ALPS チュートリアル: Introduction to ALPS Python

CMSI 神戸ハンズオン

ALPS Collaboration
http://alps.comp-phys.org/

1 ALPS python

2 Preparing the simulation

3 Running the simulation

4 Loading the simulation results

5 Plotting the results

6 Evaluating data

Launching Python

phi 上では

```
bash> source /opt/local/pythonvars-2.7.sh
bash> source /opt/nano/alps/alpsvars-20121208-r6630.sh
```

通常のインストールでは

```
bash> alpspython
```

alps python へのインストール先へパスを通すコマンドです.

Importing the ALPS modules

```
>>> import pyalps
>>> import matplotlib.pyplot as plt
>>> import pyalps.plot
```

- pyalps **は必ず必要です**
- matplotlib.pyplot **は図をプロットするのに必要です**
- pyalps.plot **は図をプロットするときやデータの変化を行うのに必要です**

Preparing the input

```
>>> parms = []
>>> for t in [1.5,2,2.5]:
>>>     parms.append(
>>>         {
>>>           'LATTICE'          : "square lattice",
>>>           'T'                : t,
>>>           'J'                : 1 ,
>>>           'THERMALIZATION'   : 1000,
>>>           'SWEEPS'           : 100000,
>>>           'UPDATE'           : "cluster",
>>>           'MODEL'            : "Ising",
>>>           'L'                : 8
>>>         }
>>>     )
```

Preparing the input

```
>>> from parms import *
>>> print parms
[{'LATTICE': 'square␣lattice', 'T': ...
```

- Wiki ではインタープリタ上でパラメータを入力していますが, ここでは import を使ってファイルから読み込みました.
- カレントディレクトリに parms.py というファイル名で先ほどの内容のスクリプトを作成して置きます. これを import します.
- phi 上では LATTICE_LIBRARY, MODEL_LIBRARY の指定が必要です
- 辞書型を要素に持つリストが定義されています

Preparing the input

先程作ったパラメータファイルを XML 形式のインプットファイルに変換します

```
>>> input_file = pyalps.writeInputFiles('parm1',parms)
>>> print input_file  # 入力ファイル名が返されいる
parm1.in.xml
>>> (ctrl-z)
[1]+ 停止 python2.7
[halm@phi ~]# ls
ALPS.xsl parm1.in.xml parm1.task1.in.xml parm1.task2.
    in.xml parm1.task3.in.xml
```

- python を一時停止してカレントディレクトリを確認すると入力ファイルがいくつかできています.
  - parm1.in.xml はシミュレーション全体の入力ファイルです
  - parm1.task# .in.xml は各タスク毎の入力ファイルです

Running the simulation on a serial machine

```
>>> pyalps.runApplication('spinmc',
        input_file,Tmin=5,writexml=True)
spinmc parm1.in.xml --Tmin 5 --write-xml
Generic classical Monte Carlo program using local or
    cluster updates
...
Checkpointing Simulation 3
Finished with everything.
(0, 'parm1.out.xml')
```

- **実行が終了すると** parm1.out.xml, parm1.task1.out.h5, $\cdots$ **といったファイルが出力されています.**

Running the simulation on a parallel machine

```
>>> pyalps.runApplication('spinmc',input_file,Tmin=5,
    writexml=True,MPI=4)
```

- MPI=4 で MPI プロセス 4 並列でシミュレーションを実行することを指示しています.

Getting the result files

```
>>> result_files = pyalps.getResultFiles(prefix='parm1
    ')
>>> print result_files
['./parm1.task2.out.xml', './parm1.task3.out.xml', './
    parm1.task1.out.xml']
```

- カレントディレクトリ内からプレフィックスが 'parm1' のファイルを正規表現によるパターンマッチで探してファイル名を出力します.

