

TeNeS

Tensor Network Solver for Quantum Lattice Systems

<https://www.pasums.issp.u-tokyo.ac.jp/tenes>

<https://github.com/issp-center-dev/TeNeS>

YM and et al., Comput. Phys. Commun. 279, 108437 (2022)

質問・要望は
GitHub のissue か
メール (tenes-dev@issp.u-tokyo.ac.jp)へ！

Yuichi Motoyama (ISSP)

2022-10-25

for TeNeS ver 1.3.1

TeNeS 開発チーム

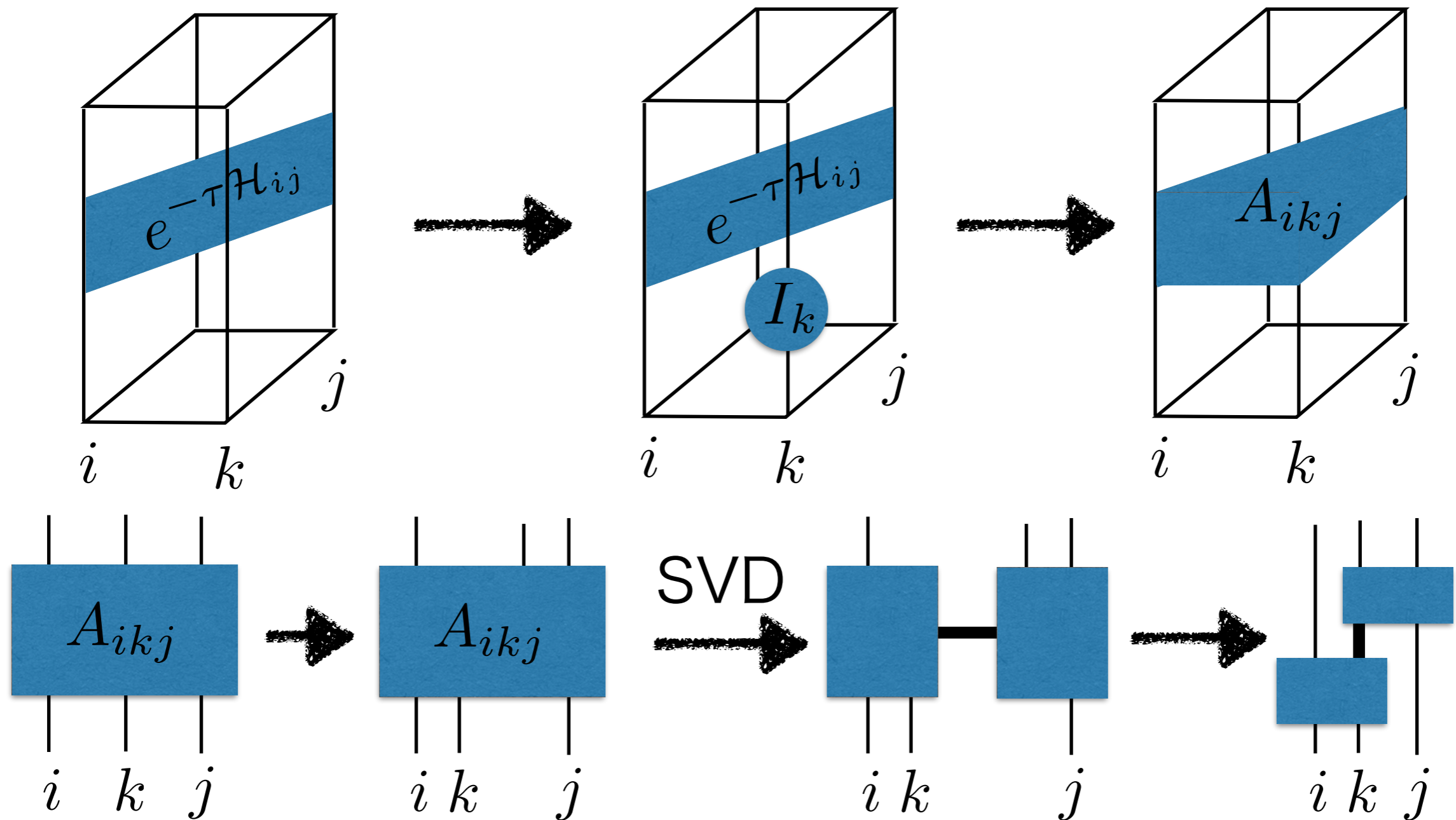
- 大久保 毅 (東大院理)
- 森田 悟史 (東大物性研)
- 本山 裕一 (東大物性研)
- 吉見 一慶 (東大物性研)
- 加藤 岳生 (東大物性研)
- 川島 直輝 (東大物性研)

TeNeS の特徴

- 二次元量子格子模型の基底状態探索を行うソフトウェア
 - 正方格子iTPS 波動関数を ITE (SU/FU) で最適化する
 - 無限系の縮約にはCTMRG または平均場環境を利用する
 - 現状、フェルミオン系は扱えない (スピンとボソンがメイン)
- 正方格子上の任意の短距離2サイト相互作用を計算可能
 - 現状はx, y 方向にそれぞれ3サイト先まで
 - 相互作用の形を工夫することで正方格子以外も計算可能
 - 例：正方格子の斜め方向に入れて三角格子
- 1サイト演算子・2サイト演算子の期待値が計算可能
 - 多サイト演算子や相関長の計算は導入予定
- テンソル演算には `mptensor` (<https://github.com/smorita/mptensor>) を利用
 - MPI 実行するだけで ScaLAPACK を利用した分散メモリ並列がなされる

TeNeS の特徴

- TeNeS のITE は簡単のために正方格子最近接ボンドのみ扱う
- 長距離ITE テンソルは正方格子最近接 ITE テンソルの積に変換しておく



TeNeS の特徴

- よく使われそうな模型・格子は定義済み
- 模型や格子のパラメータを与えるだけで良い
- 模型
 - 量子スピン模型 (スピンの大きさ S は任意)

$$\mathcal{H} = \sum_{i < j} \left[\left(\sum_{\alpha}^{x,y,z} J_{ij}^{\alpha} S_i^{\alpha} S_j^{\alpha} \right) + B_{ij} \left(\vec{S}_i \cdot \vec{S}_j \right)^2 \right] - \sum_i \left[\sum_{\alpha}^{x,y,z} h^{\alpha} S_i^{\alpha} + D (S_i^z)^2 \right]$$

- Bose-Hubbard 模型 (粒子数カットオフ付き)

$$\mathcal{H} = \sum_{i < j} \left[-t_{ij} \left(b_i^{\dagger} b_j + \text{h.c.} \right) + V_{ij} n_i n_j \right] + \sum_i \left[U \frac{n_i (n_i - 1)}{2} - \mu n_i \right]$$

