

2DMAT

on



Yuichi Motoyama (ISSP)

2021-04-20
for 2DMAT ver 1.0

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る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

必要なファイルなどを入手する

- サンプルファイルやスクリプトをGitHub からダウンロードする

```
$ cd # ホームディレクトリへ移動
$ wget https://www.dropbox.com/s/i3em792nvg7h0vc/2dmat_install.sh
$ sh 2dmat_install.sh
```

(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なので、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まず確実に空白がなくなってしまうことに注意)

- 事前準備としてすでに実行済みだった場合

- 2DMAT のサンプルファイルが追加されているので更新が必要
- 簡単なやり方は 2DMAT ディレクトリを削除してやり直すこと
 - \$ rm -rf 2DMAT
 - \$ sh 2dmat_install.sh
- (もちろん、git pull などが分かる方はそれでも大丈夫です)

ファイル構成

- ホームディレクトリは次のようになっている

```
user@malive:~$ ls
2DMAT          Documents     Pictures      Templates
2dmat_install.sh Downloads     Public        Videos
Desktop        Music         sim-trhepd-rheed
```

- 重要なのは 2DMAT と sim-trhepd-rheed
 - 2DMAT が本講習会のメインターゲット
 - sim-trhepd-rheed は TRHEPD/RHEED 実験のシミュレータ
- sim-trhepd-rheed/src の下に実行ファイル bulk.exe と surf.exe がある

```
user@malive:~$ 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)

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
user@malive:~/2DMAT$ ls
doc      pyproject.toml  sample  src      tool
misc    README.md       script  tests
```
- 重要なのは src と sample と script
 - src 以下がプログラム
 - src/py2dmat_main.py がメインプログラム
 - 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ある

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

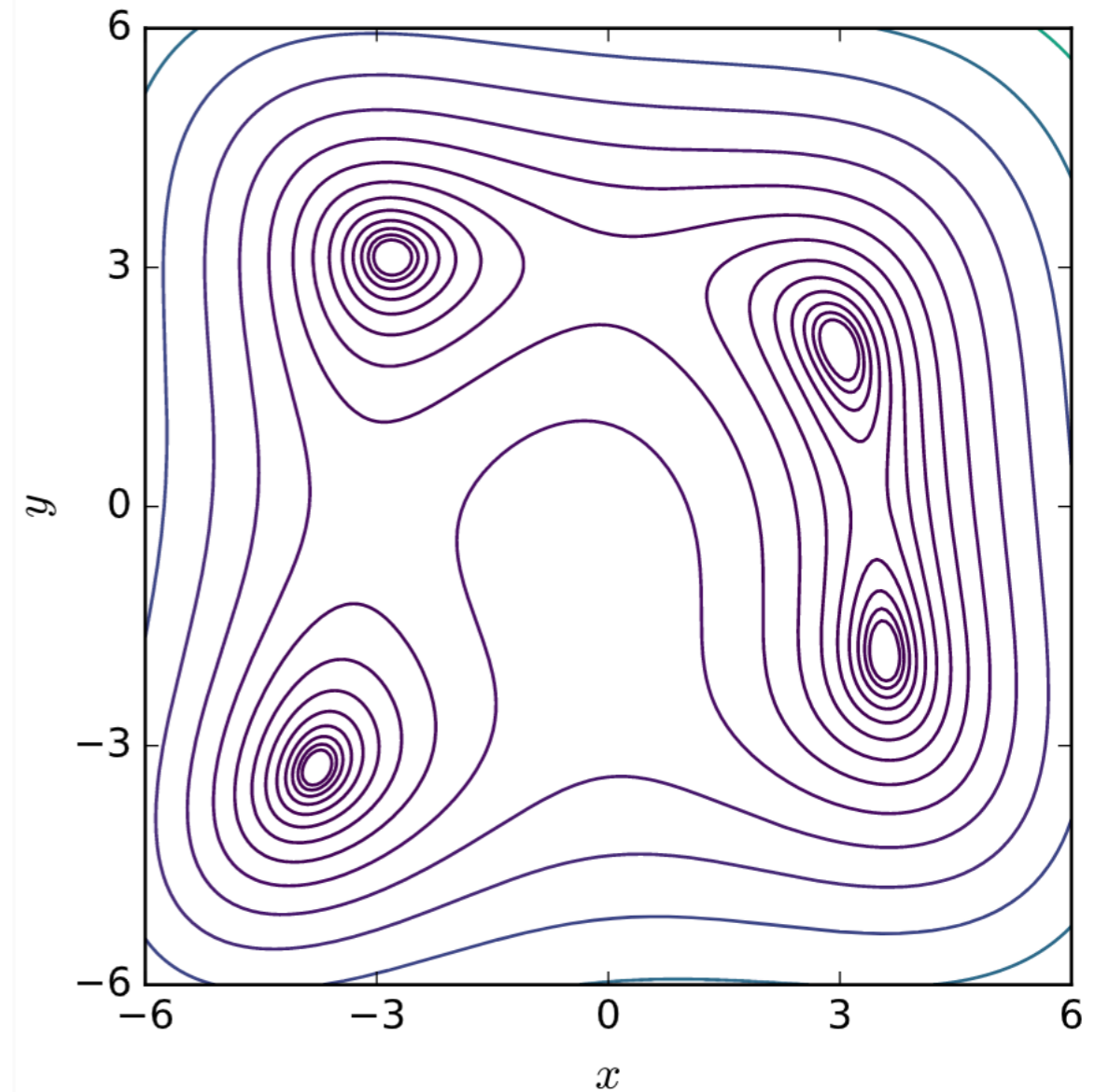
- `2dmat_install.sh` で `pip install` した `py2dmat` や `physbo` などが入っている
 - `bin` 以下が実行可能ファイル (プログラム・スクリプト)
 - `py2dmat` がここに入っている
 - `lib` 以下には `python module` が入っている
- デフォルトでは `~/.local/bin` に `PATH` が通っていない点に注意
 - `py2dmat` を実行したい場合は `~/.local/bin/py2dmat` とする
 - もしくは `PATH` を通してください (分かる方向け)

