2DMAT

on MateriApps LIVE!

Yuichi Motoyama (ISSP)

2024-12-02

for ODAT-SE a.k.a. 2DMAT ver 3.0.0

MateriApps LIVE!

- Tips
  - ユーザ名: user
  - パスワード: live
  - 端末は「左下のマーク」→「System Tools」→「LXTerminal」
  - 日本語キーボードを使っていて記号がおかしい場合
    - System Tools → Switch to Japanese Keyboard Layout
- その他困ったら
  - https://github.com/cmsi/MateriAppsLive/wiki/OnlineTutorial
  - および一番下にある setup.pdf を参照に
- 講習会向けUnix の使い方
  - https://gist.github.com/yomichi/0fb2cb7cbad641f77d762124b79af364

必要なファイルなどを入手する

- 全部自動でやるスクリプトを用意しています

```
$ cd   # ホームディレクトリへ移動
# スクリプトのダウンロード
$ wget https://bit.ly/install_2dmat_20241202
$ sh install_2dmat_20241202   # パスワードを要求されたら→ live
```

(先頭の $ は入力待ち状態を示す記号なので入力しなくても良いです)

(# 以降はコメントを示しているので、やはり入力しなくて大丈夫です)

(PDF からコピーペーストすると、必要な空白がなくなることがあります)

- このスクリプトがやること
  - 必要なDebian パッケージをインストール（これがパスワードを要求してきます)
  - odat-se のインストールおよびソースコード、サンプルのダウンロード
  - sim-trhepd-rheed, sxrdcalc をコンパイル
  - 実際に実行して簡便なテスト
    - sim-trhepd-rheed の方でFAILと出ますが、浮動小数点数の誤差の範囲で問題ないはずです（結果の比較まで行っている時点でインストールはうまく行っています)

ファイル構成

- ホーム直下の2DMATディレクトリ以下に入る

```
user@malive:~$ ls ~/2DMAT
ODAT-SE odatse-LEED odatse-STR odatse-SXRD  sim-trheapd-rheed
sxrdcalc
```

  - `ODAT-SE` が本講習会のメインターゲット
    - https://github.com/issp-center-dev/ODAT-SE
  - `sim-trhepd-rheed` は TRHEPD/RHEED 実験のシミュレータ
    - https://github.com/sim-trhepd-rheed/sim-trhepd-rheed
- `sxrdcalc` はSXRD 実験のシミュレータ
  - https://github.com/sxrdcalc/sxrdcalc
- `odatse-STR, -SXRD, -LEED` はそれぞれの実験シミュレータのラッパー
  - https://github.com/2DMAT/odatse-STR
  - https://github.com/2DMAT/odatse-SXRD
  - https://github.com/2DMAT/odatse-LEED

ファイル構成2

- 2DMAT ディレクトリ以下は次の通り

```
user@malive:~$ cd 2DMAT/ODAT-SE
user@malive:~/2DMAT/ODAT-SE$ ls
README.md  doc  pyproject.toml
sample  script  src  tests
```

- 重要なのは src と sample と script
  - src 以下がプログラム
    - src/odatse_main.py がメインプログラム
    - src/odatse がodatseパッケージ
- sample はチュートリアル用のサンプルファイル
- script は補助スクリプト

ファイル構成3

- ホームディレクトリ以下に、 .local という隠しディレクトリができている

```
user@malive:~/2DMAT/2DMAT$ cd ~/.local
user@malive:~/.local$ ls
bin  lib  share
```

- install_2dmat が pip install した odatse や physbo などが入っている
  - bin 以下が実行可能ファイル（プログラム・スクリプト）
    - odatseコマンドなどがここに入っている
  - lib 以下には python module が入っている
- デフォルトでは ~/.local/bin にPATH が通っていない点に注意
  - odatse を実行したい場合は ~/.local/bin/py2dmat とする
  - もしくはPATH を通してください（分かる方向け）

ファイル構成4

- sim-trhepd-rheed, sxrdcalc ともにプログラムがコンパイルされている
- sim-trhepd-rheed
  - sim-trhepd-rheed/src の下に実行ファイル bulk.exe と surf.exe がある

```
user@malive:~/2DMAT$ ls sim-trhepd-rheed/src/*.exe
sim-trhepd-rheed/src/bulk.exe     sim-trhepd-rheed/src/surf.exe
sim-trhepd-rheed/src/potcalc.exe  sim-trhepd-rheed/src/xyz.exe
```

(ファイル名の途中でtab キーを押すと補完してくれる)

(* はワイルドカード。この場合は .exe で終わるファイル全部にマッチ)

