

TeNeS

Tensor Network Solver for Quantum Lattice Systems

on



Yuichi Motoyama (ISSP)

2022-10-25
for TeNeS ver 1.3.1

TeNeS を最新版にする

- TeNeS を使う前にMALIVE! に入っているTeNeS を更新しておきます
 - パスワードは live

```
$ sudo apt update          # パッケージ一覧を更新する  
$ sudo apt install tenes   # TeNeS を更新する
```

(先頭の \$ は入力待ち状態を示す記号なので入力しないでください)

(# 以降はコメントを示しています、入力しなくて大丈夫です)

サンプルをコピー

- TeNeS のサンプル集を手元にコピーしてきます

```
$ cd # HOME directory へ移動
$ cp -r /usr/share/tenes . # TeNeS のサンプルをコピー
$ cd tenes/sample
```

(ファイル名などは途中まで入力した後に Tab キーを押すと適宜補完可能です)

sample-1 横磁場イジング模型

- 最初の例として正方格子横磁場イジング模型を扱います
 - $S=1/2$ の演算子で書かれていることと符号に注意
 - $h_x \sim 1.5$ あたりに量子相転移があります

$$\mathcal{H} = J_z \sum_{\langle ij \rangle} S_i^z S_j^z - h_x \sum_i S_i^x$$

```
$ cd 01_transverse_field_ising
$ cat simple.toml # 入力ファイルの確認
... Skipped ...
```

```
[lattice]
type = "square lattice"  正方格子
L = 2                    2x2 ユニットセル
W = 2
virtual_dim = 2          ボンド次元 = 2
initial = "ferro"

[model]
type = "spin"           スピン模型
Jz = -1.0               強磁性イジング Jz=-1, Jx=Jy=0
Jx = 0.0
Jy = 0.0
hx = 0.0               横磁場 hx = 0
```

sample-1 横磁場イジングモデル

- 入力ファイル `simple.toml` を `input.toml` へと変換して `tenes` を実行

```
$ tenes_simple simple.toml # convert to std.toml
$ tenes_std std.toml      # convert to input.toml
$ tenes input.toml        # main calculation

# ... Skipped ...

Onesite observables per site:
  Sz          = 0.5 0          <Sz> の実部・虚部
  Sx          = -1.28526262482e-13 0  <Sx> の実部・虚部
Twosite observables per site:
  hamiltonian = -0.5 0          エネルギー
  SzSz        = 0.5 0
  SxSx        = -1.7374919982e-18 0  最近接相関
  SySy        = 1.73749202733e-18 0

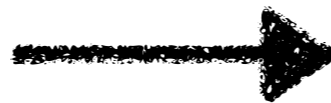
# ... Skipped ...
```

sample-1 横磁場イジングモデル

- 横磁場を入れてみましょう
 - 具体的には、 $hx = 3$ に変更してみます

```
$ mv output output_hx0 # 計算結果を退避しておく
$ mousepad simple.toml # 入力ファイルを編集
```

```
[model]
type = "spin"
Jz = -1.0
Jx = 0.0
Jy = 0.0
hx = 0.0
```



```
[model]
type = "spin"
Jz = -1.0
Jx = 0.0
Jy = 0.0
hx = 3.0
```

(mousepad はメモ帳のような単純なエディタです)

(MALIVE! には他にも vim, emacs, nano が入っているのでお好きにどうぞ)

(MALIVE! on docker では vim と emacs のみ?)

sample-1 横磁場イジングモデル

- 編集した simple.toml を input.toml へと変換して tenes を実行

```
$ tenes_simple simple.toml # convert to std.toml
$ tenes_std std.toml      # convert to input.toml
$ tenes input.toml        # main calculation
```

```
# ... Skipped ...
```

Onesite observables per site:

```
Sz          = 8.05352165651e-09 0    <Sz> の実部・虚部
```

```
Sx          = 0.492509565167 0     <Sx> の実部・虚部
```