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 py2dmat 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
- `sample/py2dmat/*-himmelblau` 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
- `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` で見る
- png ファイルは `firefox` で見る

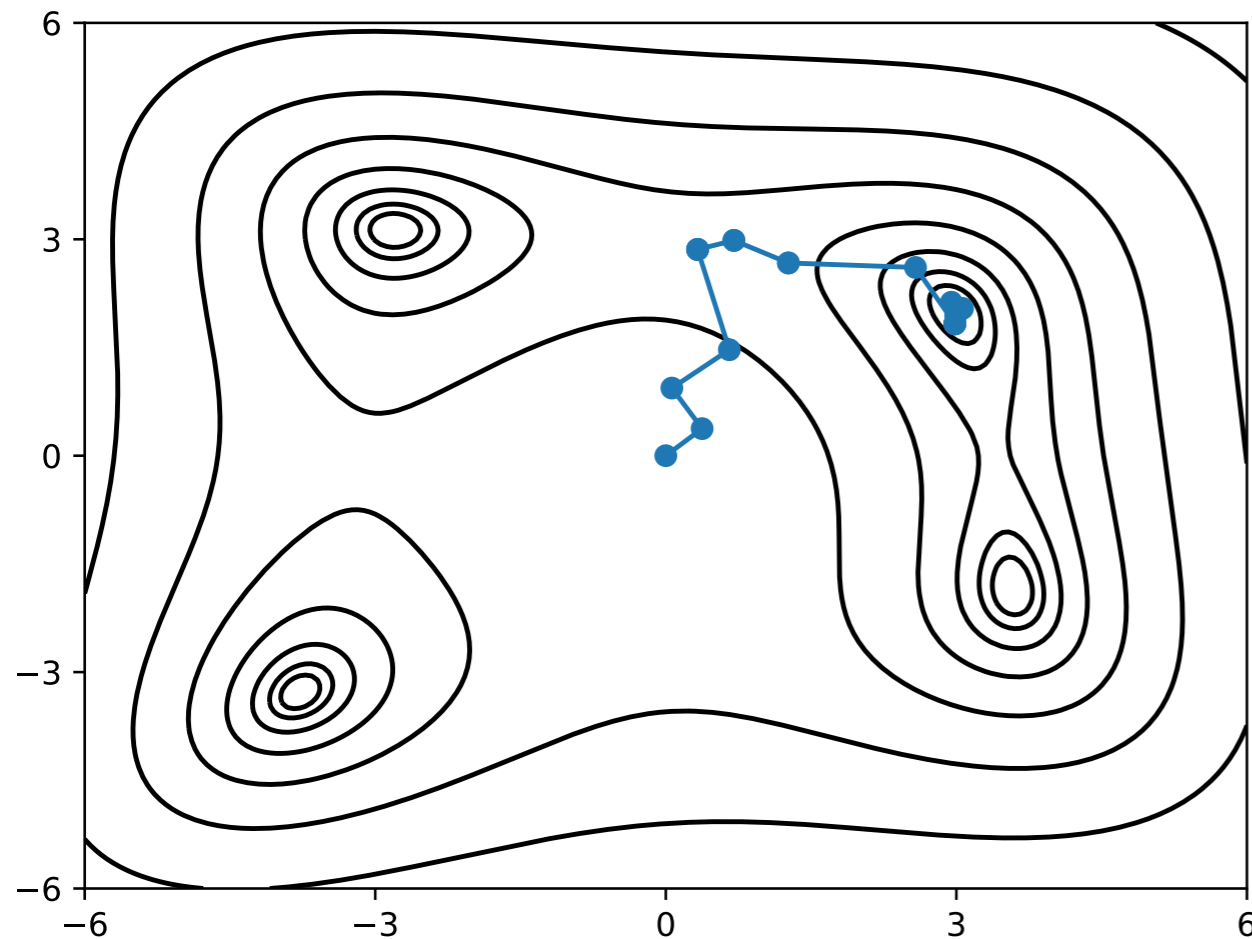


taken from wikipedia
(Himmelblau's function)

Nelder-Mead (minsearch)

```
$ cd ~/2DMAT/sample/py2dmat/minsearch-himmelblau  
$ sh ./do.sh  
$ evince output/res.pdf
```