- sxrdcalc
  - sxrdcalc/ 直下に実行ファイル sxrdcalc がある

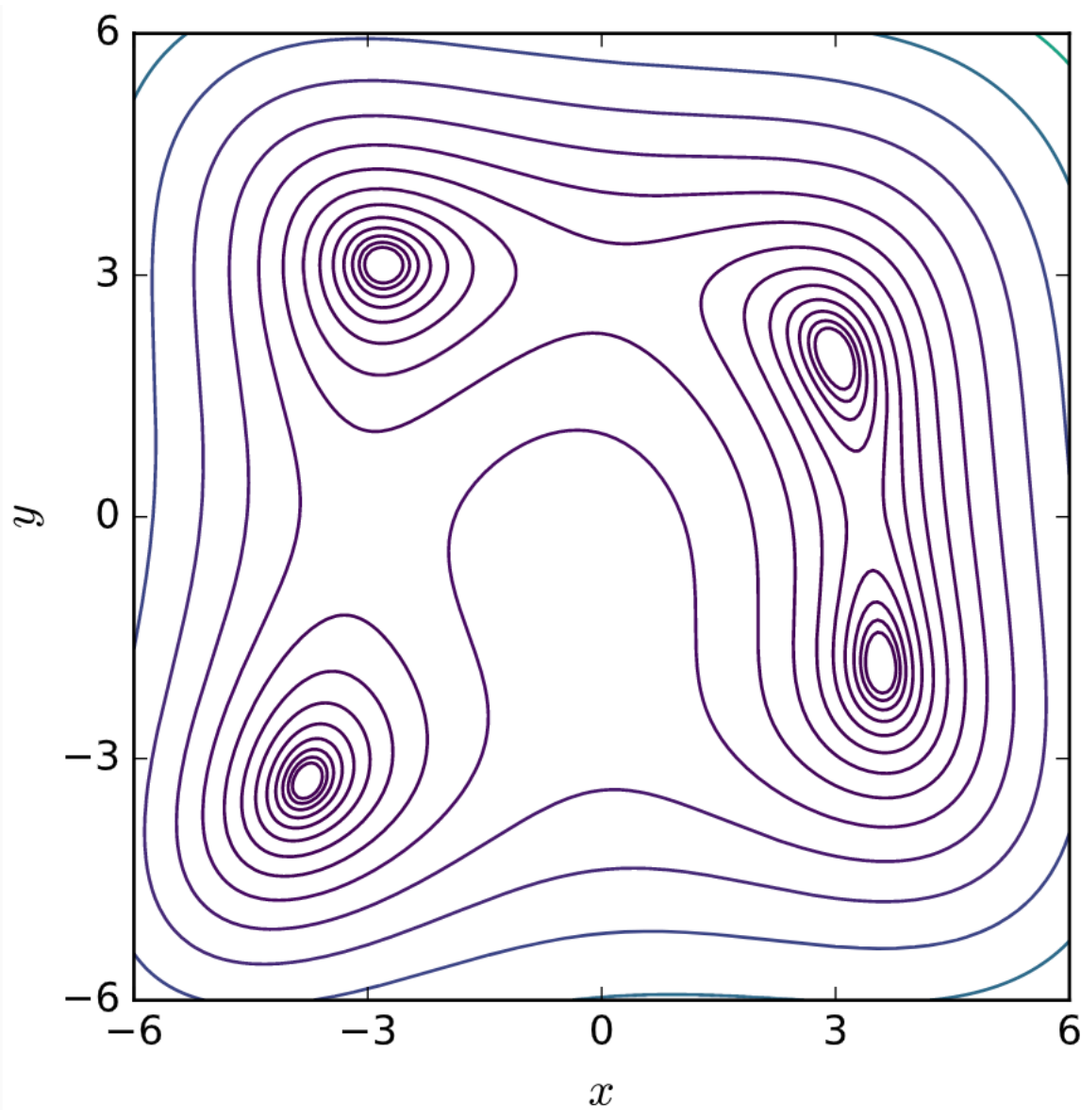
```
user@malive:~/2DMAT$ ls sxrdcalc/sxrdcalc
sxrdcalc/sxrdcalc
```

Himmelblau 関数を用いたチュートリアル

- 最適化問題（最小化問題）を odatse を用いて解析していく
- 目的関数 f(x) として Himmelblau 関数を用いる

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

  - 最小値0 を取る点が4つある
- `ODAT-SE/sample/analytical/*` に各種解析アルゴリズムごとの入力ファイルのサンプルが入っている
- `input.toml`: 入力ファイル
- `do.sh`: 解析し、結果を描画するスクリプト
  - `$ sh do.sh`



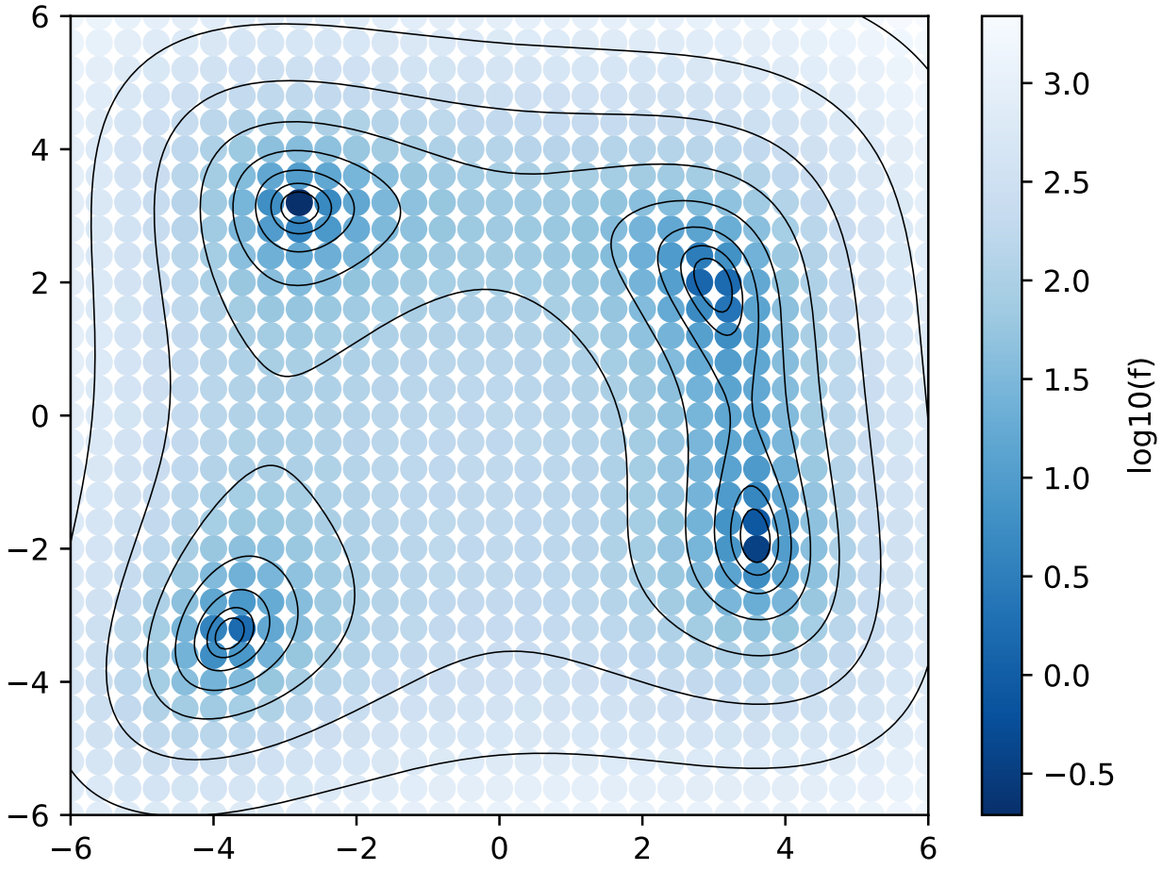
taken from wikipedia
(Himmelblau's function)