TeNeS の特徴

- よく使われそうな模型・格子は定義済み
- 模型や格子のパラメータを与えるだけで良い
- 格子
 - 格子回転異方性や最近接・2次近接・3次近接相互作用を導入可能
 - 正方格子
 - 三角格子
 - 蜂の巣格子
 - カゴメ格子

TeNeS の目標

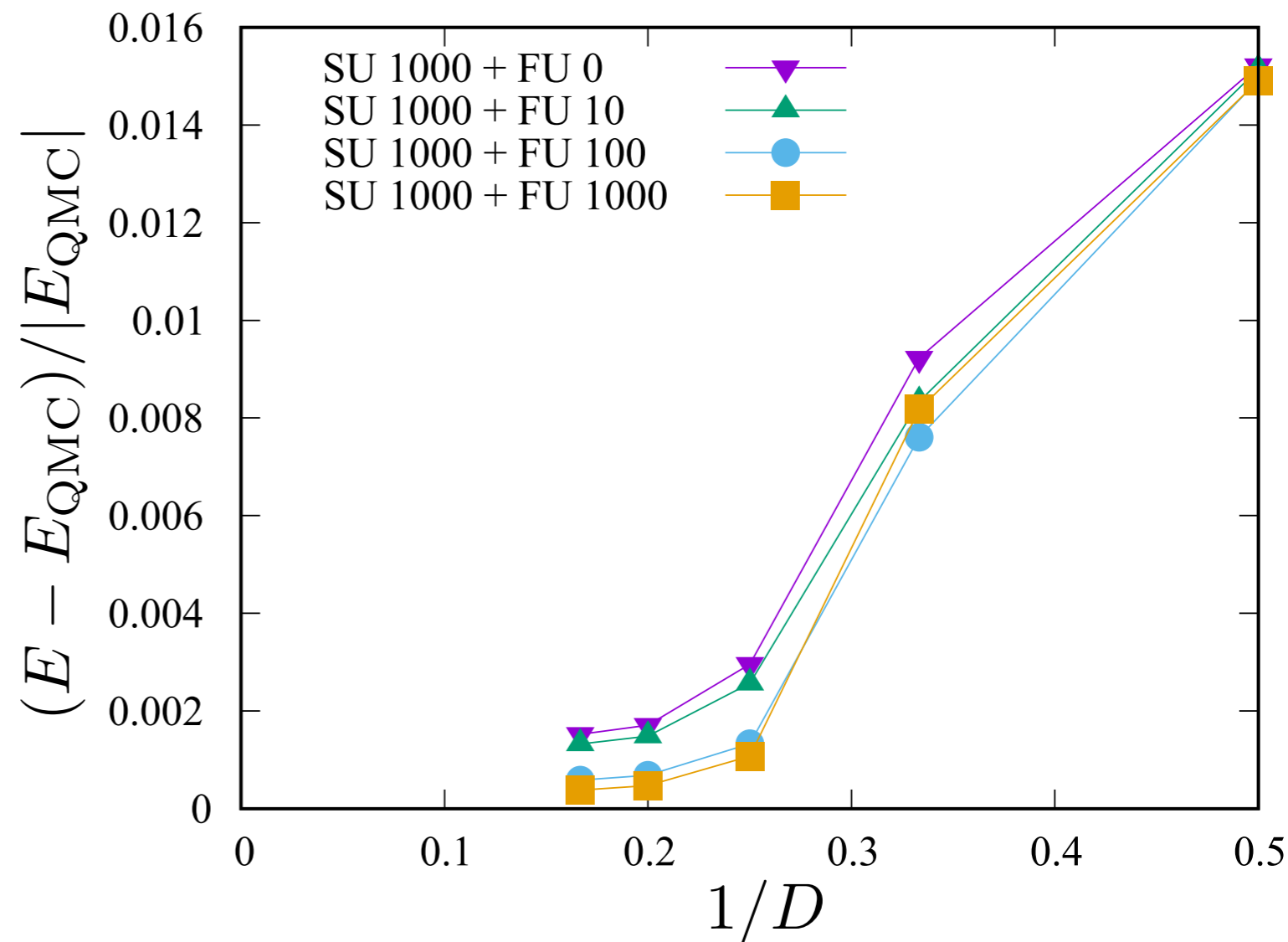
- 量子格子模型におけるテンソルネットワーク法研究への参入障壁を下げる
 - TN 法のアルゴリズム・プログラム開発
 - テンソル演算のライブラリはそれぞれの言語で多数存在
 - アルゴリズムも（ダイアグラム表記のおかげで）割とわかりやすい
 - 実際に1から組み上げるのは結構難しい（つらい）
 - テンソルの添字の順番など、システムティックに構築しないとすぐに混乱する
 - 複数のテンソルを縮約する場合、順番によって計算量オーダーが変わる
 - 計算結果をどう評価するか？
 - ボンド次元など、ハイパーパラメータが結構多い
 - **TeNeS をベンチマークとして用いる！**
 - 計算結果の比較
 - ソースコードを読む・TeNeS をベースに新アルゴリズムを入れる
 - GNU GPL v3 であることに留意

TeNeS の目標

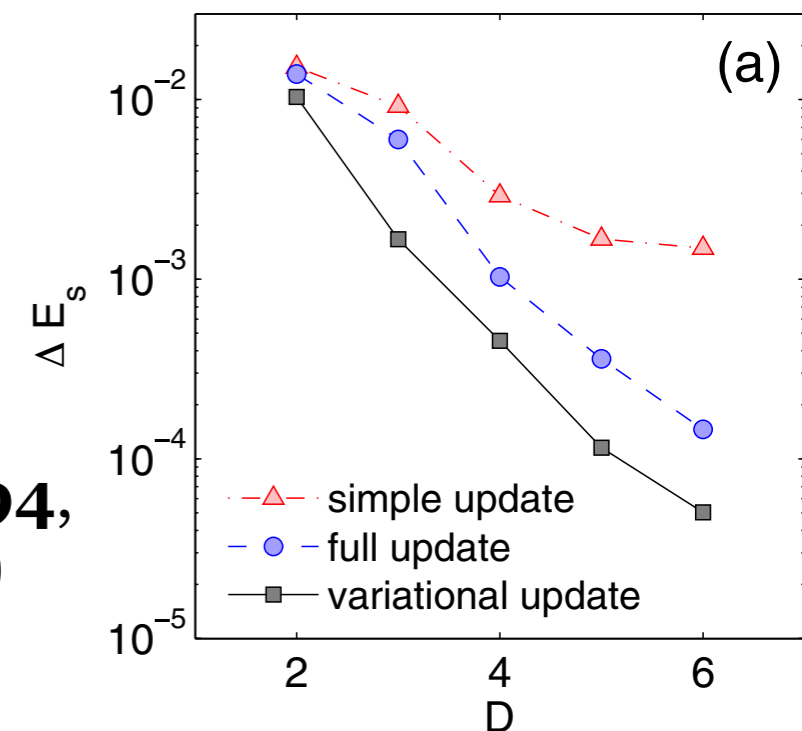
- 量子格子模型におけるテンソルネットワーク法研究への参入障壁を下げる
 - 別の分野の研究者
 - 実験家
 - 実験データのフィッティングなど
 - TN 以外の手法開発者
 - TN 法との比較のために使う
- 使いやすさや汎用性を重視
 - 特定の模型や格子に強く依存した最適化はしない
 - 常に正方格子iTPS として扱う
- 「とりあえずTN 計算を試してみる」ができるようにする