Twosite observables per site:

```
hamiltonian = -1.52140952585 0     エネルギー
```

```
SzSz        = 0.0438808303474 0
```

```
SxSx        = 0.488706591605 0     最近接相関
```

```
SySy        = -0.0407090179083 0
```

```
# ... Skipped ...
```

sample-1 横磁場イジングモデル

- output ディレクトリに更に詳細な結果が出力されます
- 例えば S_i^z などの1サイト演算子のサイト毎の期待値は oneseite_obs.dat に出力

```
$ head output/onesite_obs.dat
```

```
# $1: op_group
```

```
# $2: site_index
```

```
# $3: real
```

```
# $4: imag
```

```
0 0 8.05351406015284822e-09 0.000000000000000000000000e+00
```

```
0 1 8.05351251325784800e-09 0.000000000000000000000000e+00
```

```
0 2 8.05351273424285110e-09 0.000000000000000000000000e+00
```

```
0 3 8.05351179505659039e-09 0.000000000000000000000000e+00
```

```
1 0 4.92509565166792285e-01 0.000000000000000000000000e+00
```


sample-1 横磁場イジングモデル

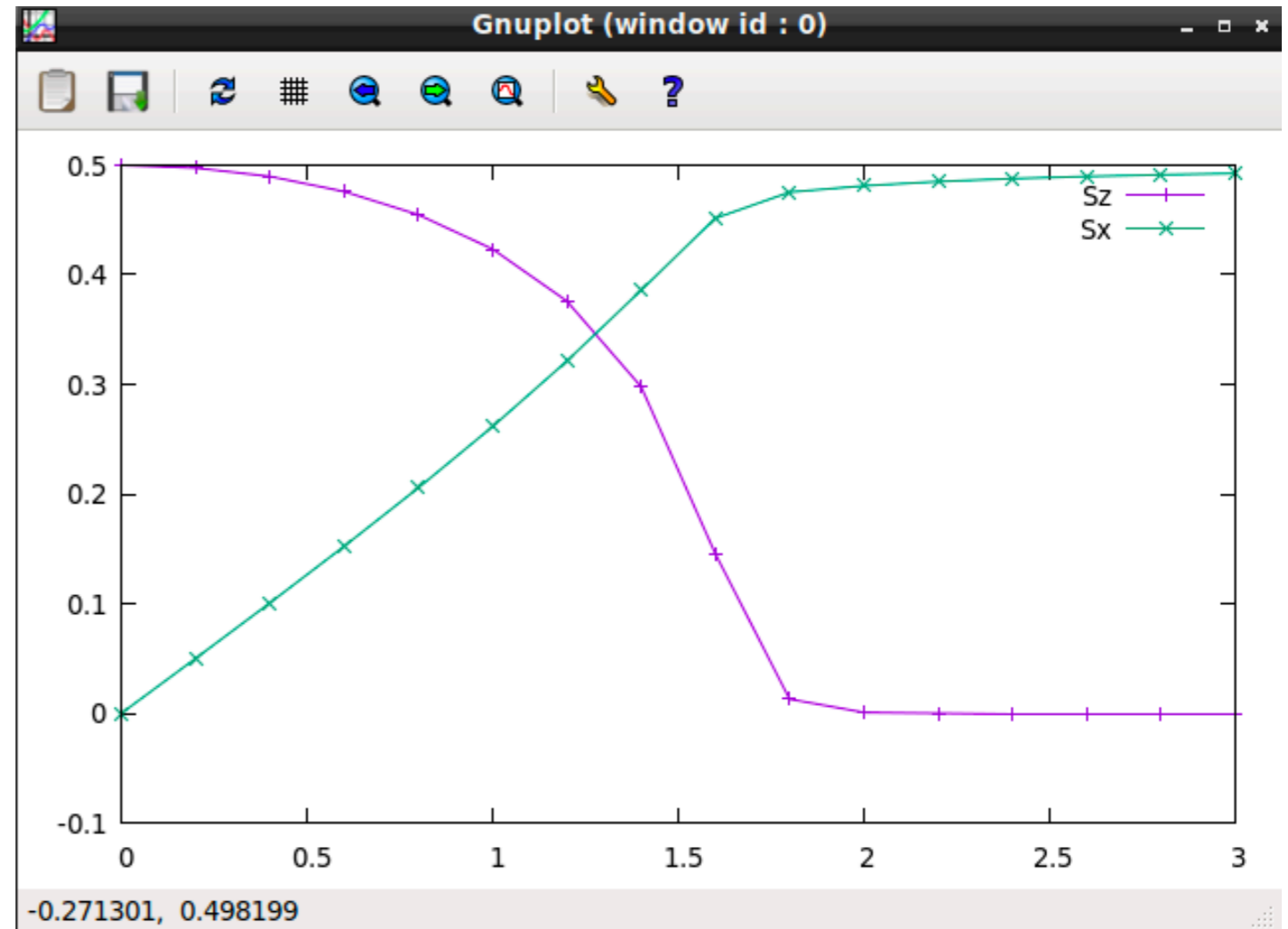
- 横磁場をすこしずつ変えながら計算していくと量子相転移を観測できます
 - 単純作業を手でやるのは不毛なのでプログラムを書くのが良いです
 - `tutorial_example.py` と `tutorial_read.py` が用意されています

```
$ mv output output_hx3          # 一応計算結果を退避しておく
$ python3 tutorial_example.py    # ひたすら実行
$ python3 tutorial_read.py       # 計算結果をまとめる
0.0 -5.00000000000000000000000e-01 5.00000000000000000000000e-01
-1.28526262481782106e-13
0.2 -5.05004176692116613e-01 4.97482489246813320e-01
5.00836559134336876e-02
0.4 -5.20067358911737165e-01 4.89712356048554176e-01
1.00677102836350646e-01
    # ... Skipped ...

$ python3 tutorial_read.py > result.dat # ファイルへ保存
```

