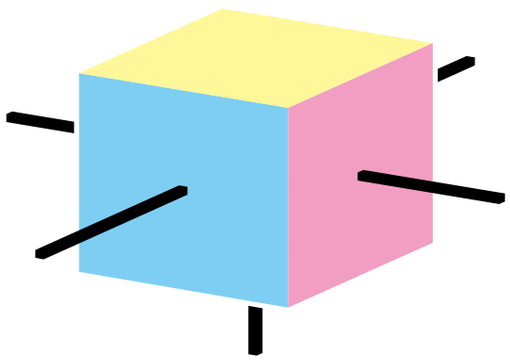


TeNeS

Tensor Network Solver for Quantum Lattice Systems

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

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

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

Yuichi Motoyama (ISSP)

2020-11-10
for TeNeS ver 1.1.2

TeNeS 開発チーム

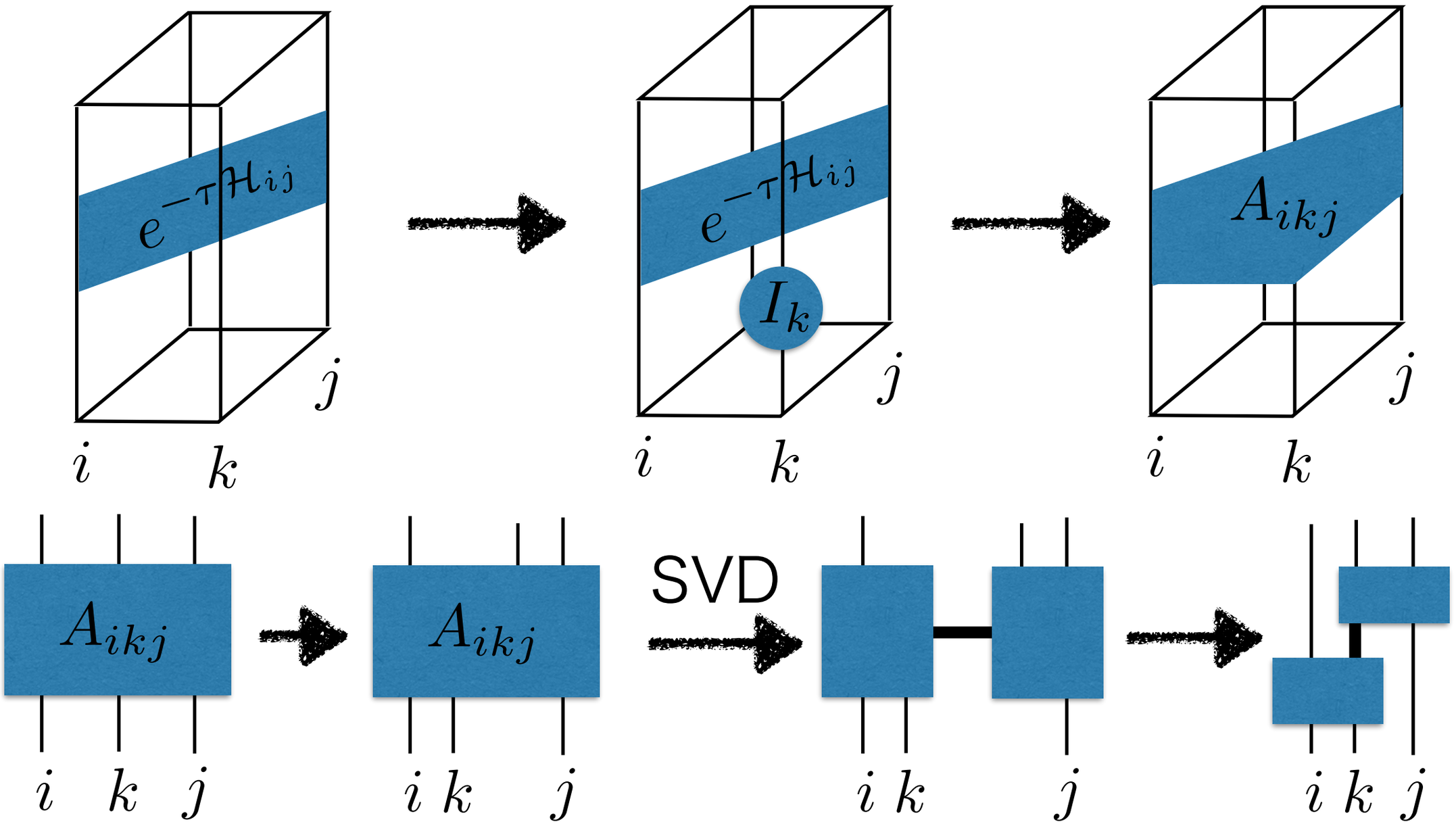
- **大久保毅**（東大院理）
  - アルゴリズム部分の実装
- **森田悟史**（東大物性研）
  - 関連ライブラリ・ツール作成（mptensor, tensordot）
- **本山裕一**（東大物性研）
  - メインプログラム・入力ファイル作成ツールの設計・実装
- **吉見一慶**（東大物性研）
  - ユーザテスト、サンプル（自動実行スクリプトなど）の作成、Pj マネージメント
- **加藤岳生**（東大物性研）
  - ユーザテスト、サンプル作成
- **川島直輝**（東大物性研）
  - プロジェクトリーダー