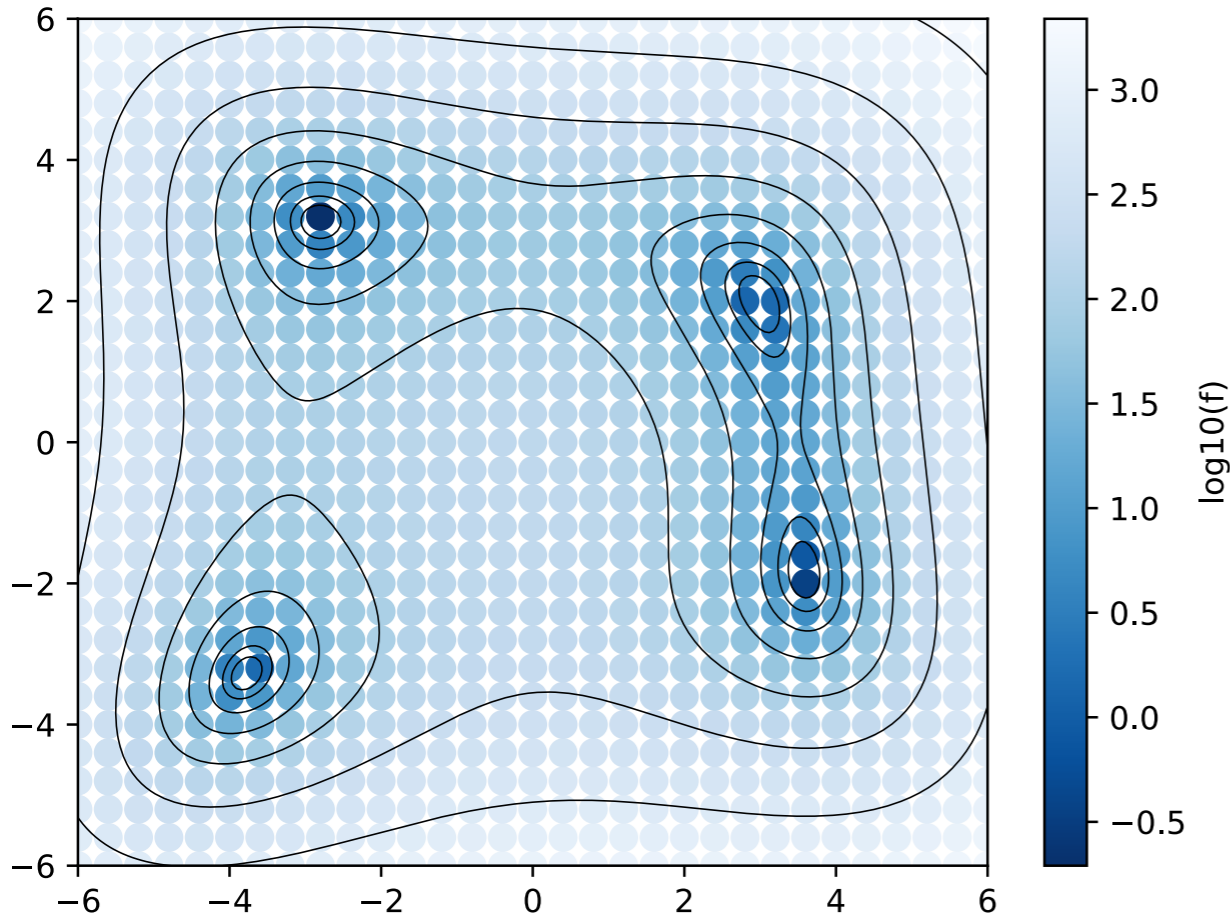
```
output/SimplexData.txt  
=> output/res.pdf
```



```
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
```


グリッドサーチ (mapper)