計算例：正方格子 $S=1/2$ 反強磁性ハイゼンベルグ模型のエネルギー

- 正方格子 $S=1/2$ 反強磁性ハイゼンベルグ模型の基底エネルギー
- 横軸はボンド次元の逆数
- 縦軸はQMCによる見積もりからの相対誤差
- A.W. Sandvik, AIP Conf. Proc. **1297**, 135 (2010)
- ユニットセルの大きさは 2×2
- $\chi = D^2$
- 24CPU の計算機で数時間程度

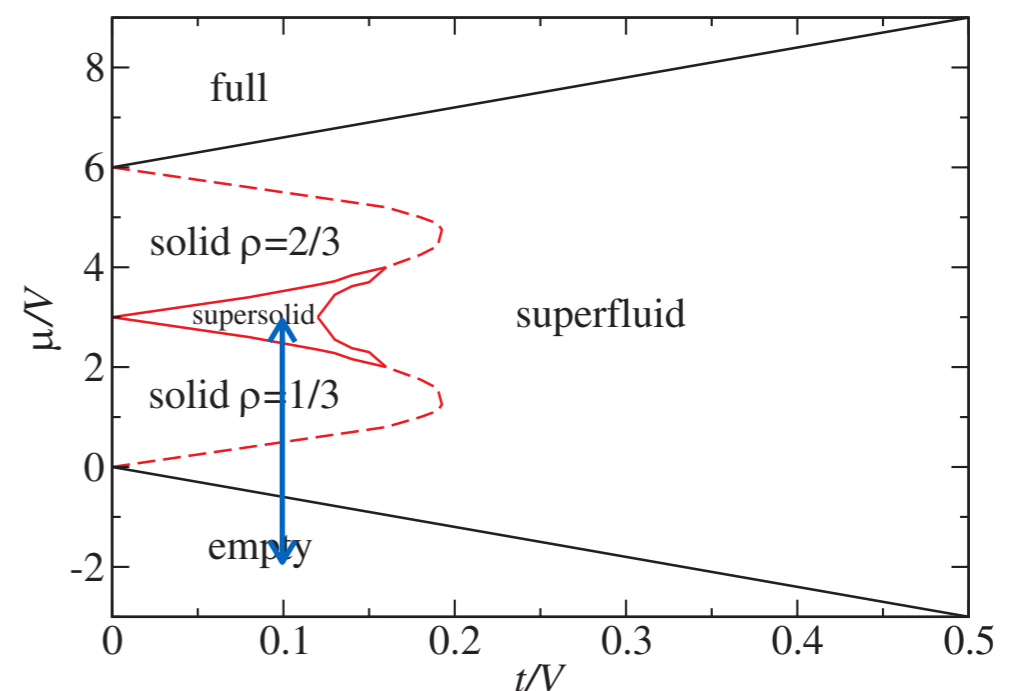
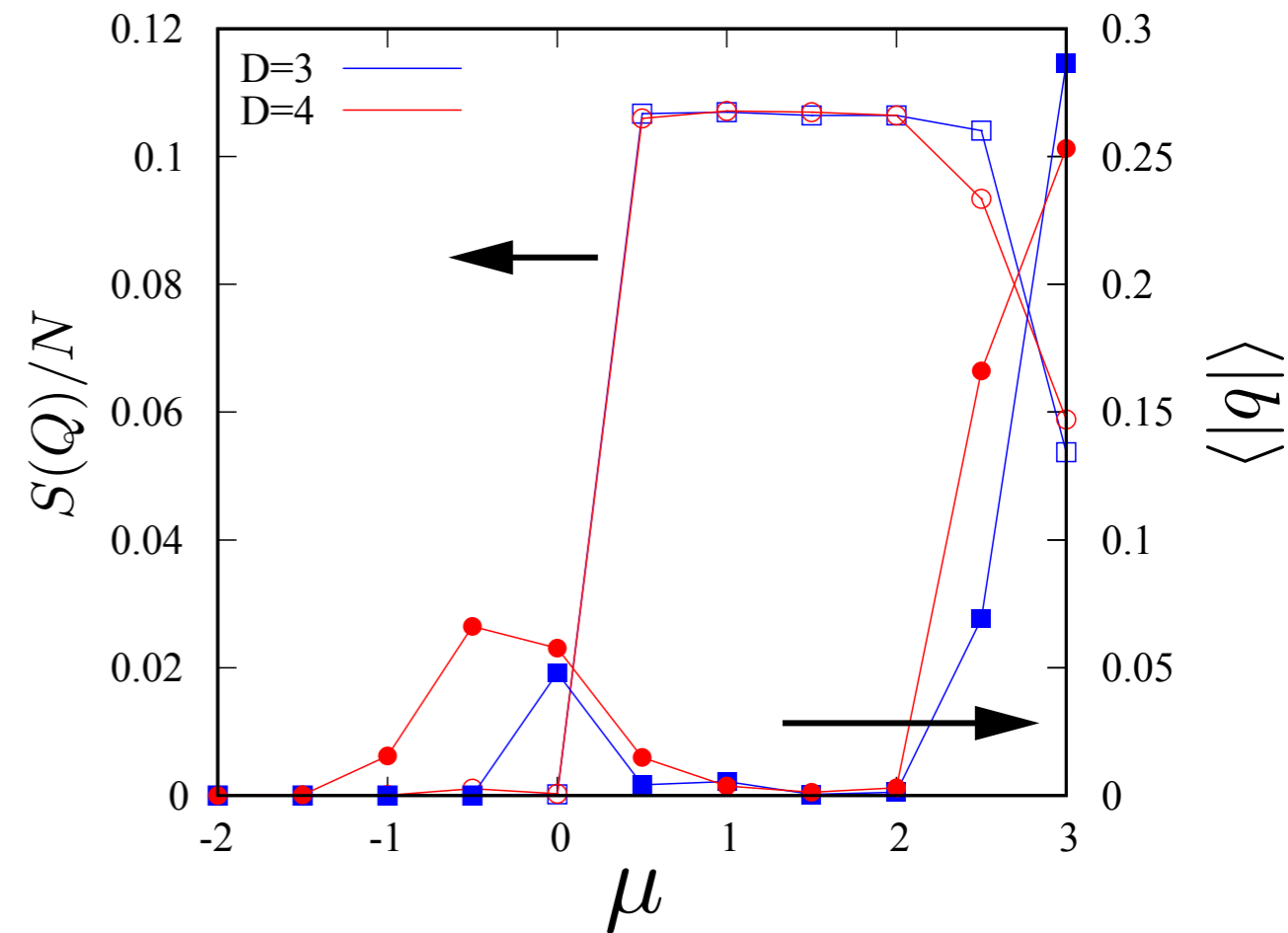