sample-1 横磁場イジングモデル

- 出力された 4列の意味は順番に、
 - 横磁場 h_x
 - サイトあたりのエネルギー
 - サイトあたりの縦磁化 S_z
 - サイトあたりの横磁化 S_x
- gnuplot でプロットしてみましよう

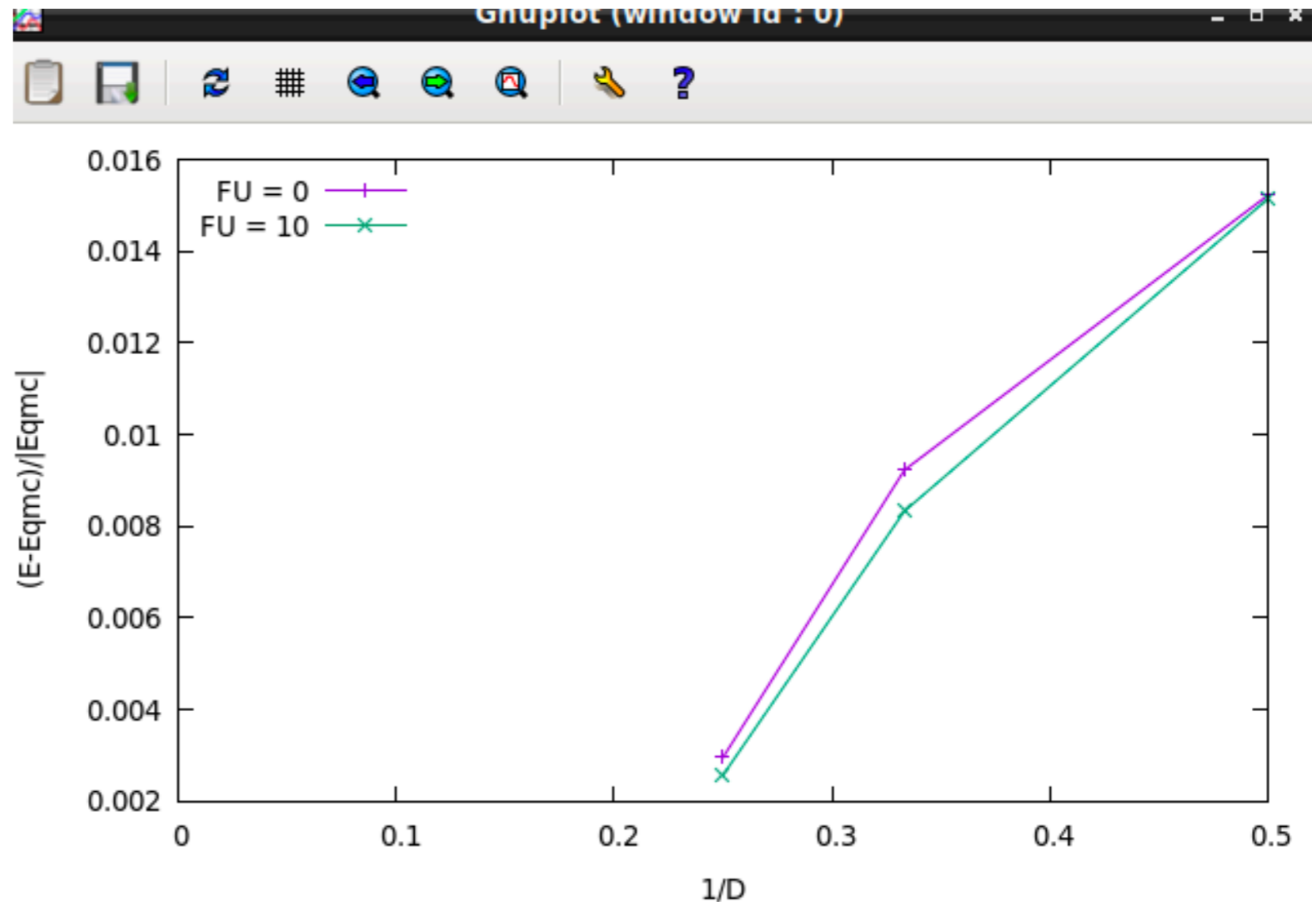


```
$ gnuplot
gnuplot> set style data lp
gnuplot> pl 'result.dat' u 1:3 t 'Sz', '' u 1:4 t 'Sx'
```