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



input.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

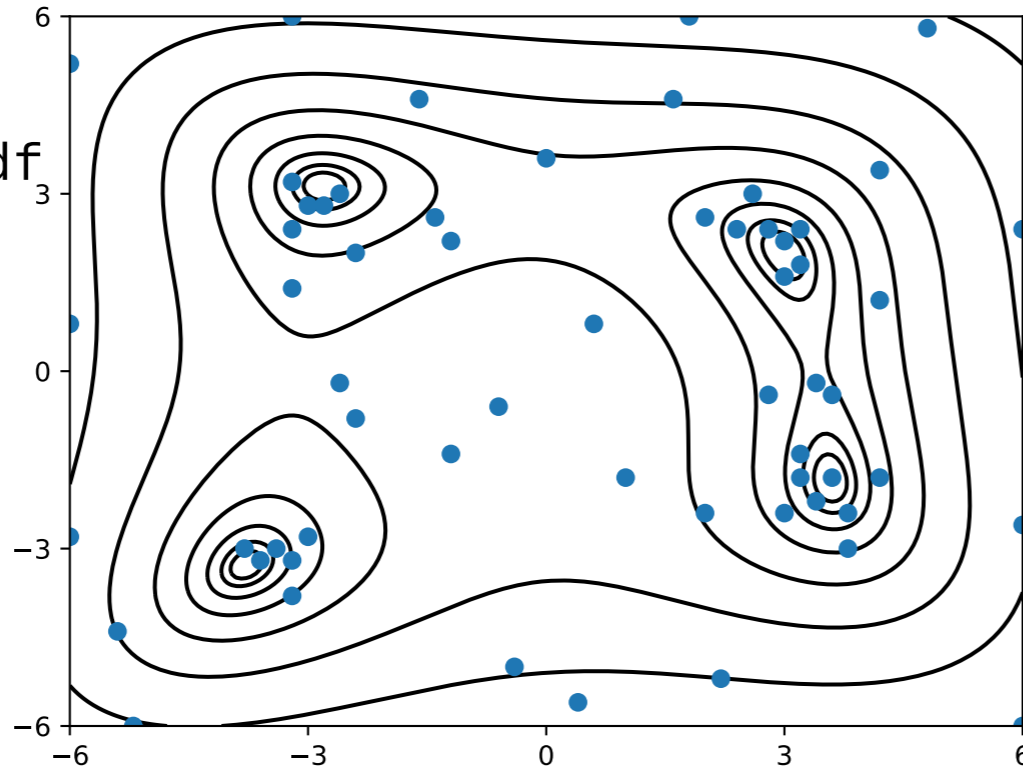
[solver]
name = "analytical" # ベンチマーク関数を使う
function_name = "himmelblau" # 関数名
```

ベイズ最適化 (bayes)

output/BayesData.txt

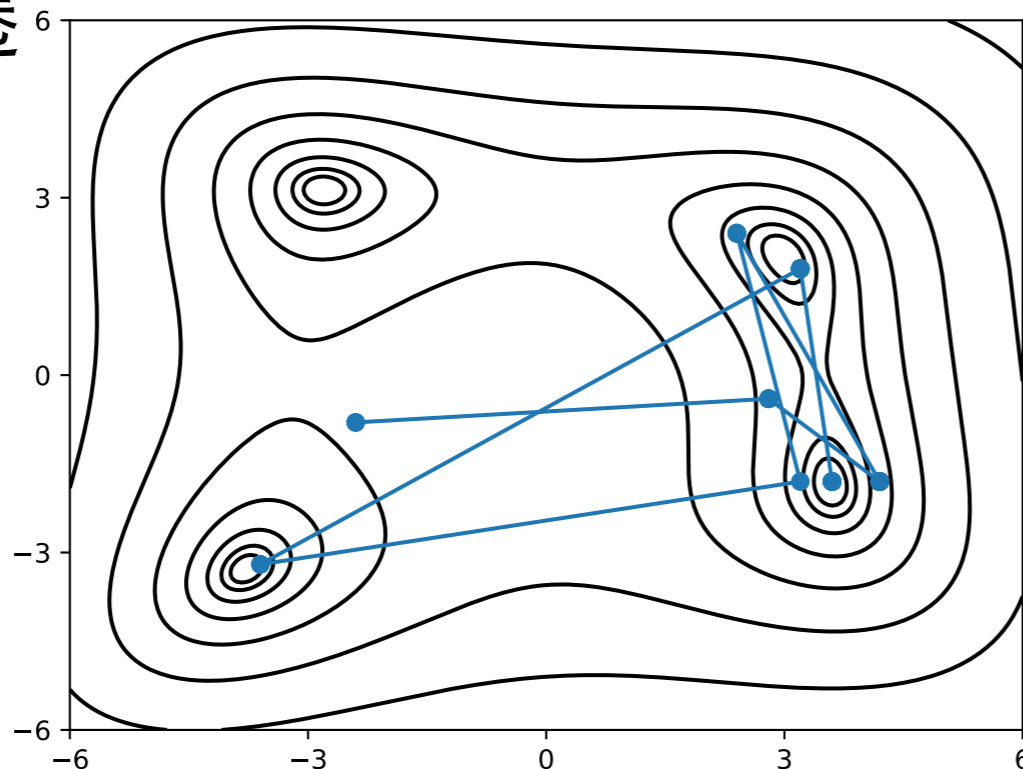
探索した点

output/actions.pdf



最小値の変遷

output/res.pdf



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
```

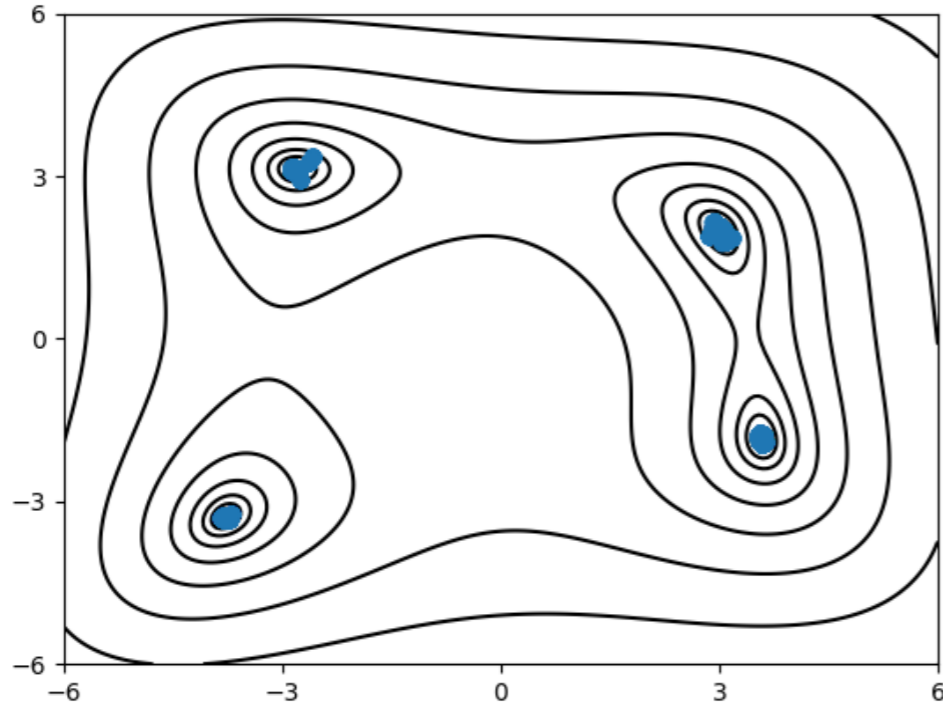
```
[solver]
name = "analytical"
function_name = "himmelblau"
```