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** 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** 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）

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

    - Bose-Hubbard 模型（粒子数カットオフ付き）

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

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

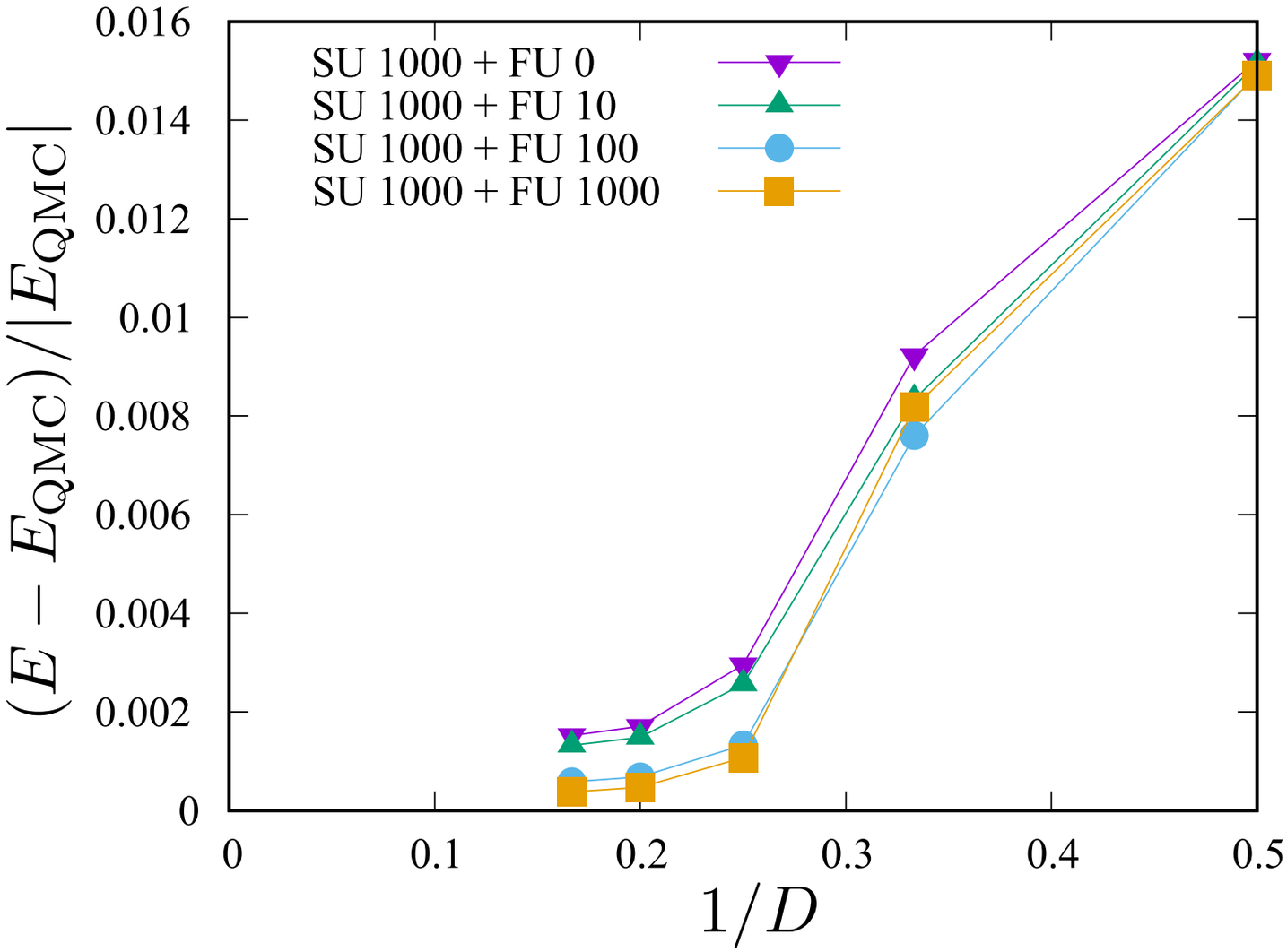
- 量子格子模型におけるテンソルネットワーク法研究への参入障壁を下げる
  - TN 法のアルゴリズム・プログラム開発
    - テンソル演算のライブラリはそれぞれの言語で多数存在
    - アルゴリズムも（ダイアグラム表記のおかげで）割とわかりやすい
    - 実際に1から組み上げるのは結構難しい（つらい）
      - テンソルの添字の順番など、システマティックに構築しないとすぐに混乱する
      - 複数のテンソルを縮約する場合、順番によって計算量オーダーが変わる
  - 計算結果をどう評価するか？
    - ボンド次元など、ハイパーパラメータが結構多い
  - **TeNeS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