Himmelblau 関数を用いたチュートリアル

- do.sh: 解析し、結果を描画するスクリプト
  - output ディレクトリ以下に結果が保存される
  - PDFファイルは evince コマンドで見る
    - $ evince output/res.pdf
  - PNGファイルは eog コマンドで見る
    - $ eog output/res.png
  - MateriApps LIVE! には入っていないが、2dmat_installer.sh でインストールされている
- MPIプログラムの実行にはmpiexecコマンドを使っている
  - MateriApps LIVE! の場合はOpenMPI
  - 並列数(-npオプション)が使えるCPU数より多いと動かない
    - --oversubscribe オプションを追加してください
    - VirtualBox版は割当CPU数が1になっているので必要

グリッドサーチ (mapper)

output/ColorMap.txt
   => output/ColorMapFig.pdf



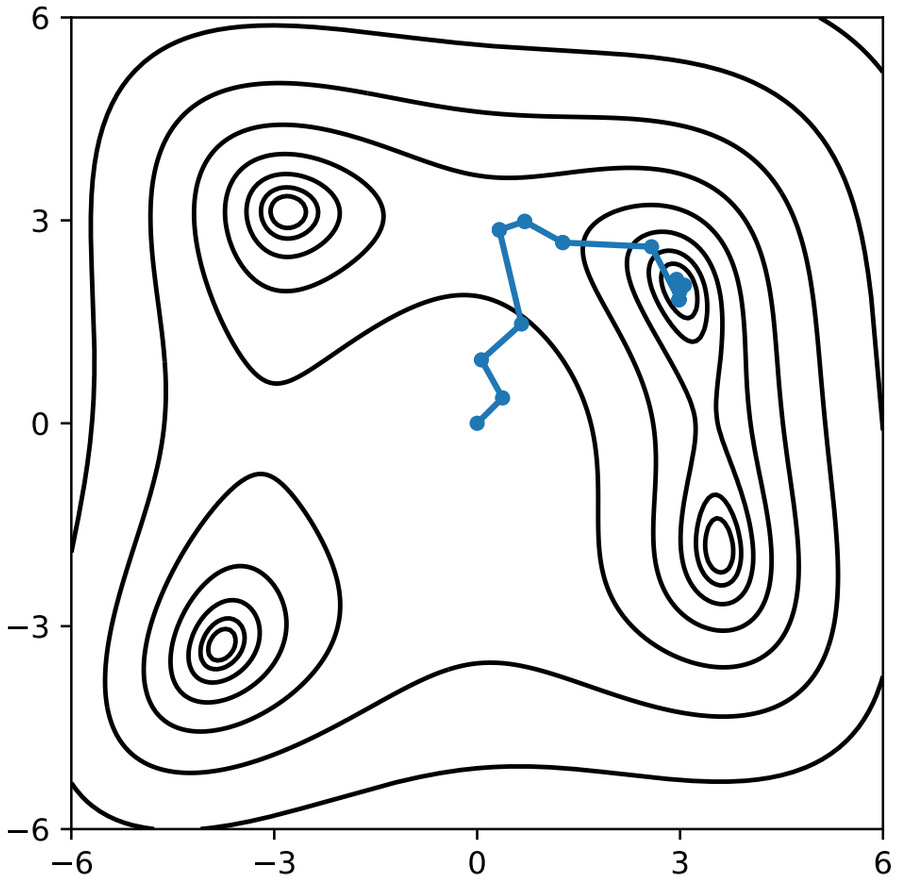
input.toml

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

[algorithm]
name = "mapper"            # グリッドサーチ

[algorithm.param]          # 探索空間
max_list = [6.0, 6.0]    # 上限
min_list = [-6.0, -6.0] # 下限
num_list = [31, 31]      # 刻み数

[solver]
name = "analytical"    # ベンチマーク関数を使う
function_name = "himmelblau" # 関数名

[runner]  # 今回の講習会では気にしなくて良いです
[runner.log]  # Solver の呼び出し間隔のログ
interval = 20
```

Nelder-Mead (minsearch)

```
$ cd ~/2DMAT/2DMAT/sample/analytical/minsearch
$ sh ./do.sh
$ evince output/res.pdf
```

output/SimplexData.txt
=> output/res.pdf



input.toml

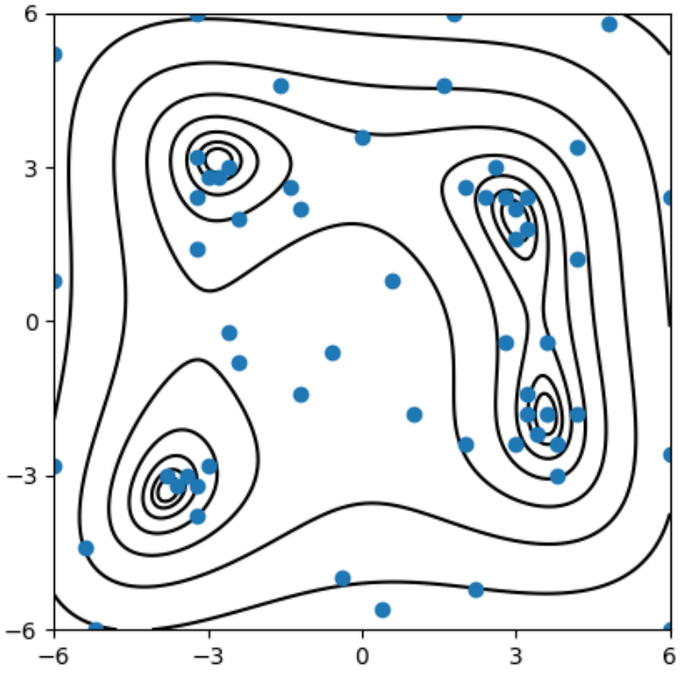
```
[base]
dimension = 2
output_dir = "output"

[algorithm]
name = "minsearch"  # Nelder-Mead
seed = 12345

[algorithm.param]          # 探索空間
max_list = [6.0, 6.0]    # 上限
min_list = [-6.0, -6.0] # 下限
initial_list = [0, 0]    # 初期値