P. Corboz, PRB **94**, 035133 (2016)



計算例：三角格子ハードコアボースハバード模型

- 1/3 相と 2/3 相の間に超固体相がある
 - 超流動性と固体性が同時に存在
- $t=0.1, V=1.0$
- $D=3$ および $D=4$
- $\chi = D^2$
- ユニットセルは 3×3
- SU 2000 step, FU なし
- 構造因子 $S(Q)$ (左軸) と超流動オーダー $|b|$ (右軸)
 - Q は $\sqrt{3} \times \sqrt{3}$ オーダーの波数
- 先行研究(QMC) と矛盾しない結果



Wessel and Troyer, PRL 95, 127205 (2005)

前回講習会のバージョンからの差分

- 前回の講習会 (2020-11-10) は TeNeS v1.1.2 だった
- そこからの主な更新
 - 平均場環境による iTNS 縮約
 - CTMRG の代わりに使える
 - CTMRG より早い $O(D^{10}) \rightarrow O(D^4)$
 - 変分原理を満たさないのでエネルギーの比較には注意
 - (バルクテンソルの) 相関長計算機能
 - 転送行列の固有値を用いて計算される
 - サイトハミルトニアンおよび1サイトITE テンソル
 - これまでは常に2サイト (ボンド) で書いていた
- 論文ができました！
 - YM and et al., Comput. Phys. Commun. 279, 108437 (2022)

TeNeS のつかいかた・準備編

インストール

TeNeS の使い方 -- 前準備

- インストール済み環境を使う
 - **MateriApps LIVE!**
 - 物性研スパコン
- パッケージインストール
 - Debian のパッケージが利用可能
 - 詳しくはMALIVE! のサイトへ
- ソースからインストール
 - 次項・次次項

TeNeS の使い方 -- 前準備

- TeNeS に必要なもの
 - C++11 に対応したコンパイラが必要
 - よほど古い計算機でなければ問題ないはず
 - BLAS/LAPACK が必要
 - 分散並列をしたい場合は MPI, ScaLAPACK も必要
 - CMake3.6 が必要
 - 内部で依存ライブラリをダウンロードするために Git も必要
- 入力ファイル生成ツール (tenes_simple, tenes_std) に必要なもの
 - Python3
 - モジュールとして numpy, scipy, toml が必要

TeNeS の使い方 -- インストール

- TeNeS のダウンロード
 - <https://github.com/issp-center-dev/TeNeS>
 - リリースページから適当なバージョンのtar をダウンロード・展開
 - もしくは git clone
- TeNeS のインストール

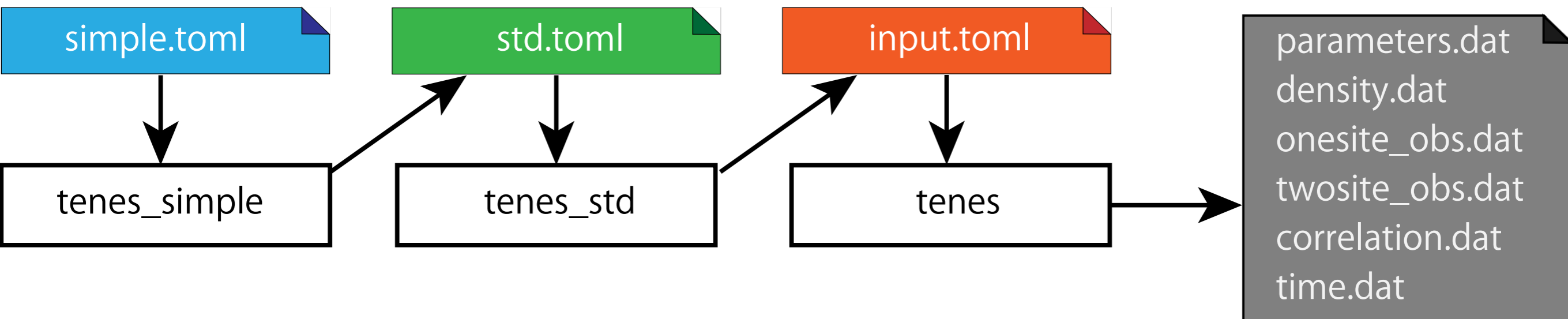
```
$ cd TeNeS          作業ディレクトリを作成・移動
$ mkdir build && cd build          コンパイラの指定
$ cmake -DCMAKE_CXX_COMPILER=`which icpc` \↵
      -DCMAKE_INSTALL_PREFIX=$HOME/opt/tenes \↵
      ../          インストール先の指定
$ make install
```

- \$HOME/opt/tenes/bin に tenes, tenes_std, tenes_simple
- \$HOME/opt/tenes/share 以下にサンプルファイル

TeNeS のつかいかた・基本編

シンプルモード (tenes_simple / simple.toml) を用いた
定義済み模型・格子の計算

TeNeS の使い方 -- 実行フロー



- `tenes_simple` (`simple.toml` → `std.toml`)
 - 定義済み模型・格子のパラメータからハミルトニアンやユニットセル情報を生成するツール
- `tenes_std` (`std.toml` → `input.toml`)
 - ハミルトニアンやユニットセル情報からITE テンソルを生成するツール
 - 長距離ITE テンソルの分解も行われる
- `tenes` (`input.toml` → `output`)
 - 実際の計算を行うメインプログラム

TeNeS の使い方 -- シンプルモード

- 定義済みの模型・格子を計算する
 - テキストベースの単純な入力ファイル1つを用意すれば良い

例：正方格子 $S=1/2$ 反強磁性ハイゼンベルグ模型

SU は $\tau=0.01$ を1000ステップ	<pre>[parameter] [parameter.simple_update] tau = 0.01 # SU の虚時間刻み num_step = 1000 # SU のステップ数</pre>
FU は行わない (0 ステップ)	<pre>[parameter.full_update] tau = 0.01 # FU の虚時間刻み num_step = 0 # FU のステップ数</pre>
角転送行列を使って縮約 角転送行列のボンド次元は16	<pre>[parameter.ctm] meanfield_ctm = false # 縮約のしかた dimension = 16 # ボンド次元 χ</pre>
正方格子 ユニットセルは 2×2 ボンド次元は4	<pre>[lattice] type = "square lattice" # 正方格子 L = 2 # ユニットセルの長さ W = 2 # ユニットセルの幅 initial = "antiferro" # 初期状態 virtual_dim = 4 # ボンド次元 D</pre>
スピン系 S=1/2 (デフォルト) 反強磁性ハイゼンベルグ	<pre>[model] type = "spin" # スピン模型 J = 1.0 # 交換相互作用</pre>