交換モンテカルロ (exchange)

output/result_T0.txt

=> output/res_T0.png

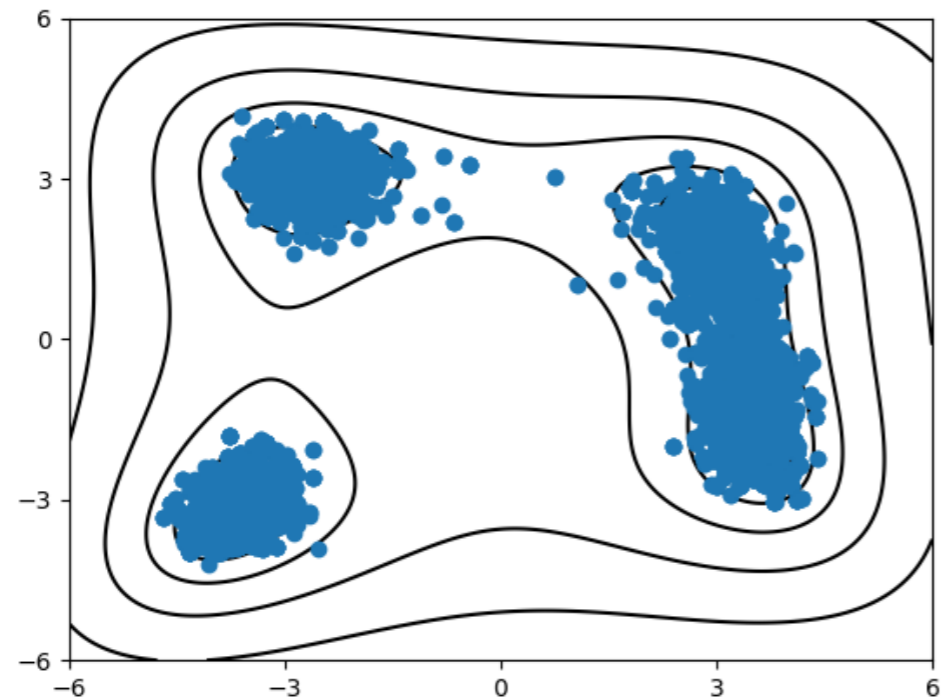
T=0.1



output/result_T3.txt

=> output/res_T3.png

T=10



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]
initial_list = [0.0, 0.0]
```

```
[algorithm.exchange]
T_min = 0.1      # 温度の下限
T_max = 10.0    # 温度の上限
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"
```

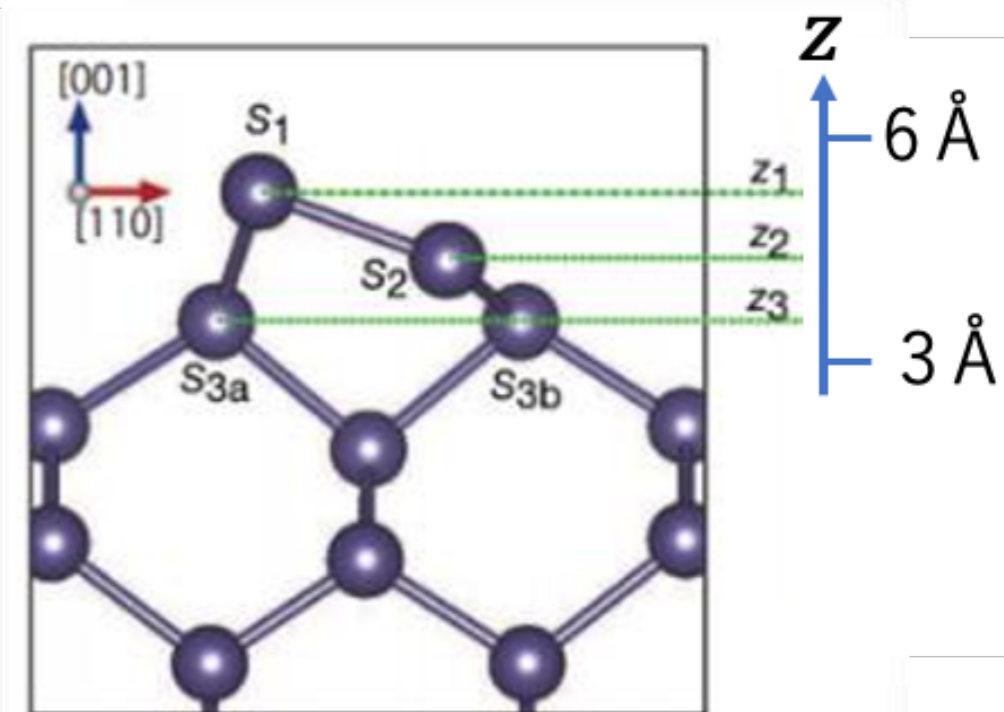
最初の20点は除外してある (初期緩和)

交換モンテカルロの後処理

- 交換モンテカルロ法では、MPI 並列の数だけレプリカ (=温度点) を用いる
- 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
- 実際には各温度でのサンプリングに並び替える必要がある
 - script/separator.py で並び替え
 - \$ python3 ~/2DMAT/script/separator.py output
 - output 以下に result_T0.txt から result_T3.txt が生成される
 - ファイルの先頭に温度が書かれている

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/py2dmat/ 以下の bayes, exchange, mapper, minsearch 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 ($z_1, z_2, z_3=z_{3a}=z_{3b}$) 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)
 - z_1 と z_2 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)

- `~/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 bayes`
 - `$ cp ~/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]
calculated_first_line = 5 # surf-bulkP.s 中、D(x) の最初の行番号
calculated_last_line = 74 # D(x) の最後の行番号
row_number = 2           # 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
degree_max = 7.0          # 視射角の最大値（読み取ったものと異なっても警告が出るだけ）

[solver.reference]
path = "experiment.txt"  # 実験データ Dexp が書いてあるファイル
first = 1                # Dexp として使う部分の最初の行番号
last = 70                # Dexp の最後の行番号
```