(実際の計算ではボンド次元などのハイパーパラメータを変えて結果の収束のしかたを確認します)

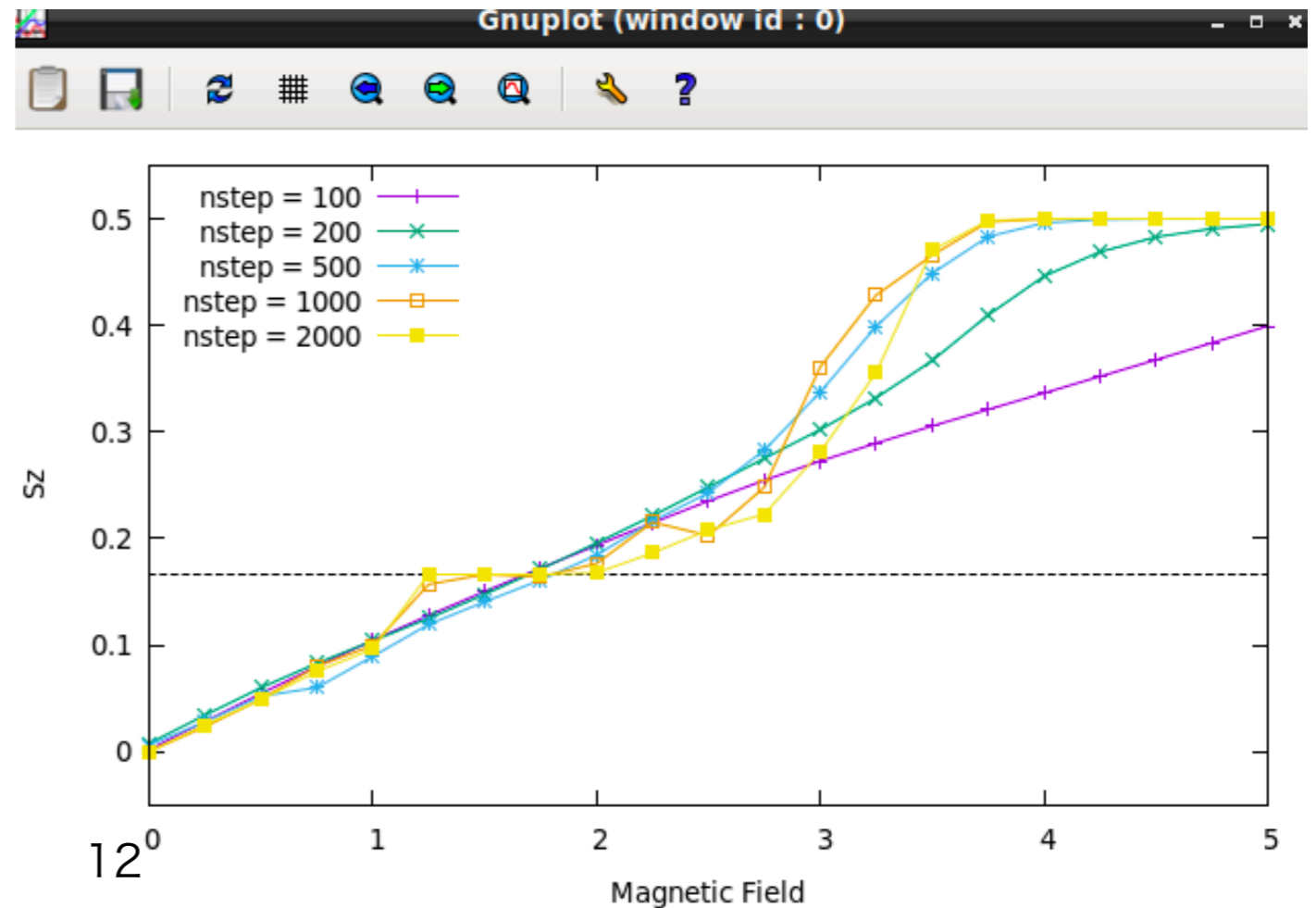
その他のサンプル

- 02_AFH_square
 - 正方格子 $S=1/2$ 反強磁性ハイゼンベルグ模型
 - ボンド次元や full update のステップ数を変えたときに得られる基底状態エネルギーの振る舞い