[solver]
name = "analytical"    # ベンチマーク関数を使う
function_name = "himmelblau" # 関数名
```

ベイズ最適化 (bayes)

output/BayesData.txt

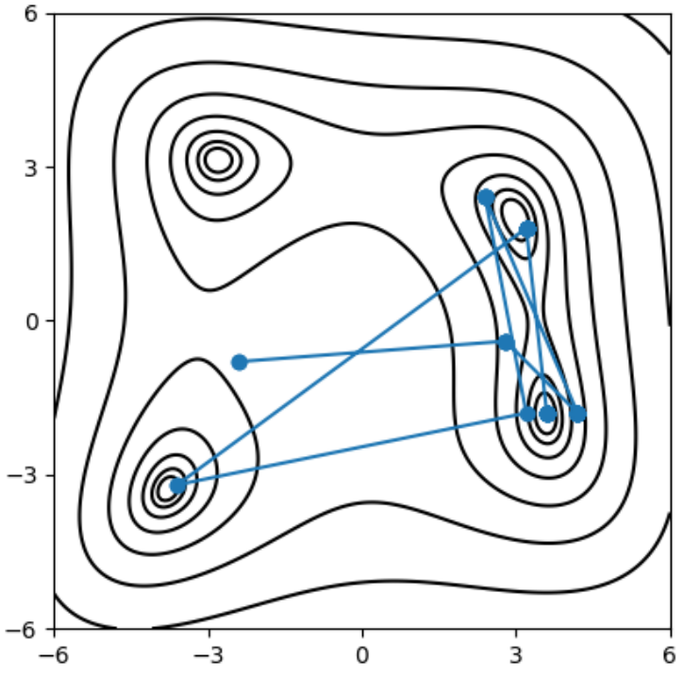
探索した点

output/actions.png



最小値の変遷

output/res.png



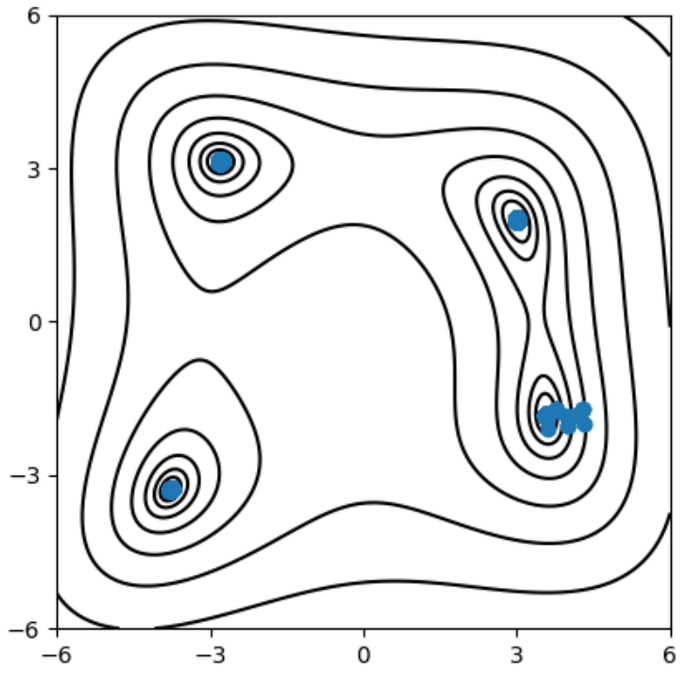
input.toml

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

    [algorithm]
    name = "bayes"
    seed = 12345

    [algorithm.param]
    max_list = [6.0, 6.0]
    min_list = [-6.0, -6.0]
    num_list = [61, 61]

    [algorithm.bayes]
    # 初期ランダムデータの数
    random_max_num_probes = 20
    # ベイズ最適化で探すデータの数
    bayes_max_num_probes = 40

    [solver]
    name = "analytical"
    function_name = "himmelblau"
```

交換モンテカルロ (exchange)

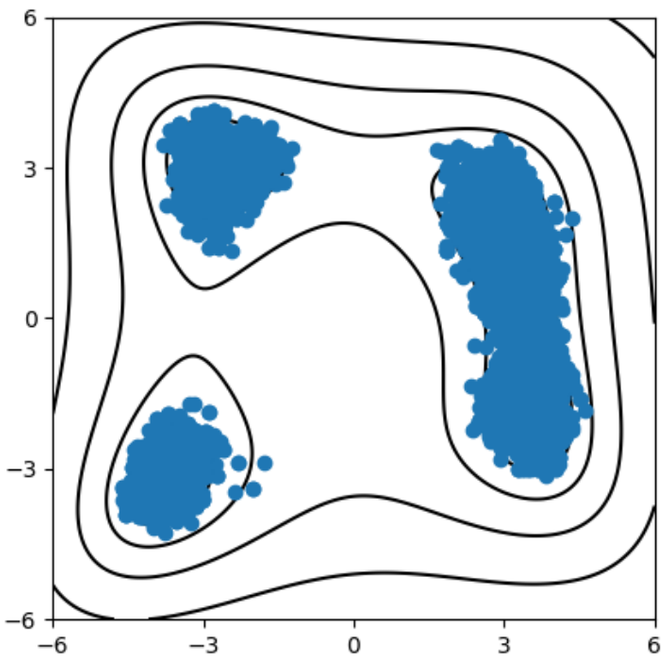
output/result_T0.txt
  => output/res_T0.png

T=0.01



output/result_T60.txt
  => output/res_T60.png

T=10.91...



最初の20点は除外してある（初期緩和）

input.toml

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

[algorithm]
name = "exchange"
seed = 12345

[algorithm.param]
max_list = [6.0, 6.0]
min_list = [-6.0, -6.0]
step_list = [0.3, 0.3]  # パラメータを動かすときのガウシアンの幅

[algorithm.exchange]
T_min =   0.01      # 温度の下限
T_max = 100.0       # 温度の上限
nreplica_per_proc = 20 #プロセスあたりのレプリカ数
numsteps = 10000    # 生成するサンプル数
numsteps_exchange = 100 # 交換間隔
# 100点生成するたびに温度交換を試みる
[solver]
name = "analytical"
function_name = "himmelblau"
```

交換モンテカルロ

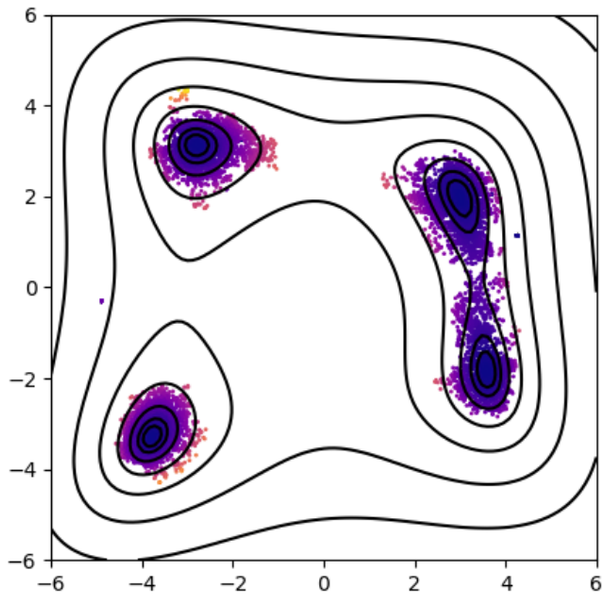
- 交換モンテカルロ法では、レプリカの数だけ温度点が計算される
  - 今回は 4 MPIprocess * 20 replicas/MPIprocess = 80 replicas
  - デフォルトでは対数空間で等間隔に刻まれる
- 各レプリカは交換操作以外は独立・並列に動作
- 本実装では、交換操作でレプリカの温度が変わる
- 最終的に、各レプリカごとにサンプリング系列をファイルに保存する
  - `[MPIRANK]/result.txt`
- 各温度でのサンプリングについては、`output` 以下に `result_T*.txt`が生成される
  - ファイルの先頭に温度が書かれている