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

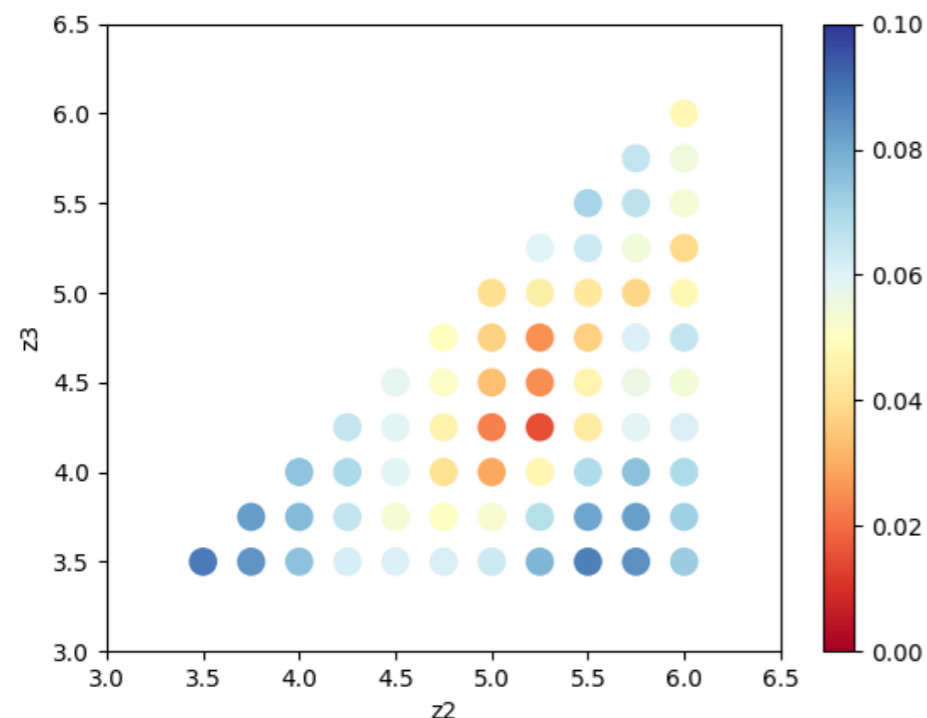
mapper

実行方法および出力

```
$ cd ~/2DMAT/sample/mapper
$ cp ~/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 $z1 \geq z2$ に空間を制限している

minsearch

実行方法および出力

```
$ cd ~/2DMAT/sample/minsearch
$ cp ~/sim-trhepd-rheed/src/*.exe .
$ sh ./do.sh
    ... many outputs ...
Solution:
z1 = 5.230524973874179
z2 = 4.370622919269477
z3 = 3.5961444501081647
```

input.toml のalgorithm 部分

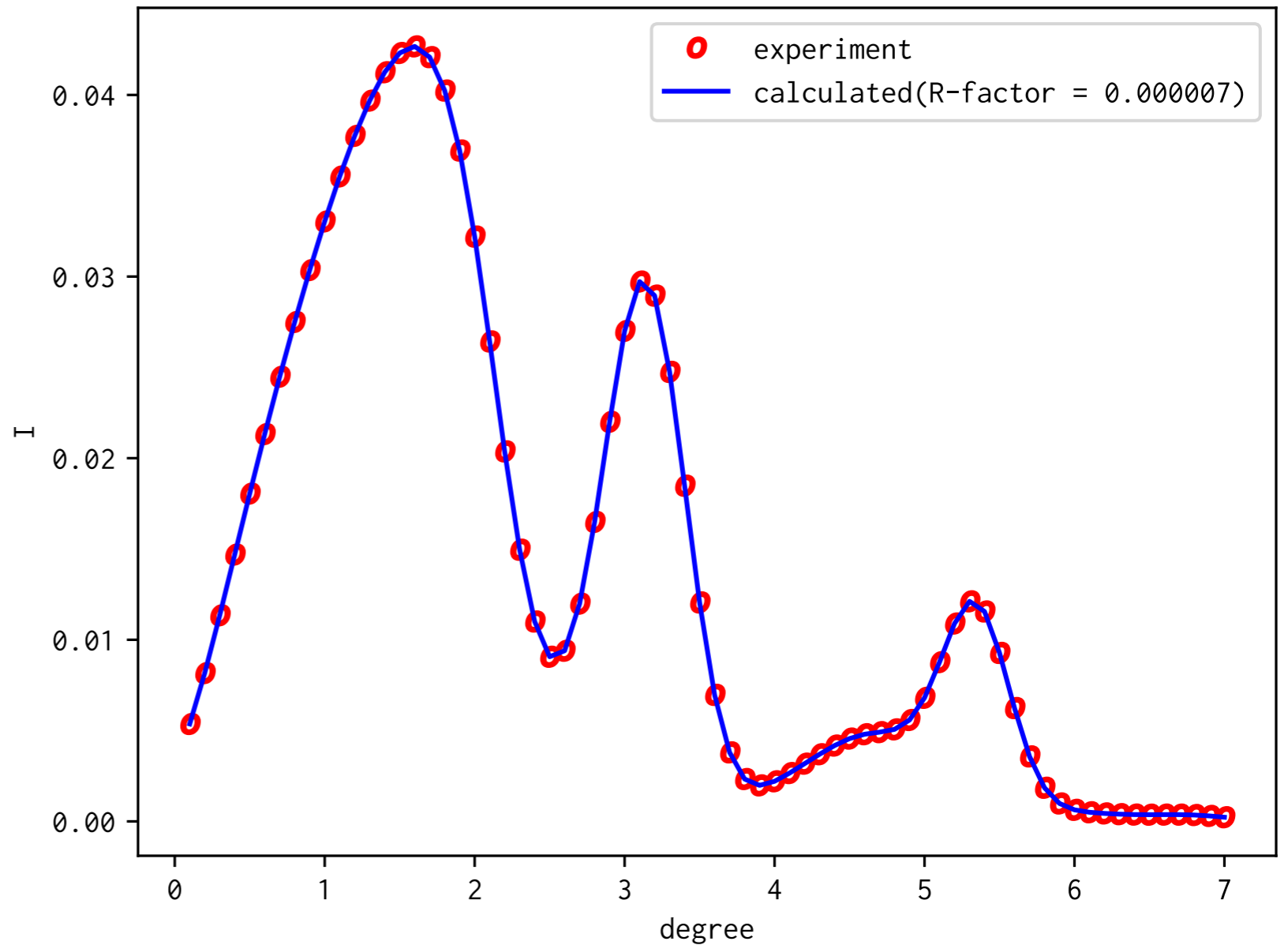
```
[algorithm]
name = "minsearch"
label_list = ["z1", "z2", "z3"]
[algorithm.param]
min_list = [0.0, 0.0, 0.0]
max_list = [10.0, 10.0, 10.0]
initial_list = [5.25, 4.25, 3.50]
```

rocking curve

- `<MPIRANK>/Log*` と `Extra_Log*` に `surf.exe` の入出力が保存されている
 - `Extra_Log` は `minsearch` のみ出力される
 - Nelder-Mead のiteration のうち、解が更新されるたびに追加される
 - `SimplexData.txt` の各ステップに対応
 - 中にあるファイルの詳細は `Log*` と同じ
- `Log*/RockingCurve.txt` に計算で得られた rocking curve が記録されている
 - いくつか列があるが、1 列目が視射角で、4列目が理論曲線（規格化済み）、5 列目が実験曲線（規格化済み）
 - `script/draw_RC_single.py` を実行することで画像に出力可能
 - カレントディレクトリ以下にある `RockingCurve.txt` を描画する

rocking curve