例:S=1/2 正方格子反強磁性ハイゼンベルグ模型

```
$ tenes_simple simple.toml # convert to std.toml
$ tenes_std std.toml      # convert to input.toml
$ tenes input.toml        # perform calculation
... 進捗報告 (省略) ...
```

Onesite observables per site:

```
Sz      = 5.70170299863e-12 8.24461048345e-19
Sx      = -1.93698629873e-08 -8.53148278222e-18
Sy      = 3.12611410373e-08 -4.29087879573e-18
```

Twosite observables per site:

```
hamiltonian = -0.667463006716 2.74670113479e-17       $E_{\text{QMC}}/N = -0.6694422$ 
SzSz        = -0.345269357812 -1.31655470664e-17
SxSx        = -0.161096823276 -4.72116684558e-18
SySy        = -0.161096825628 9.13209512715e-18
```

Save elapsed times to output/time.dat

Wall times [sec.]:

```
all          = 28.396518775
simple update = 7.993189097
full update  = 0
environment  = 18.26470468
observable   = 2.068691087
```

output ディレクトリには
サイトごとの物理量など、
もう少し詳細な出力ファイルが生成される

Done.

TeNeS の使い方 -- シンプルモード

- 定義済みの模型・格子を計算する
 - 模型
 - スピン模型 `model.type = "spin"`
 - J , J_x , J_y , J_z , h_z , h_x , ...
 - Bose-Hubbard 模型 `model.type = "boson"`
 - t , U , V , μ , ...
 - 格子
 - 正方格子 `lattice.type = "square"`
 - 三角格子 `lattice.type = "triangular"`
 - 蜂の巣格子 `lattice.type = "honeycomb"`
 - カゴメ格子 `lattice.type = "kagome"`
- その他各種パラメータについてはマニュアルを参照
 - <https://issp-center-dev.github.io/TeNeS/manual/master/ja/html/index.html>

TeNeS の使い方 -- シンプルモード

- 計算結果は `output` ディレクトリに出力される
 - `parameter.general.output` で変更可能
- `onsite_obs.dat`
 - 一体演算子の期待値 (サイトごと)
- `twosite_obs.dat`
 - 二体演算子の期待値 (ボンドごと)
- `density.dat`
 - 上記期待値のサイト平均
- `correlation.dat`
 - 相関関数
- `correlation_length.dat`
 - 相関長

TeNeS の使い方 -- シンプルモード

- oneseite_obs.dat
 - 一体演算子の期待値 (サイト毎)
 - 1列目: 演算子番号
 - std.toml 中の observable.onesite セクションの group 番号
 - group=-1 は波動関数のノルム $\langle \Psi | \Psi \rangle$
 - 2列目: サイト番号
 - 3(4)列目: 期待値の実部 (虚部)
- スピン系
 - $\langle S_i^z \rangle$ (group=0), $\langle S_i^x \rangle$ (group=1), $\langle S_i^y \rangle$ (group=2, テンソルが複素の時)
- ボース系
 - $\langle n_i \rangle$ (group=0), $\langle b_i^\dagger \rangle$ (group=1), $\langle b_i \rangle$ (group=2)

```
0 0 8.05351406015284822e-09 0.000000000000000000e+00
0 1 8.05351251325784800e-09 0.000000000000000000e+00
0 2 8.05351273424285110e-09 0.000000000000000000e+00
0 3 8.05351179505659039e-09 0.000000000000000000e+00
1 0 4.92509565166792285e-01 0.000000000000000000e+00
1 1 4.92509565166792229e-01 0.000000000000000000e+00
... continue ...
```

TeNeS の使い方 -- シンプルモード

- twosite_obs.dat
 - 二体演算子の期待値 (ボンド=サイト対(source, target))
 - 1列目: 演算子番号
 - std.toml 中の observable.twosite セクションのgroup 番号
 - group=-1 は波動関数のノルム $\langle \Psi | \Psi \rangle$
 - 2列目: source サイト番号
 - 3(4)列目: source サイトから見た target サイトの x(y) 座標
 - TeNeS ではサイトは正方格子として並んでいることに注意
 - 5(6)列目: 期待値の実部 (虚部)
 - スピン系
 - ハミルトニアン (group=0), $\langle S_i^z S_j^z \rangle$ (group=1), $\langle S_i^x S_j^x \rangle$ (group=2), $\langle S_i^y S_j^y \rangle$ (group=3)
 - ボース系
 - ハミルトニアン (group=0), $\langle N_i N_j \rangle$ (group=1), $\langle b_i^\dagger b_j \rangle$ (group = 2), $\langle b_i b_j^\dagger \rangle$ (group=3)
- ```
0 0 0 1 -7.60844868443655509e-01 0.00000000000000000000e+00
0 0 1 0 -7.60253988759186927e-01 0.00000000000000000000e+00
... continue ...
```

# TeNeS の使い方 -- シンプルモード

- density.dat
  - oneseite\_obs, twosite\_obs にある各種演算子のサイト平均
  - tenes を実行したときの標準出力にも出てくる

```
Sz = 8.05351277567753402e-09 0.00000000000000000000e+00
Sx = 4.92509565166792285e-01 0.00000000000000000000e+00
hamiltonian = -1.52140952584773581e+00 0.00000000000000000000e+00
SzSz = 4.38808303473585889e-02 0.00000000000000000000e+00
SxSx = 4.88706591604535612e-01 0.00000000000000000000e+00
SySy = -4.07090179083210504e-02 0.00000000000000000000e+00
```



# TeNeS の使い方 -- シンプルモード

- correlation.dat
  - 2つの1サイト演算子A,B の相関関数  $C(r) = \langle A(r_0)B(r+r_0) \rangle$
  - r は正方格子上の x, y 方向に沿った方向のみ計算可能
  - 入力で correlation セクションが定義されていない場合は計算しない
  - 1列目：Aの演算子番号
  - 2列目：Aのサイト番号(r0)
  - 3列目：Bの演算子番号
  - 4(5)列目：rのx(y)座標
  - 6(7)列目：期待値の実部 (虚部)