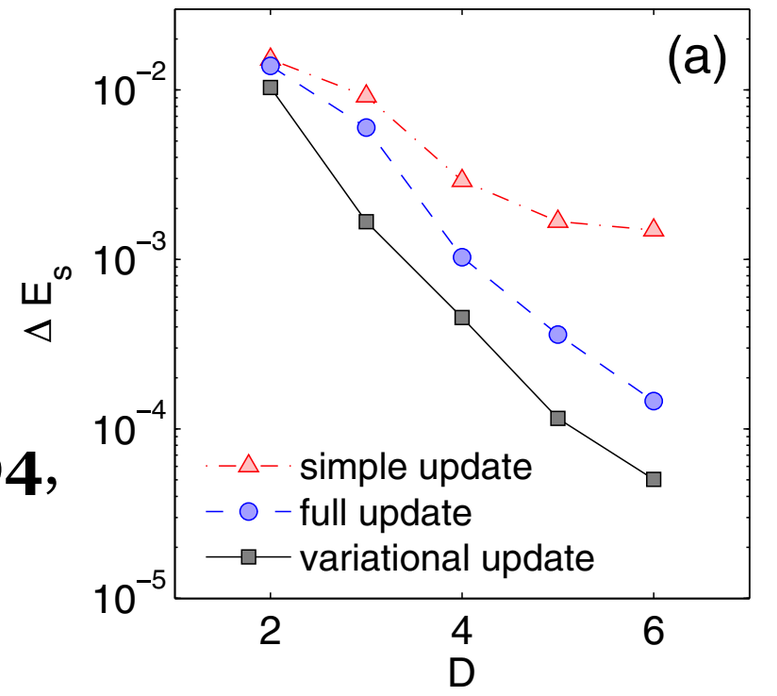
計算例：正方格子S=1/2反強磁性ハイゼンベルグ模型のエネルギー

- 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)