```
$ cd 0/Extra_Log00000062 # 最終結果  
$ python3 ~/2DMAT/script/draw_RC_single.py  
$ firefox 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/sample/exchange
$ cp ~/sim-trhepd-rheed/src/*.exe .
$ sh ./do.sh
```

... many outputs ...

Result:

```
rank = 2
step = 65
fx = 0.008233957976993406
z1 = 4.221129370933539
z2 = 5.139591716517661
```

input.toml の algorithm 部分

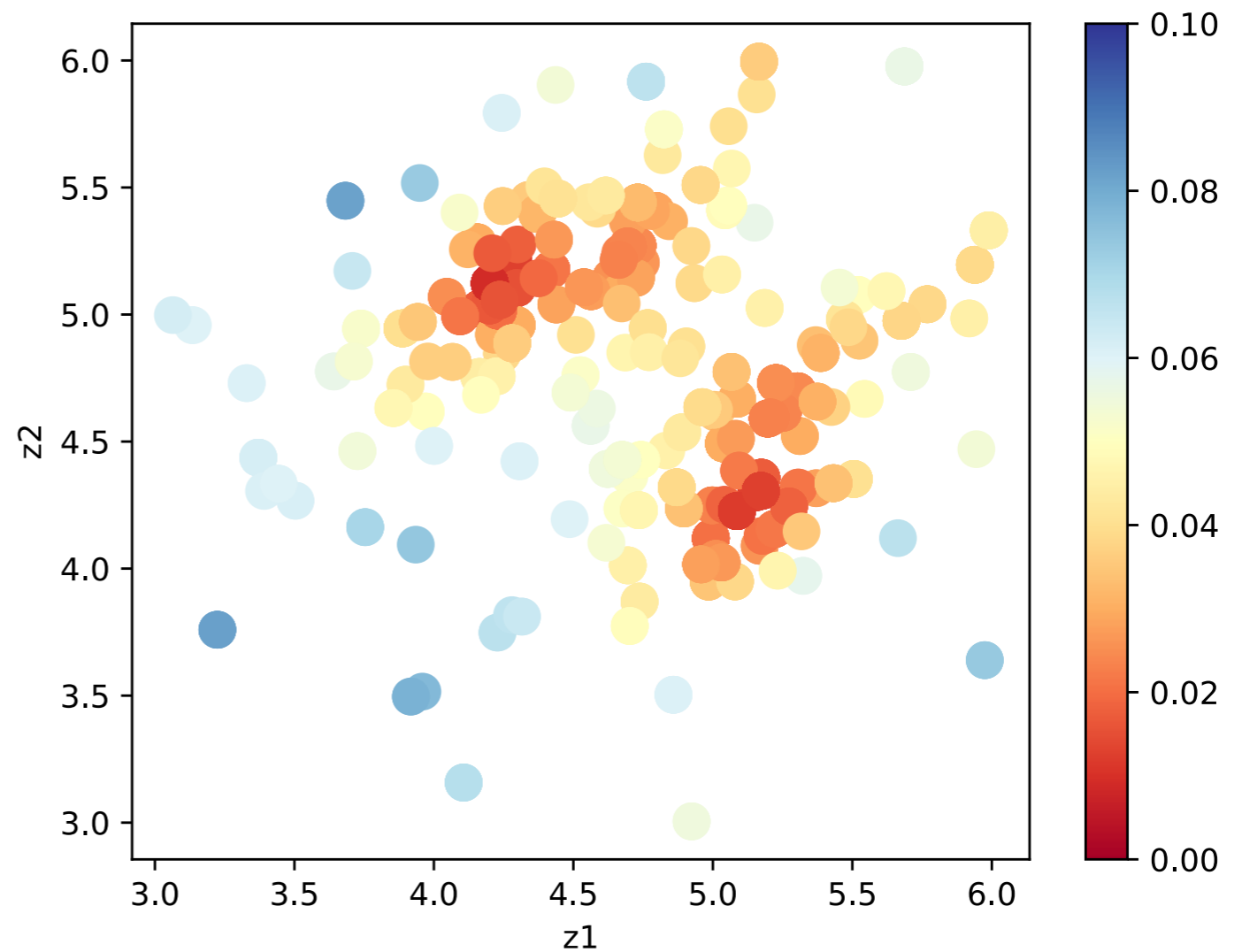
```
[algorithm]
name = "exchange"
label_list = ["z1", "z2"]
seed = 12345
```