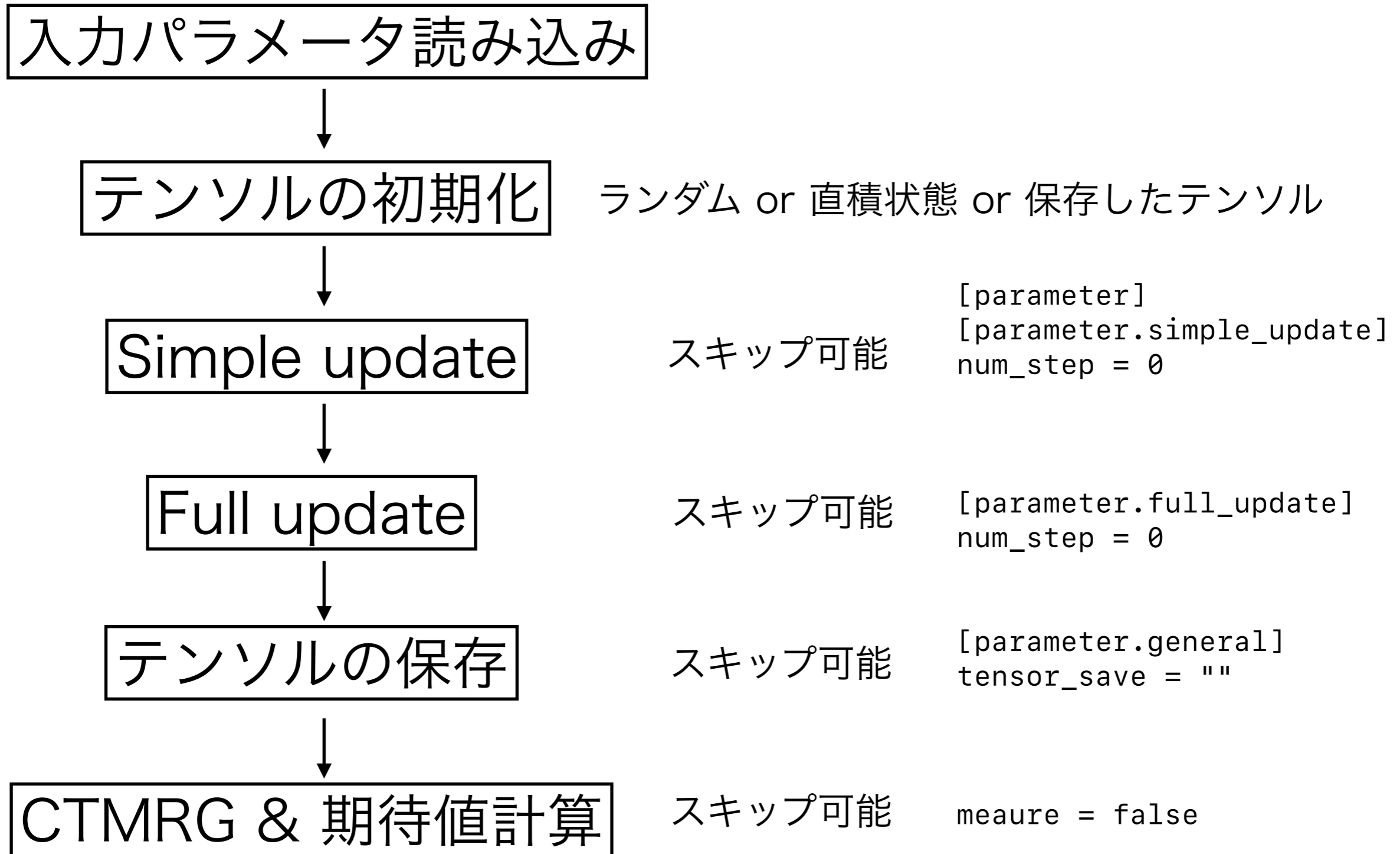
```
0 0 0 1 0 2.14896410089983579e-02 0.00000000000000000000e+00
0 0 0 2 0 2.93084989587023143e-03 0.00000000000000000000e+00
0 0 0 3 0 4.05560655447166879e-04 0.00000000000000000000e+00
... continue ...
```

# TeNeS の使い方 -- シンプルモード

- correlation\_length.dat
  - (バルクテンソルの) 相関長
    - 転送行列の固有値から計算される
  - 1列目：方向 (0: +x, 1: +y)
  - 2列目：y座標 or x座標
  - 3列目：相関長  $\xi = 1/e_1$
  - 4- 列目：固有値 $t_i$ の比  $e_i = -\log |t_i/t_0|$  ( $t_0 \geq t_1 \geq \dots$ )

```
0 0 5.47352233294713275e-01 1.82697710755765863e+00 1.88536376356653368e+00
0 1 5.47350455079123721e-01 1.82698304298558090e+00 1.88539026557094735e+00
1 0 5.49500905928057670e-01 1.81983321448959168e+00 1.87779077592520527e+00
1 1 5.49502704153696175e-01 1.81982725915812682e+00 1.87776424815483045e+00
```