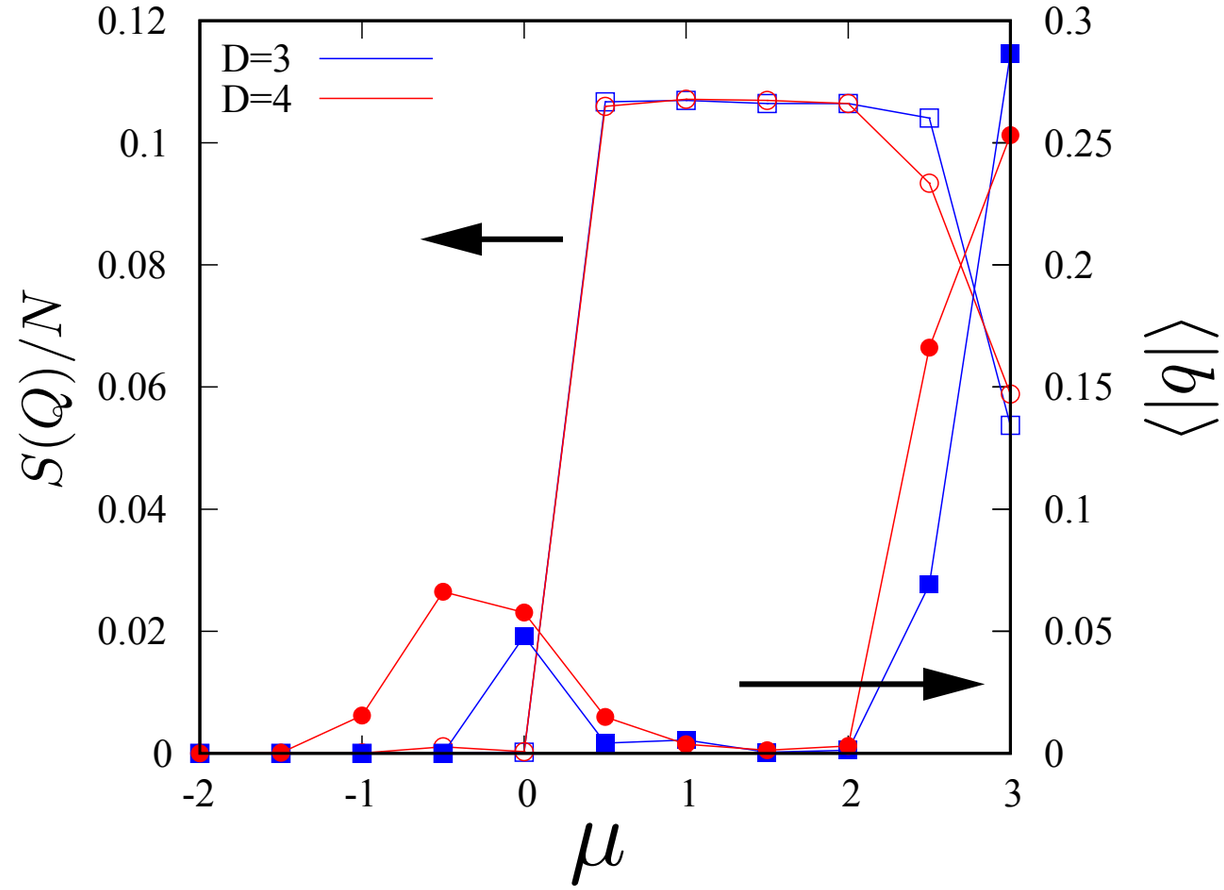
- ユニットセルの大きさは2×2
- $\chi = D^2$
- 24CPU の計算機で数時間程度