```
user@malive:~/2DMAT/sample/analytical/exchange$ head -n3 output/result_T0.txt
# T = 0.1
0 0 170.0 0.0 0.0
1 0 167.3501513326404 -0.06141229784541388 0.14368300141726448
```

ポピュレーションアニーリング (pamc)

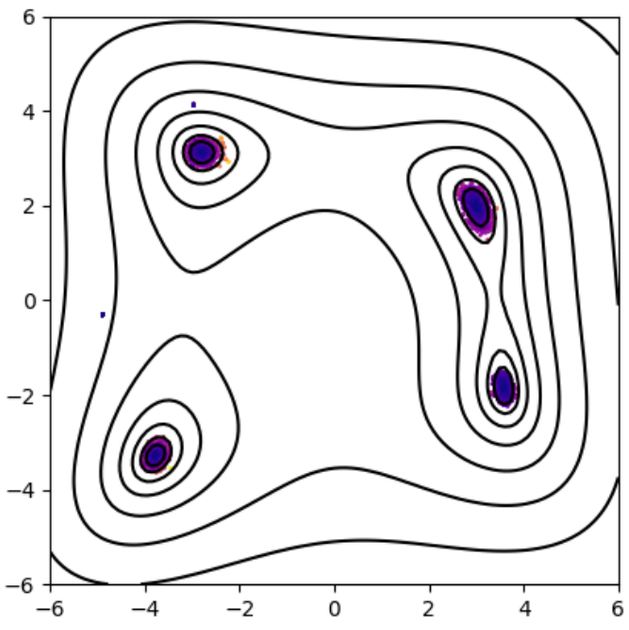
```
output/0/result_T10.txt
  => output/res_T10.png
```

T=10



```
output/0/result_T20.txt
  => output/res_T20.png
```

T=1

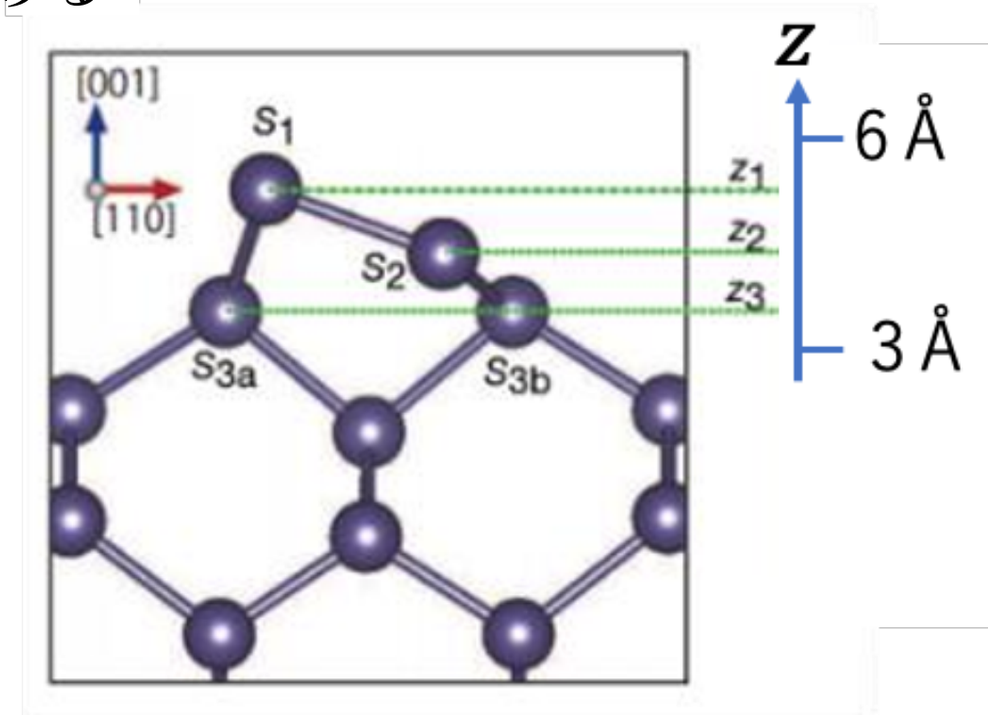


input.toml

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

[algorithm]
name = "pamc"
seed = 12345

[algorithm.param]
max_list = [6.0, 6.0]
min_list = [-6.0, -6.0]
step_list = [0.1, 0.1]

[algorithm.pamc]
Tmin = 1.0
Tmax = 100.0
Tnum = 21
Tlogspace = true
numsteps_annealing = 100 #温度降下間のステップ数
nreplica_per_proc = 100 #プロセス毎のレプリカ数

[solver]
name = "analytical"
function_name = "himmelblau"
```

TRHEPD 実験の解析（背景）

- sim-trhepd-rheed を用いた TRHEPD 実験解析のチュートリアルは
  - `sample/sim-trhepd-rheed/` 以下にある
- Ge(001)-c4x2 表面の解析を題材としている
  - sim-trhepd-rheed で計算した rocking curve を実験データとみなしている
  - s1, s2 (, s3ab) 原子のz 座標 (z1, z2, z3=z3a=z3b) を探索パラメータとする
  - s3ab を動かすのは minsearch のみ（簡単のため）
  - z1 と z2 は等価で、交換に関する対称性がある