Loading the results

指定したファイル名のリストから，指定した物理量（複数可）を取り出す

```
>>> data = pyalps.loadMeasurements(result_files,['|
    Magnetization|','Magnetization^2'])
>>> data[0]  # data の中身をみてみます
[x=[0]
y=[0.083339021352 +/- 0.00038780838614]
props={'observable': '|Magnetization|', '
    THERMALIZATION': 10000.0, 'J': -1.0, ...},
x=[0]
y=[0.0081675955476 2 +/- 7.41432666026e-05]
props={'observable': 'Magnetization^2', '
    THERMALIZATION': 10000.0, 'J': -1.0, ...}]
```

データの構造

シミュレーション結果を表すクラス

- **DataSet**: class 名
  - x: plot 用のインデックス (最初はデフォルト値 0 が入っている)
  - y: 物理量の平均値および標準誤差
  - props: 物理量名およびパラメータなどが Python の辞書形式で入っている

```
# 物理量の平均値と標準誤差を取り出してみます
>>> data[0][0].y
array([0.083339021352 +/- 0.00038780838614], dtype=
    object)
>>> data[0][0].y.mean
0.08333902135203504
>>> data[0][0].y.error
0.00038780838614
```

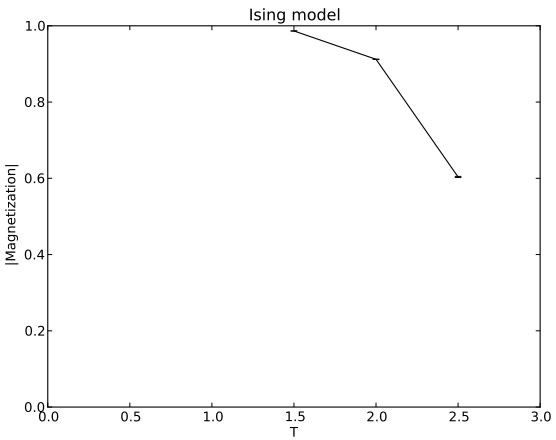
Plotting the results

```
>>> plotdata = pyalps.collectXY(data,'T','|
    Magnetization|')
>>> plotdata[0].x
array([ 1.5,  2. ,  2.5])
>>> plotdata[0].y
array([0.986621683313 +/- 5.59763992999e-05,
       0.911889051072 +/- 0.00028843276697,
       0.605402476752 +/- 0.0013356753695], dtype=
           object)
>>> plotdata[0].props
{'observable': '|Magnetization|', '...
```

- data から pyalps.collectXY により x 軸として 'T', y 軸として '|Magnetization|' を取り出しています.

Plotting in Python using matplotlib

```
>>> pyalps.plot.plot(plotdata) # データをプロット.
[<matplotlib.lines.Line2D at 0x109425dd0>]
>>> plt.xlim(0,3)                    # x 軸の範囲を設定
>>> plt.ylim(0,1)                    # y 軸の範囲を設定
>>> plt.title('Ising␣model')         # 図のタイトル
>>> plt.savefig('ising.pdf')         # ファイルへ出力
>>> plt.show()                       # X11 などで画面に表示
```



Converting to other formats

ほかのデータフォーマットに変換できます

```
>>> print pyalps.plot.convertToText(plotdata)
>>> print pyalps.plot.makeGnuplotPlot(plotdata)
>>> print pyalps.plot.makeGracePlot(plotdata)
```

text 形式の出力例

```
#
# X: T
# Y: |Magnetization|
1.5 0.986496240665 +/- 2.82049446481e-05
2.0 0.912126100521 +/- 0.00035020147466 7
2.5 0.603552651599 +/- 0.0014615425377
```

Example of evaluating data

binder 比 $< m^2 > / < |m| >^2$ の計算をしてみます

```
>>> binder = pyalps.DataSet()
>>> binder.props = pyalps.dict_intersect([d[0].props
    for d in data])
>>> binder.x = [d[0].props['T'] for d in data]
>>> binder.y = [d[1].y[0]/(d[0].y[0]*d[0].y[0]) for d
    in data]
>>> print binder
```

- 空のデータセットを作ります
- 複数の辞書から key:val ともに一致する項目を抜き出す
- データから x 軸となる 'T' の数列をとりだす.
- $d[0]$ から m^2, $d[1]$ から $|m|$ を取り出して binder 比を計算