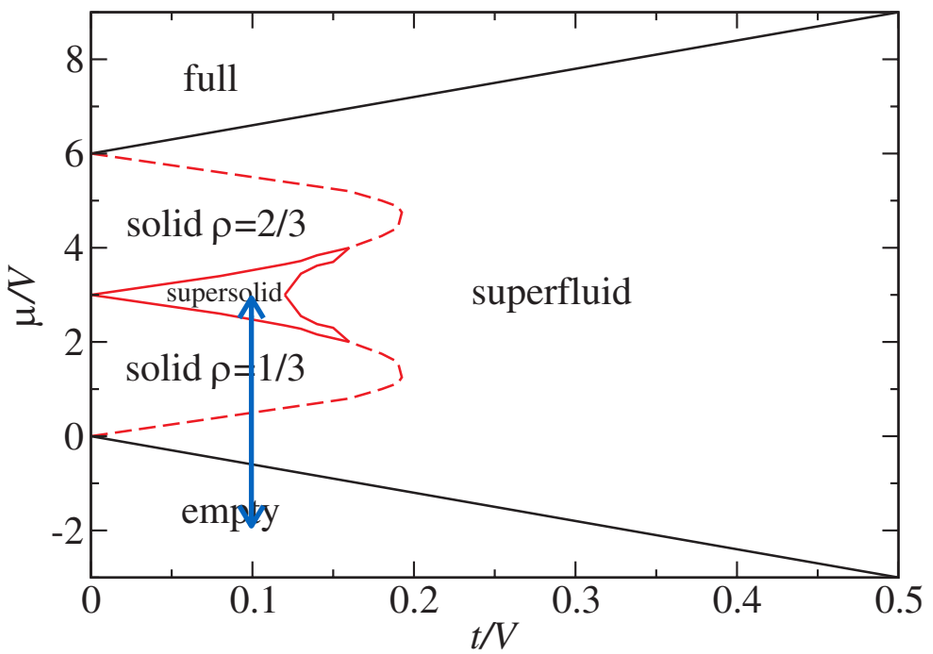


P. Corboz, PRB **94**, 035133 (2016)

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√3×√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)