TRHEPD 実験の解析（共通箇所）

- ~/2DMAT/sim-trhepd-rheed/src 以下の bulk.exe と surf.exe をそれぞれのディレクトリにコピーしてくる必要がある
  - $ cd ~/2DMAT/odatse-STR/sample/single_beam/bayes
  - $ cp ~/2DMAT/sim-trhepd-rheed/src/*.exe ./
  - （もちろんシンボリックリンクでも良い）
  - （PATH が通っている場所にコピーするのでも良い）
- py2dmat を実行する前に bulk.exe を実行して bulkP.b を作成する
  - $ ./bulk.exe
- do.sh はbulk.exe と py2dmat を順番に実行するスクリプト
  - 実行結果を予め用意されている参照ファイルと比較するテストも行う
- bulk.txt や surf.txt の詳細はsim-trhepd-rheed のドキュメントを参照のこと
  - https://github.com/sim-trhepd-rheed/sim-trhepd-rheed の doc にある

py2dmat から sim-trhepd-rheed を利用する

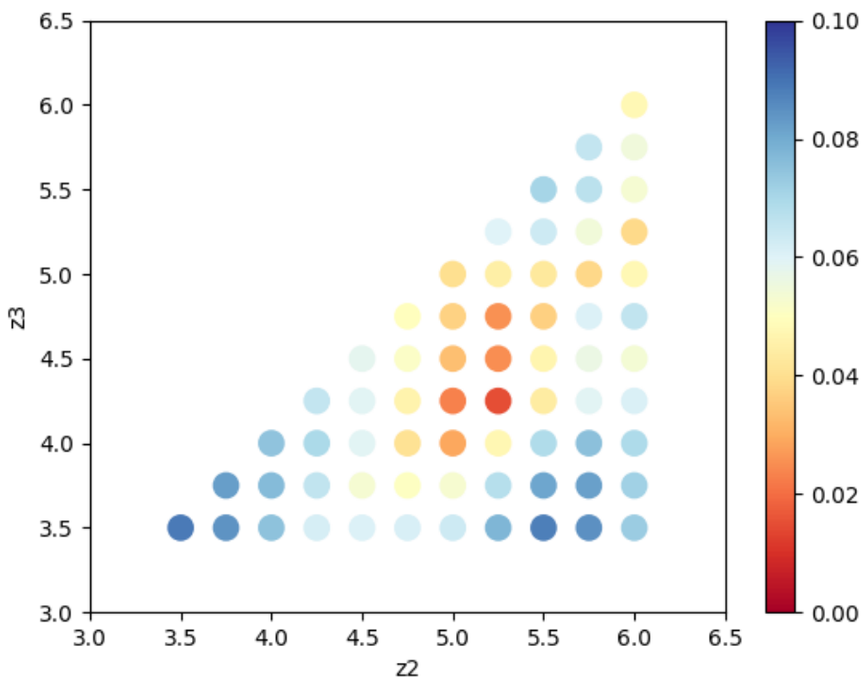
```
[base]
dimension = 2                  # 探索空間の次元 = 動かすパラメータの数

[solver]
name = "sim-trhepd-rheed"  # 順問題ソルバーとして sim-trhepd-rheed を使う

[solver.config]
cal_number = [1]               # D(x) の値が入っている列番号

[solver.param]
string_list = ["value_01", "value_02" ]  # テンプレートファイルのどの文字列を置き換えるか

[solver.reference]
path = "experiment.txt"     # 実験データ Dexp が書いてあるファイル
exp_number = [1]               # D(x) の値が入っている列番号
```

青字部分は、一度適当な入力で surf.exe を実行して、
出てきたファイルから必要な情報を調べる

mapper

実行方法および出力

```
$ cd ~/2DMAT/odatse-STR/sample/single_beam/mapper
$ cp ~/2DMAT/sim-trhepd-rheed/src/*.exe ./
$ sh ./do.sh
    ... many outputs ...
[35.    5.25  4.25]

$ python3 ./plot_colormap_2d.py
$ firefox ColorMapFig.png
```



input.toml のalgorithm 部分

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

MeshData.txt として

候補点の集合を定義可能

```
1 6.000000 6.000000
2 6.000000 5.750000
3 6.000000 5.500000
```

今回はz1 >= z2 に空間を制限している

minsearch

実行方法および出力

```
$ cd ~/2DMAT/odatse-STR/sample/single_beam/minsearch
$ cp ~/2DMAT/sim-trhepd-rheed/src/*.exe .
$ sh ./do.sh
   ... many outputs ...
Solution:
z1 = 5.230524973874179
z2 = 4.370622919269477
z3 = 3.596144450108164 7
```

input.toml のalgorithm 部分

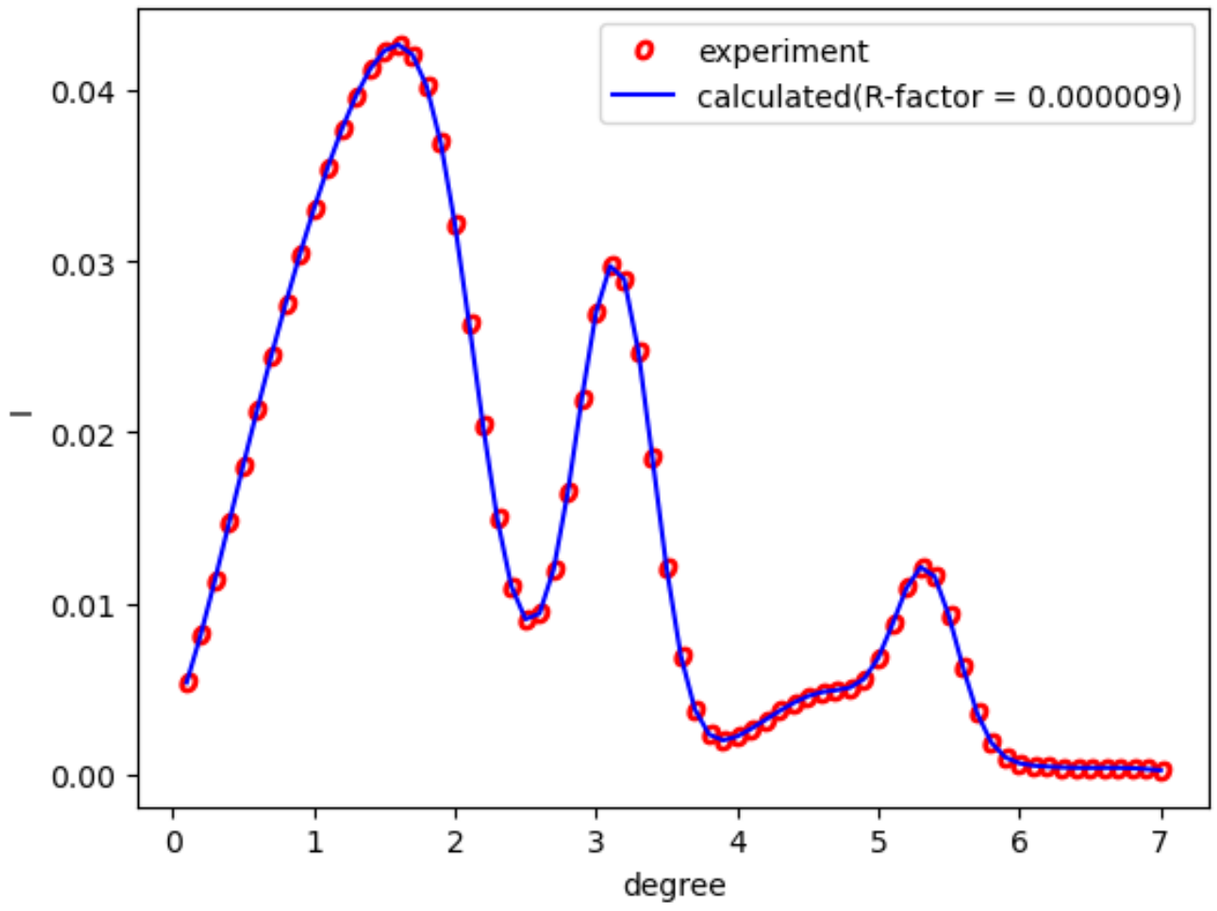
```
[algorithm]
name = "minsearch"
label_list = ["z1", "z2", "z3"]
[algorithm.param]
min_list = [0.0, 0.0, 0.0]
max_list = [10.0, 10.0, 10.0]
initial_list = [5.25, 4.25, 3.50]
```