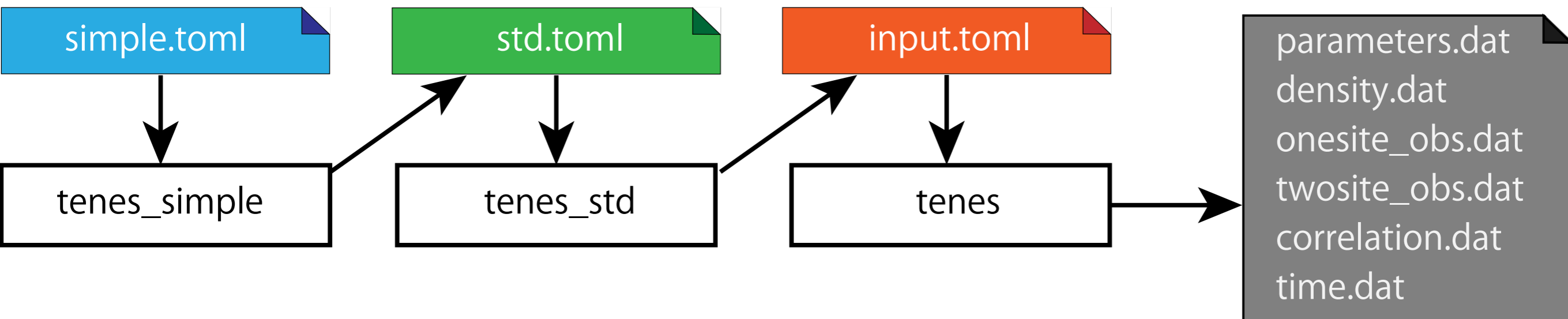
# tenes (本体プログラム) の内部フロー



TeNeS のつかいかた・発展編

スタンダードモード (tenes\_std/std.toml) を用いた  
格子・模型・演算子の定義

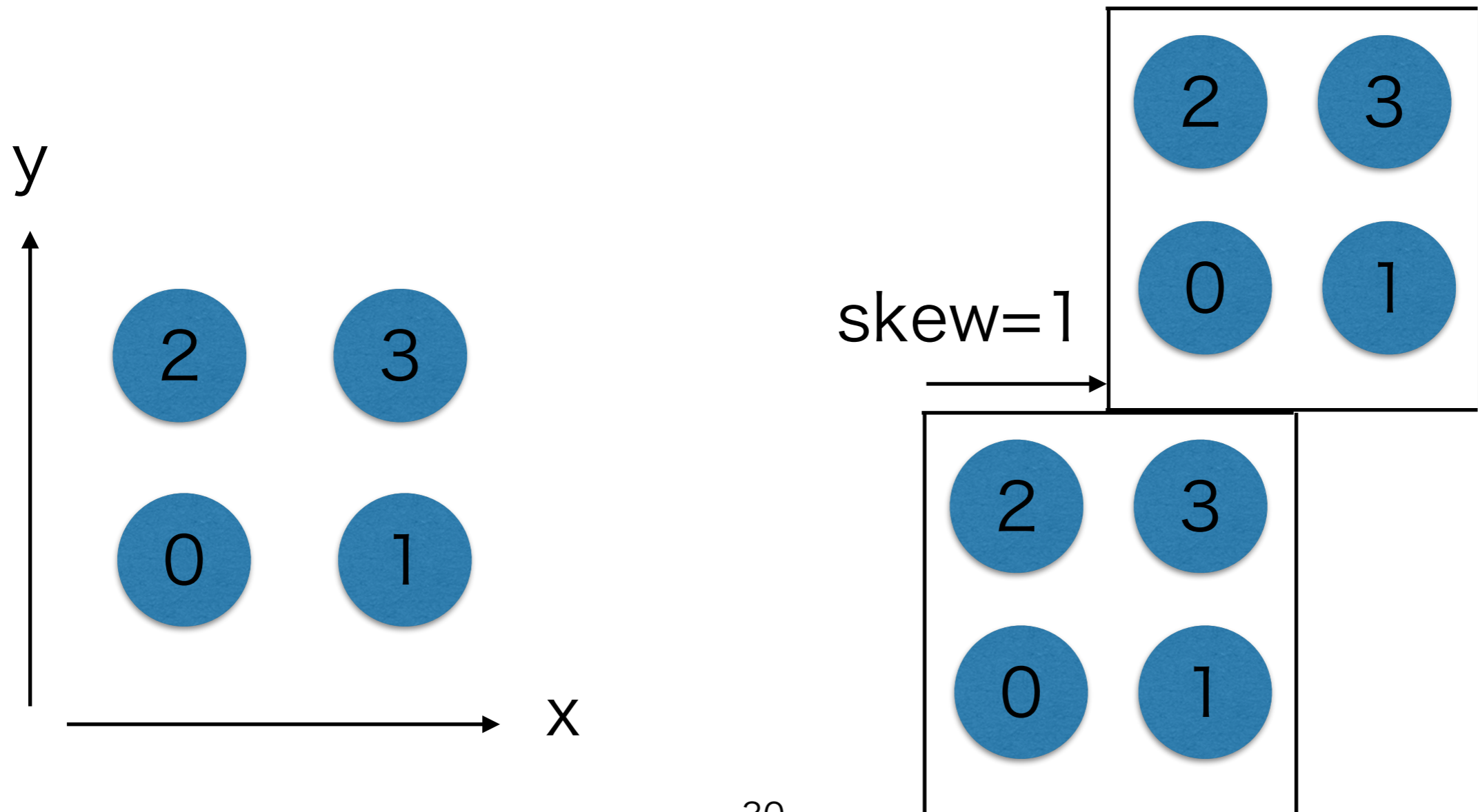
# TeNeS の使い方 -- 実行フロー



- `tenes_simple` (`simple.toml` → `std.toml`)
  - 定義済み模型・格子のパラメータからハミルトニアンやユニットセル情報を生成するツール
- `tenes_std` (`std.toml` → `input.toml`)
  - ハミルトニアンやユニットセル情報からITE テンソルを生成するツール
  - 長距離ITE テンソルの分解も行われる
- `tenes` (`input.toml` → `output`)
  - 実際の計算を行うメインプログラム

# standard mode: ユニットセル

```
[tensor]
type = "square lattice" # 無視される (simple.toml の名残)
L_sub = [2, 2] # 2x2 unitcell
skew = 0 # y 方向の境界を超えたときの x 方向のずれ
```



# standard mode: ユニットセル

```
[[tensor.unitcell]]
virtual_dim = [4, 4, 4, 4] # ボンド次元 (← ↑ → ↓ の順)
index = [0, 3] # ユニットセル中のどのテンソルかを示す番号
physical_dim = 2 # 物理ボンドの次元
initial_state = [1.0, 0.0] # 初期状態の係数
noise = 0.01 # 初期テンソルのゆらぎ
```

```
[[tensor.unitcell]]
virtual_dim = [4, 4, 4, 4]
index = [1, 2]
physical_dim = 2
initial_state = [0.0, 1.0]
noise = 0.01
```

全体の初期状態  $\psi$  はサイトごとの初期状態  $\psi_i$  の直積状態  $|\Psi\rangle = \otimes_i |\Psi_i\rangle$

$\psi_i$  は、initial\_state 配列の要素を前から  $a_0, a_1, \dots, a_{d-1}$  とすると、

$$|\Psi_i\rangle \propto \sum_k^{d-1} a_k |k\rangle$$

# standard mode: ハミルトニアン

TeNeS はハミルトニアンをサイトハミルトニアン (1サイトハミルトニアン) と  
ボンドハミルトニアン (2サイトハミルトニアン) の和として考える

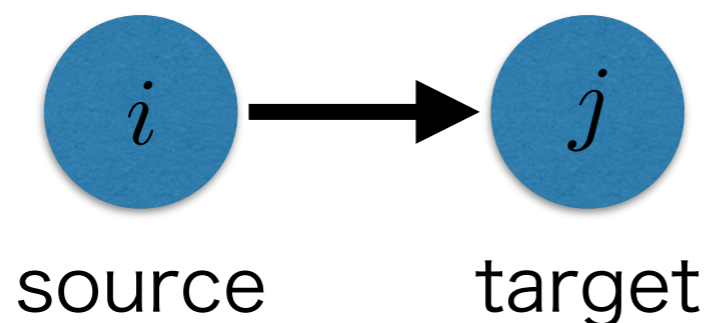
$$\mathcal{H} = \sum_{ij} \mathcal{H}_{ij} + \sum_i \mathcal{H}_i$$

これら局所ハミルトニアンは、その行列要素と作用するサイト・ボンドで規定する

行列要素を定義すれば模型を定義できる

ボンドを定義すれば格子を定義できる

ボンドは source サイトと target サイトの組であると考え





# standard mode: ボンドハミルトニアン

std.toml でのボンドハミルトニアンの定義

```
[[hamiltonian]]
dim = [2, 2] # 作用するボンド [source, target] の取りうる状態数の対
bonds = """ # 作用するボンドの集合 (1行1ボンド)
0 1 0 # 1列目: ユニットセル内の source の番号
1 1 0 # 2列目: source からみた target の x 座標 (変位)
2 1 0 # 3列目: source からみた target の y 座標 (変位)
3 1 0
0 0 1
1 0 1
2 0 1
3 0 1
"""
```

