experimental data are the same as those in the previous tutorials.

3.3.4 Calculation execution

First, move to the folder where the sample files are located. (We assume that you are directly under the directory where you downloaded this software.)

```
$ cd sample/user_program
```

Copy `feff85L` to the current directory.

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

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 look as follows.

```
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.
↪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 = 1.12000000
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 ␣
```

(continues on next page)

(continued from previous page)

```
↪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
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`, and `value_03` are the candidate parameters at each step, and `R-factor` is the function value at that point. The results at each step are also written in the folder `output/LogXXXX_YYYY` (where XXXX and YYYY are the step counts). The final estimated parameters will be written to `output/res.dat`. In the current case, the following result will be obtained:

```
fx = 0.20606977805890725
x_S = 1.125299780290939
y_S = 0.9597181918334485
z_S = -1.596967599355829
```

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-XAFS module is a `Solver` package that uses FEFF to calculate X-ray absorption spectra from the atomic position x and returns the deviation from the experimental XAFS spectra 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 dictionary 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 templates of the input file for FEFF. The output data are the output files and log files generated by `feff85L` of FEFF. Their contents will be shown in this section.

4.1 Input parameters

Input parameters can be specified in the subsections `config`, `param`, `reference` in `solver` section.

4.1.1 [solver] section

- `dimension`

  Format: integer

  Description: Number of parameters. It must be specified either in the solver section of in the base section. When both specified, the value in the solver section is used. The value should be equal to the length of `string_list`.

4.1.2 [solver.config] subsection

- `feff_exec_file`

  Format: string (default: "feff85L")

  Description: Path to FEFF solver.

- `feff_input_file`

  Format: string (default: "feff.inp")

  Description: Input file for FEFF solver. For `feff85L`, it is fixed to `feff.inp`.

- `feff_output_file`

  Format: string (default: "chi.dat")

  Description: Output file for X-ray absorption spectrum among FEFF output files. For `feff85L`, it is fixed to `chi.dat`.

- `feff_template_file`

  Format: string (default: "template.txt")

  Description: Template for the input file of FEFF.

- `remove_work_dir`

  Format: boolean (default: false)

  Description: If it is set to true, the work directories `Log%%%%_####` will be removed after the calculation.

- `use_tmpdir`

  Format: boolean (default: false)

  Description: If it is set to true, the output files of FEFF will be written in a temporary directory in /tmp (or in the directory specified by the environment variable `TMPDIR`). It is automatically removed after the calculation.

4.1.3 [solver.param] subsection

- `string_list`

  Format: list of strings. The length should match the value of dimension (default: ["value_01", "value_02"]).

  Description: List of placeholders to be used in the reference template file to create the input file for the solver. These strings will be replaced with the values of the parameters being searched for.

- `polarization_list`

  Format: list that consists of three strings.

  Description: List of placeholders for the polarization vector to be used in the reference template file.

- `polarization`

  Format: list of lists that consist of three floats.

  Description: List of polarization vectors.

- `calculated_first_k`

  Format: float

  Description: Lower end of the range of wave length in which the calculated values and the experimental data are compared.

- `calculated_last_k`

  Format: float

  Description: Upper end of the range of wave length in which the calculated values and the experimental data are compared.

- `k_range`

  Format: list of floats

  Description: The range of wave length specified by a list in the form [lower value, upper value] in which the calculated values and the experimental data are compared. This parameter and the pair of parameters `calculated_first_k` and `calculated_last_k` are exclusive, and either one should be specified.

4.1.4 [solver.reference] subsection

- path_epsilon

  Format: string

  Description: Path to the reference data file.

4.2 Reference files

4.2.1 Input template file

The input template file `template.txt` is a template for creating an input file for `feff85L`. The parameters to be varied in odatse-XAFS (such as the atomic coordinates you want to find) should be replaced with the appropriate string, such as `value_*`. The strings to be used are specified by `string_list` in the `[solver.param]` section of the input file for the solver. Similarly, the elements of the polarization vector are replaced by the strings such as `polarization_*` specified by the `polarization_list` parameter.

An example template is shown below.

```
* 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 ]
*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
```

(continues on next page)

(continued from previous page)

```
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
```

In this case, `value_01`, `value_02`, and `value_03` are the parameters to be varied, and `polarization_01`, `polarization_02`, and `polarization_03` are the elements of the polarization vector.

See the FEFF reference manual for the format of the input file.

4.2.2 Target file

The target file that contains experimental data is specified by the `path_epsilon` parameter in the `[solver.reference]` section. The format of the data is as follows: The first column contains the wave number, and the second and latter columns contain the spectrum intensity and its uncertainty for each polarization direction. The first two lines correspond to the header.

An example of the file is shown below.

```
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
...
```

4.3 Output files

For odatse-XAFS, the files generated by `feff85L` will be written in `call_01`, `call_02`, and `call_03` in `Log%%%%%_#####` created under the folder with the rank number. (When `use_tmpdir` is `True`, they are stored in `/tmp` or in a temporary directory specified by the environment variable `TMPDIR`.) `%%%%%` stands for the index of iteration in `Algorithm` (e.g., steps in Monte Carlo), `#####` stands for the index of group (e.g., replica index in Monte Carlo), and `call_01`, ..., are the labels for the polarization directions.

In large calculations, the number of files generated during the execution may

In large-scale calculations, the number of files generated during the execution may become huge and reach to the limitation of storage systems. For such cases, let the `solver.config.remove_work_dir` parameter be `true` in order to remove these folders.

CHAPTER

FIVE

ACKNOWLEDGEMENTS

The development of odatse-XAFS was supported by “Project for advancement of software usability in materials science” (FY2024) of The Institute for Solid State Physics, The University of Tokyo.

For the implementation of the forward problem solver, we thank T. Sawagashira (Tottori Univ.) and S. Takakusagi (Hokkaido Univ.).

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-XAFS 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 @)