```
[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

```
$ python3 ~/2DMAT/script/separateT.py . # 温度ごとにまとめる  
$ 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

bayes

実行方法および出力

```
$ cd ~/2DMAT/sample/bayes
$ cp ~/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
```

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
```

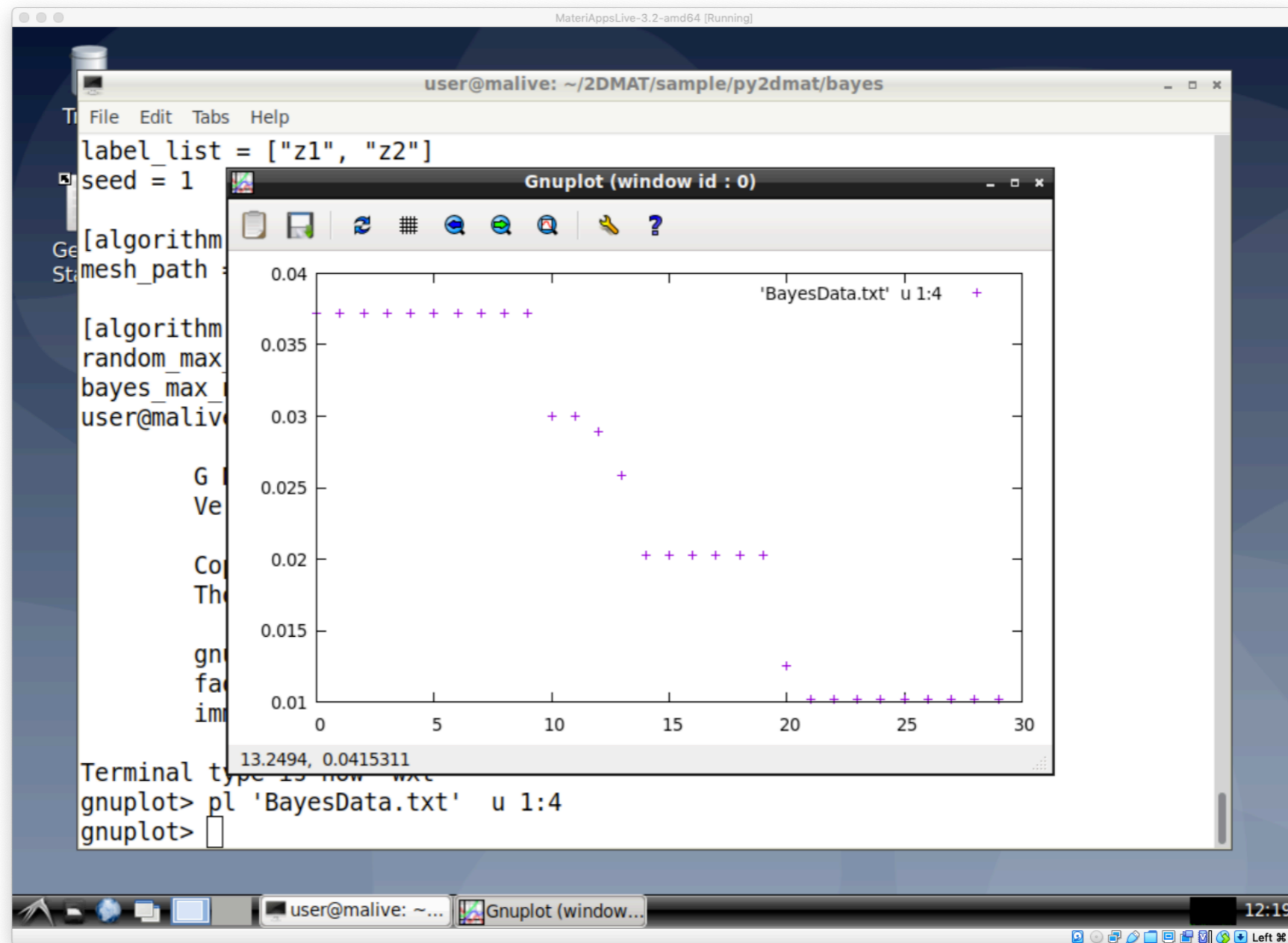
MeshData.txt として

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
```

bayes (2) 描画

```
$ gnuplot  
gnuplot> pl '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)

その他（演習例）

- それぞれの手法でパラメータ数を変えてみる
 - 刻み幅
 - 変数の数
- bayes などの最終結果を minsearch の初期条件 (initial_list) に設定してみる
 - 関数の評価回数は変わる？
- minsearch 以外でも rocking curve を描画してみる