▶ 次のページ

```
elements = """ # ハミルトニアンの (非ゼロな) 行列要素 (1行1要素)
0 0 0 0 0.25 0.0 # 1列目: 作用前の source の状態
1 0 1 0 -0.25 0.0 # 2列目: 作用前の target の状態
0 1 1 0 0.5 0.0 # 3列目: 作用後の source の状態
1 0 0 1 0.5 0.0 # 4列目: 作用後の target の状態
0 1 0 1 -0.25 0.0 # 5列目: 要素の実部
1 1 1 1 0.25 0.0 # 6列目: 要素の虚部
"""
```

▶ 次の次のページ

$$\vec{S}_i \cdot \vec{S}_j$$

# standard mode: ボンドハミルトニアン

ボンドハミルトニアンの作用するボンドの定義

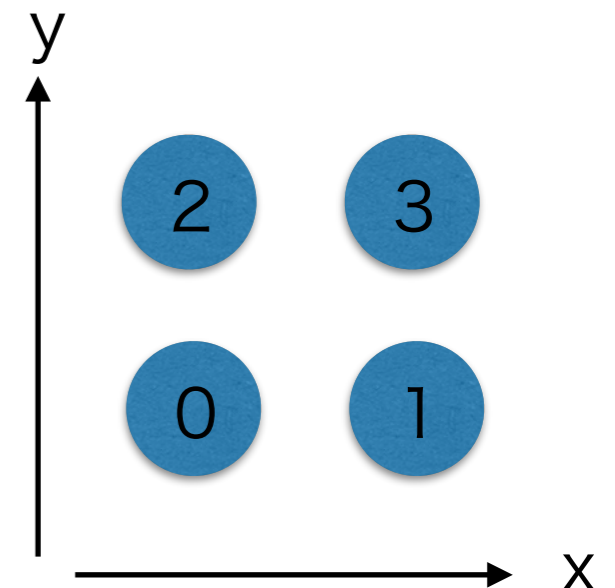
```
[[hamiltonian]]
dim = [2, 2] # 作用するボンド [source, target] の取りうる状態数の対
bonds = "" # 作用するボンドの集合 (1行1ボンド)
0 1 0 # 1列目: ユニットセル内の source の番号
1 1 0 # 2列目: source からみた target の x 座標 (変位)
2 1 0 # 3列目: source からみた target の y 座標 (変位)
3 1 0
0 0 1
1 0 1
2 0 1
3 0 1
""
```

右のユニットセルの時

0 1 0 は 0 番と右隣(1) ( $x+=1, y+=0$ )

1 0 1 は 1 番と上隣(3) ( $x+=0, y+=1$ )

1 1 0 は 1 番と右隣 (右のセルの0)



# standard mode: ボンドハミルトニアン

ボンドハミルトニアン演算子の行列要素の定義

|                          |   | elements | = | " " " |
|--------------------------|---|----------|---|-------|
| 1列目：作用前の source site の状態 | → | <u>0</u> |   | 0.25  |
| 2列目：作用前の target site の状態 | → | <u>1</u> |   | -0.25 |
| 3列目：作用後の source site の状態 | → | <u>0</u> |   | 0.5   |
| 4列目：作用後の target site の状態 | → | <u>1</u> |   | 0.5   |
| 5列目：要素の実部                | → | <u>0</u> |   | -0.25 |
| 6列目：要素の虚部                | → | <u>1</u> |   | 0.25  |
|                          |   | 0 0 0 0  |   | 0.0   |
|                          |   | 1 0 1 0  |   | 0.0   |
|                          |   | 0 1 1 0  |   | 0.0   |
|                          |   | 1 0 0 1  |   | 0.0   |
|                          |   | 0 1 0 1  |   | 0.0   |
|                          |   | 1 1 1 1  |   | 0.0   |
|                          |   | " " "    |   | " " " |

例

$$\underline{0\ 0\ 0\ 0\ 0.25\ 0.0} \text{ は } \langle 00 | \mathcal{H}_b | 00 \rangle = 0.25$$

$$\underline{0\ 1\ 1\ 0\ 0.5\ 0.0} \text{ は } \langle 10 | \mathcal{H}_b | 01 \rangle = 0.5$$

# standard mode: サイトハミルトニアン

std.toml でのサイトハミルトニアンの定義

```
[[hamiltonian]]
dim = 2 # 作用するサイトの取りうる状態数
sites = [] # 作用するサイトのリスト（空リストは全サイト）

elements = """ # ハミルトニアンの（非ゼロな）行列要素（1行1要素）
0 0 -0.5 0.0 # 1,2 列目: 作用する前後の状態
1 1 0.5 0.0 # 3,4 列目: 行列要素
"""
```

$$-S_i^z$$

行列要素の定義方法はボンドハミルトニアンの1サイト版

# standard mode: 演算子

最終的に期待値を計算する演算子の定義

現在は1サイト演算子と2サイト演算子を計算可能

エネルギー演算子 = ボンドハミルトニアンも改めて指定する必要がある  
(tenes\_std が0番の演算子として自動でコピーしてくれる)

```
[observable]
[[observable.onesite]] # 1サイト演算子
name = "Sz" # 名前
group = 1 # 1サイト演算子の識別番号
sites = [] # 1サイト演算子が作用するテンソルの番号 ([] はすべてを意味する)
dim = 2 # 1サイト演算子の次元
elements = "" # 1サイト演算子行列の非ゼロ要素 (1行1要素)
0 0 0.5 0.0
1 1 -0.5 0.0
""
```

$$S^z = \begin{pmatrix} 0.5 & 0.0 \\ 0.0 & -0.5 \end{pmatrix}$$

# standard mode: 演算子

最終的に期待値を計算する演算子の定義

現在は1サイト演算子と2サイト演算子を計算可能

エネルギー演算子 = ボンドハミルトニアンも改めて指定する必要がある  
(tenes\_std が0 番の演算子として自動でコピーしてくれる)

```
[[observable.twosite]]
name = "SzSz" # 名前
group = 1 # 2サイト演算子の識別番号 (1サイトとは独立)
dim = [2, 2] # 次元
bonds = "" # 作用するボンド (サイト対)
0 1 0 # ボンドハミルトニアンと同様の書式
1 1 0
2 1 0
3 1 0
0 0 1
1 0 1
2 0 1
3 0 1
""
ops = [1, 1] # 1サイト演算子の直積で書ける場合、その識別番号
 # 今回は "Sz" が1 番の1サイト演算子
 # elements として行列要素を陽に書くことも可能
 # (ボンドハミルトニアンと同様の書式)
```

$$S_i^z S_j^z$$

# standard mode: まとめ

- `tenes_std` の入力ファイル (`std.toml`) を編集すると格子・模型を自作可能
  - ユニットセル (正方格子上に並ぶバルクテンソル)
  - ハミルトニアン (サイト・ボンド)
    - ボンドハミルトニアンのつながりで具体的な格子が定義される
- まずは `tenes_simple` の出力をいじるのがよい
  - 磁場項などの一体項はサイトハミルトニアンではなくボンドハミルトニアンに吸収されて出力される
    - `--use-site-hamiltonian` オプションをつけるとサイトハミルトニアンとして出力される