rocking curve

- output/<MPIRANK>/Log数字_数字 に surf.exe の入出力が保存されている
  - 最初の数字は評価順、後ろの数字は系列（アルゴリズム依存）
  - minsearch は2つの系列がある（後ろの数字が0 or 1）
    - 1 はこれまでの最適解の系列
      - こちらの系列で一番最後のものが全体の最適解
    - SimplexData.txt の各ステップに対応
- Log*/RockingCurve_calculated.txt に計算で得られた rocking curve が記録されている
  - script/draw_RC_single.py をつかうと可視化できる
    - カレントディレクトリ以下にある RockingCurve.txtとexperiment.txtをプロットする

rocking curve

```
$ ls -d output/0/Log*1 | tail -n1
output/0/Log00000177_00000001
$ RESDIR=$(ls -d output/0/Log*1 | tail -n1)
$ cp ${RESDIR}/RockingCurve_calculated.txt ./RockingCurve.txt
$ python3 ~/2DMAT/odatse-STR/script/draw_RC_single.py
$ eog RC_single.png
```



実験データ（人工的なもの）をよく再現できた

exchange

実行方法および出力

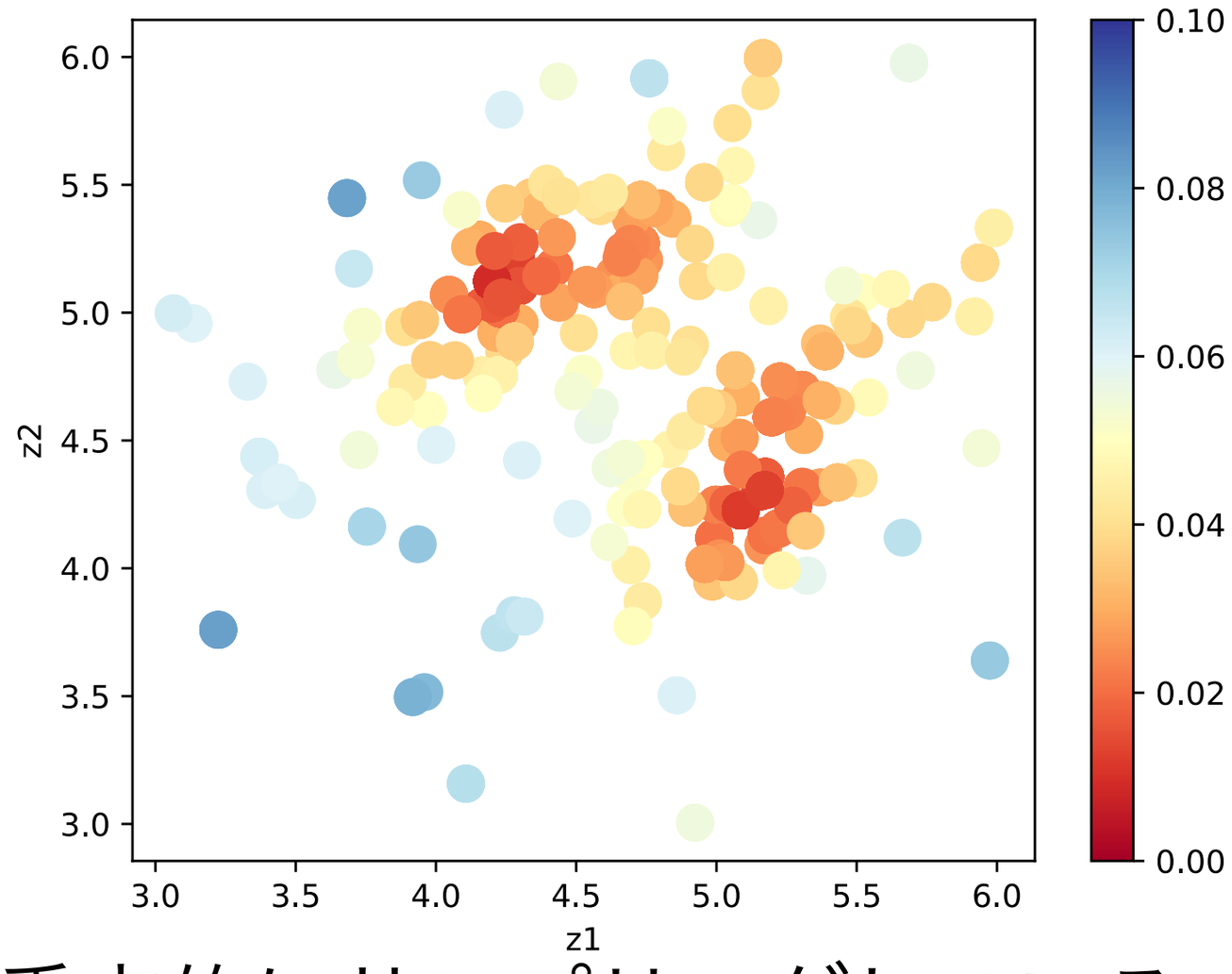
```
$ cd ~/2DMAT/odatse-STR/sample/single_beam/exchange
$ cp ~/2DMAT/sim-trhepd-rheed/src/*.exe .
$ sh ./do.sh
   ... many outputs ...
Best Result:
  rank = 1
  step = 282
  walker = 0
  fx = 0.008415217647192712
  z1 = 5.164773671165013
  z2 = 4.226467514644945
```