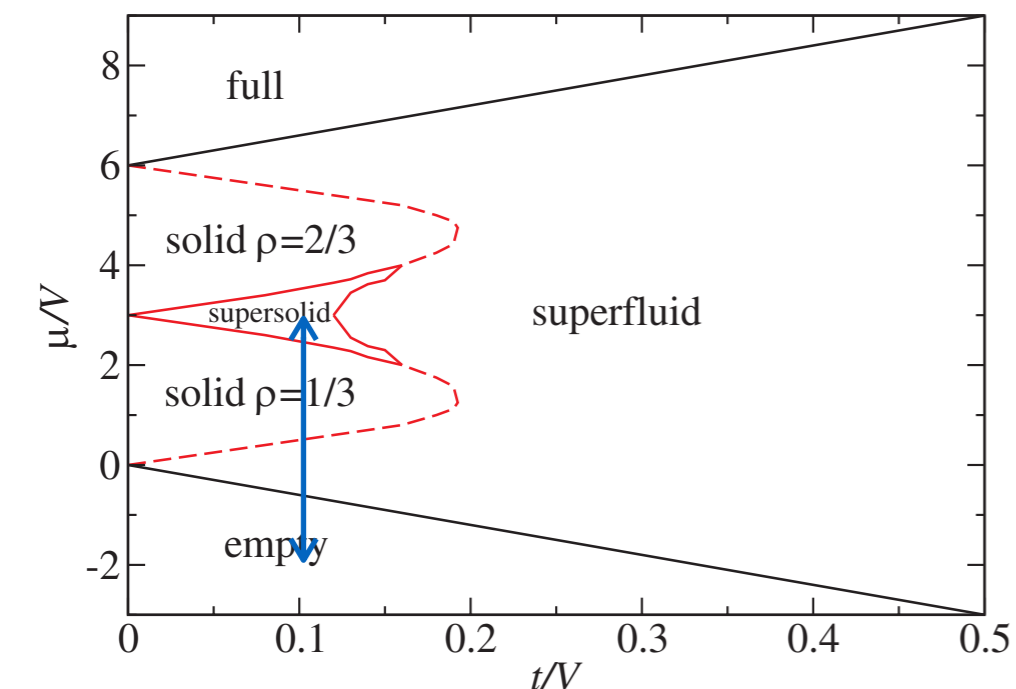
その他のサンプル

- 03_S1_AFH_square
 - $S=1$ 反強磁性ハイゼンベルグ模型の入力ファイル例
- 04_Kitaev_honeycomb
 - $S=1/2$ 蜂の巣格子 Kitaev 模型の入力ファイル例
- 05_magnetization
 - $S=1/2$ 反強磁性ハイゼンベルグ模型の、正方格子・三角格子での磁化曲線