input.toml のalgorithm 部分

```
[algorithm]
name = "exchange"
label_list = ["z1", "z2"]
seed = 12345

[algorithm.param]
min_list = [3.0, 3.0]
max_list = [6.0, 6.0]

[algorithm.exchange]
numsteps = 1000
numsteps_exchange = 20
Tmin = 0.005
Tmax = 0.05
Tlogspace = true
```

exchange(2) 描画

```
$ cd output
$ python3 ../plot_result_2d.py   # 作図（下から2番めの温度）
$ evince result.pdf
```



解 (5.2, 4.3)付近を重点的にサンプリングしている

（※この系にはz1,z2の対称性があることに注意）

bayes

実行方法および出力

```
$ cd ~/2DMAT/odatse-STR/sample/single_beam/bayes
$ cp ~/2DMAT/sim-trhepd-rheed/src/*.exe ./
$ sh ./do.sh
   ... many outputs ...
0030-th step: f(x) = -0.020246 (action=179)
   current best f(x) = -0.010217 (best action=143)

end of run
Best Solution:
z1 = 5.1
z2 = 4.2
```

MeshData.txt として
候補点の集合を定義可能

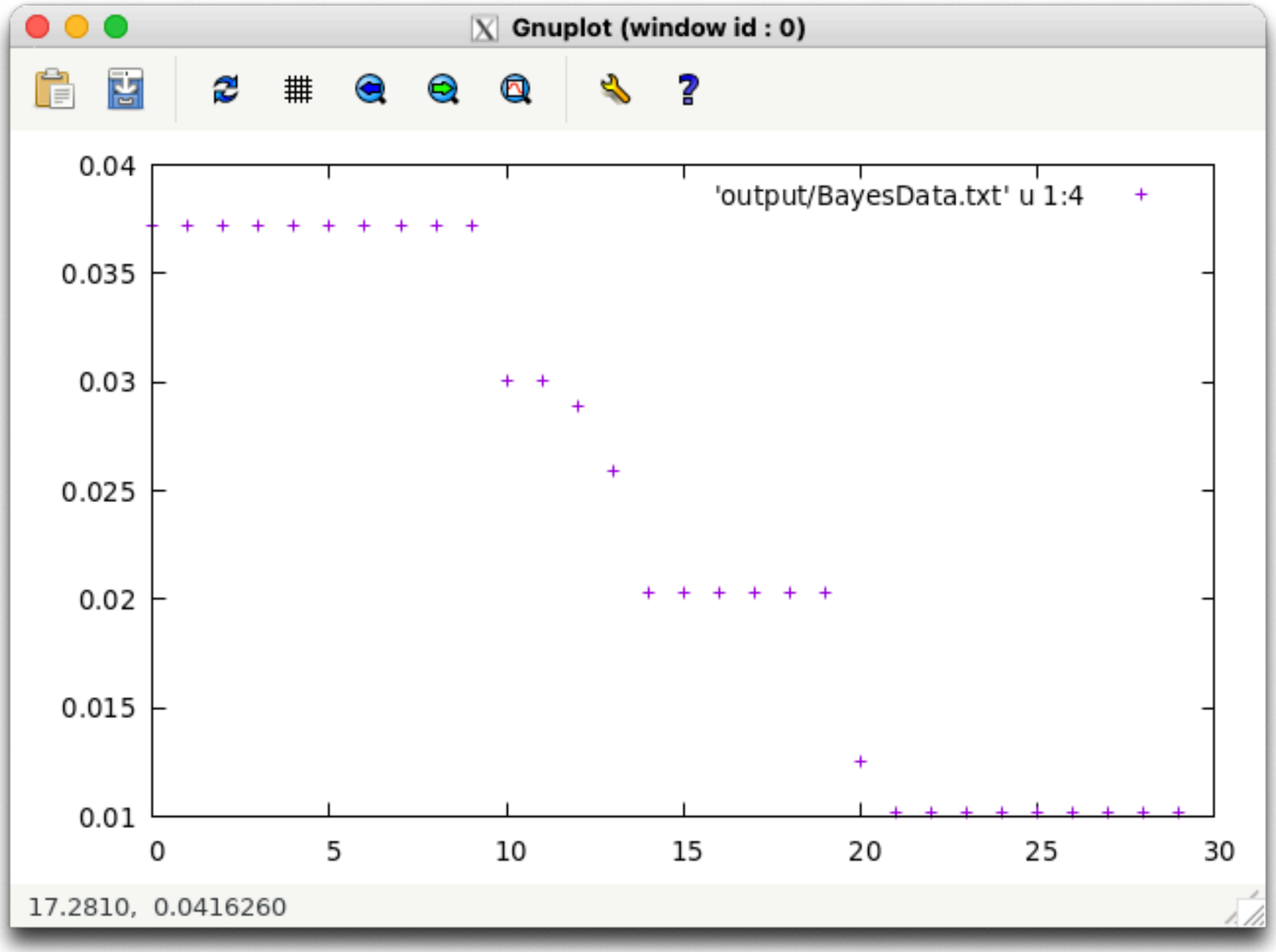
```
1 3.5 3.5
2 3.6 3.5
3 3.6 3.6
```

input.toml のalgorithm 部分

```
[algorithm]
name = "bayes"
label_list = ["z1", "z2"]
seed = 1

[algorithm.param]
mesh_path = "./MeshData.txt"

[algorithm.bayes]
random_max_num_probes = 10
bayes_max_num_probes = 20
```

bayes (2) 描画

```
$ gnuplot
gnuplot> pl 'output/BayesData.txt' u 1:4
```



10+10 回程度でかなり良い結果を得られた

（候補は351点あった）