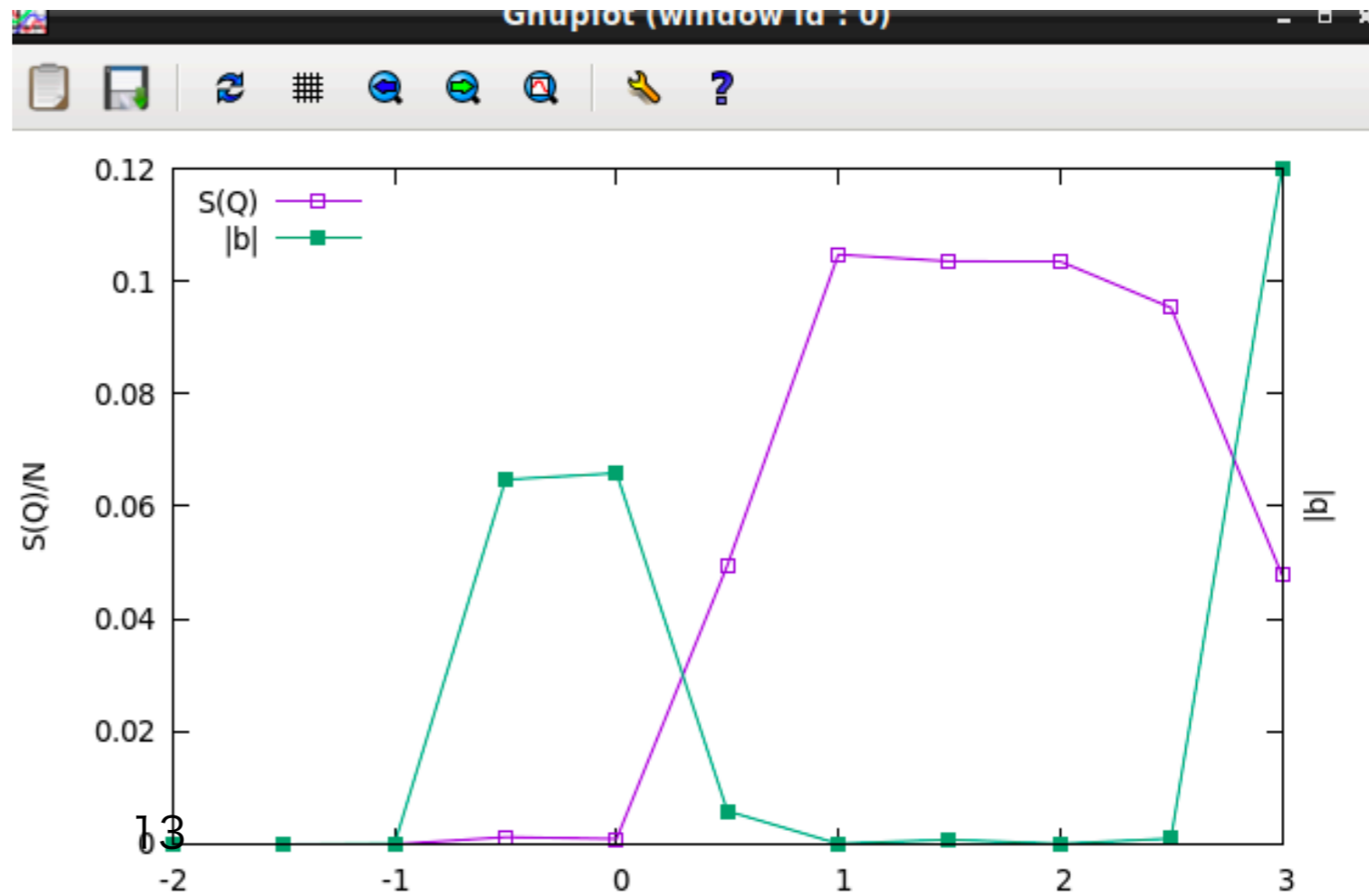


その他のサンプル

- 06_hardcore_boson_triangular
 - 三角格子ハードコアボースハバード模型
 - 固体相・超流動相・超固体相の計算
 - 超流動性は非対角秩序変数 $\langle b_i \rangle$ で判断する
 - 固体相のために構造因子 $S(Q)$ を計算する例を含む (やや高度)



Wessel and Troyer, PRL 95, 127205 (2005)



その他のサンプル

- `std-01_model`
 - `std.toml` を編集して模型を定義するサンプル
 - 交代磁場下のスピン模型を計算してNeel 状態の iTNSを得る
 - `tensor_save` = "ディレクトリ名" で得られた状態 を保存可能
 - `tensor_load` = "ディレクトリ名" で初期状態として利用可能

MALIVE! 以外での注意

- sample 以下にあるPython スクリプトについて
 - `tenes` や `tenes_simple` などを修飾なしに実行します
 - これらの実行ファイルへの PATH が通っている必要があります
 - `tenes` を MPI 並列実行したい場合は、スクリプト中の最初の方にある `MPI_cmd` を修正してください
 - 例: `MPI_cmd = "mpiexec -np 2"`
 - OpenMP の並列数は環境変数 `OMP_NUM_THREADS` を設定してください
- `tenes_simple` / `tenes_std` ではいくつかのPython パッケージが必要です
 - NumPy, SciPy, toml
 - 実行時に見つからなかった場合は適宜入れましょう
 - `pip3 install --user numpy scipy toml`
 - `--user` を使うと自分のホームディレクトリ以下にインストールできます