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
$ cmake -DCMAKE_CXX_COMPILER=`which icpc` \↵          コンパイラの指定
        -DCMAKE_INSTALL_PREFIX=$HOME/opt/tenes \↵
        ../                                  インストール先の指定
$ make install
```

- $HOME/opt/tenes/bin に実行ファイル 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)を用いた定義済み模型・格子の計算

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τ=0.01 を1000ステップ

FU は行わない (0 ステップ)

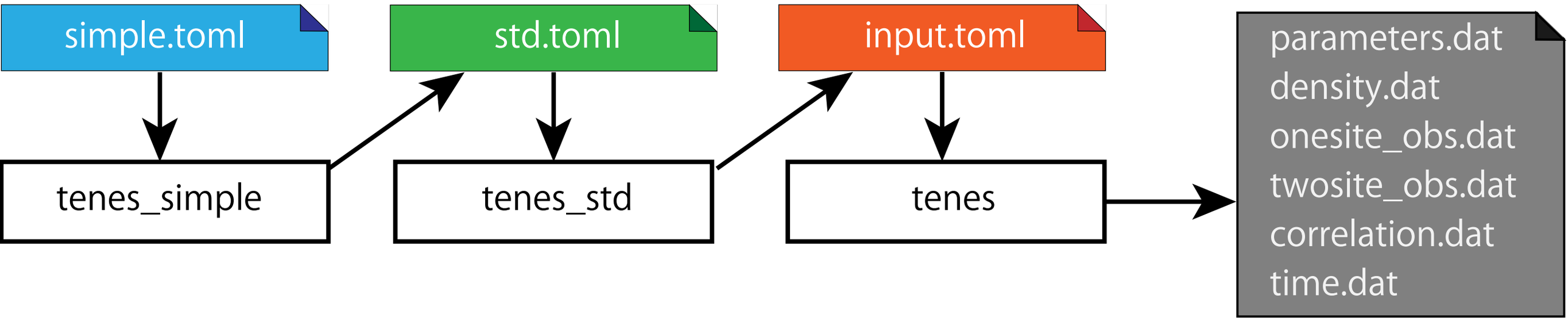
角転送行列のボンド次元は16

正方格子
ユニットセルは2x2
ボンド次元は4

スピン系
S=1/2 （デフォルト）
反強磁性ハイゼンベルグ

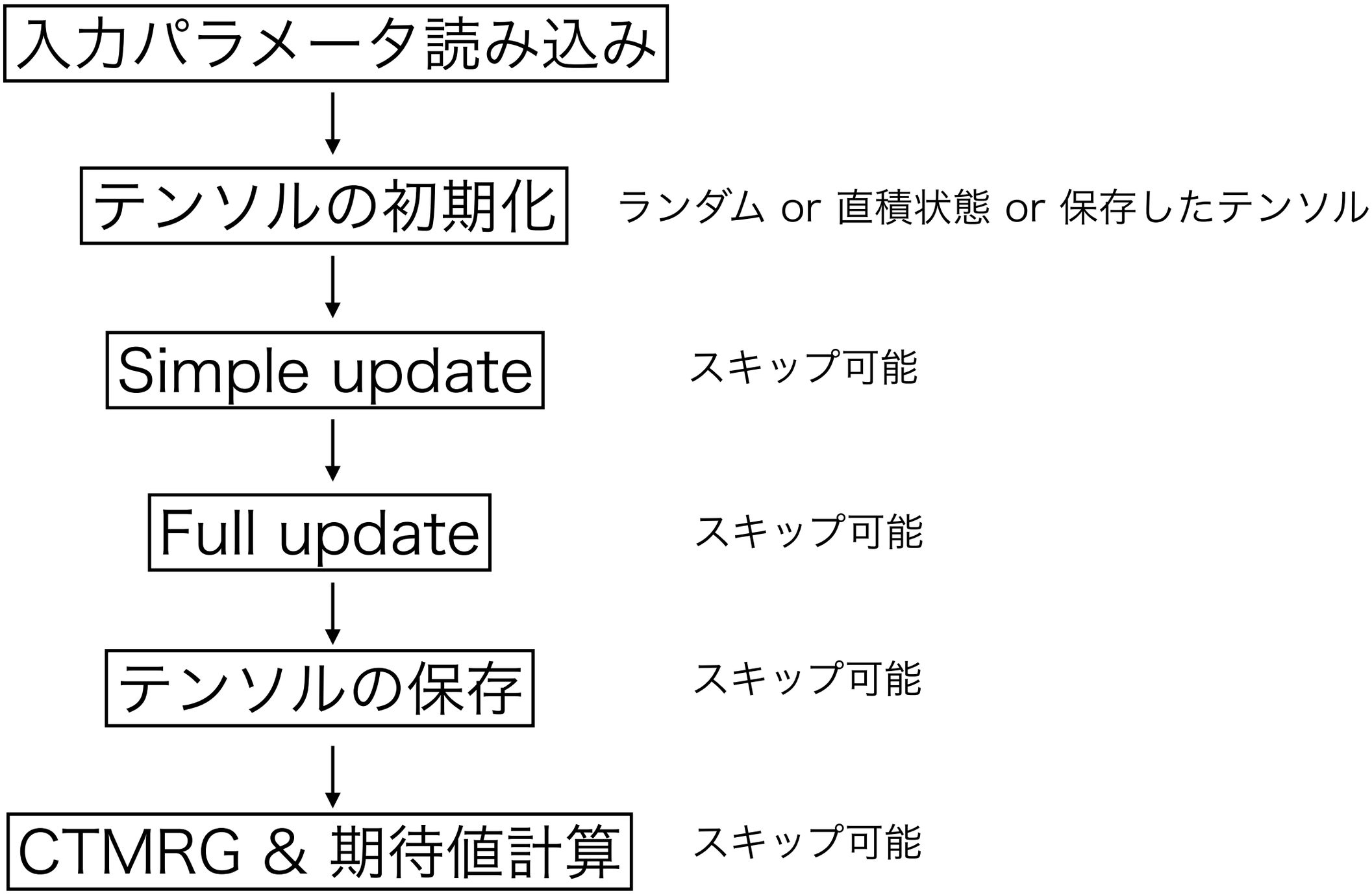
```
[parameter]
[parameter.simple_update]
tau = 0.01        # SU の虚時間刻み
num_step = 1000   # SU のステップ数

[parameter.full_update]
tau = 0.01        # FU の虚時間刻み
num_step = 0      # FU のステップ数

[parameter.ctm]
dimension = 16    # ボンド次元 χ

[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

[model]
type = "spin"     # スピン模型
J = 1.0           # 交換相互作用
```

TeNeS の使い方 -- 実行フロー



- tenes_simple
  - 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
  - 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
  - 実際の計算を行うメインプログラム

tenes（本体プログラム）の内部フロー



例:S=1/2 正方格子反強磁性ハイゼンベルグ模型

```
$ $HOME/opt/tenes/bin/tenes_simple simple.toml  # convert to std.toml
$ $HOME/opt/tenes/bin/tenes_std std.toml        # convert to input.toml
$ $HOME/opt/tenes/bin/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
  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
  environmnent  = 18.26470468
  observable    = 2.068691087

Done.
```

$E_{\mathrm{QMC}}/N = -0.6694422$

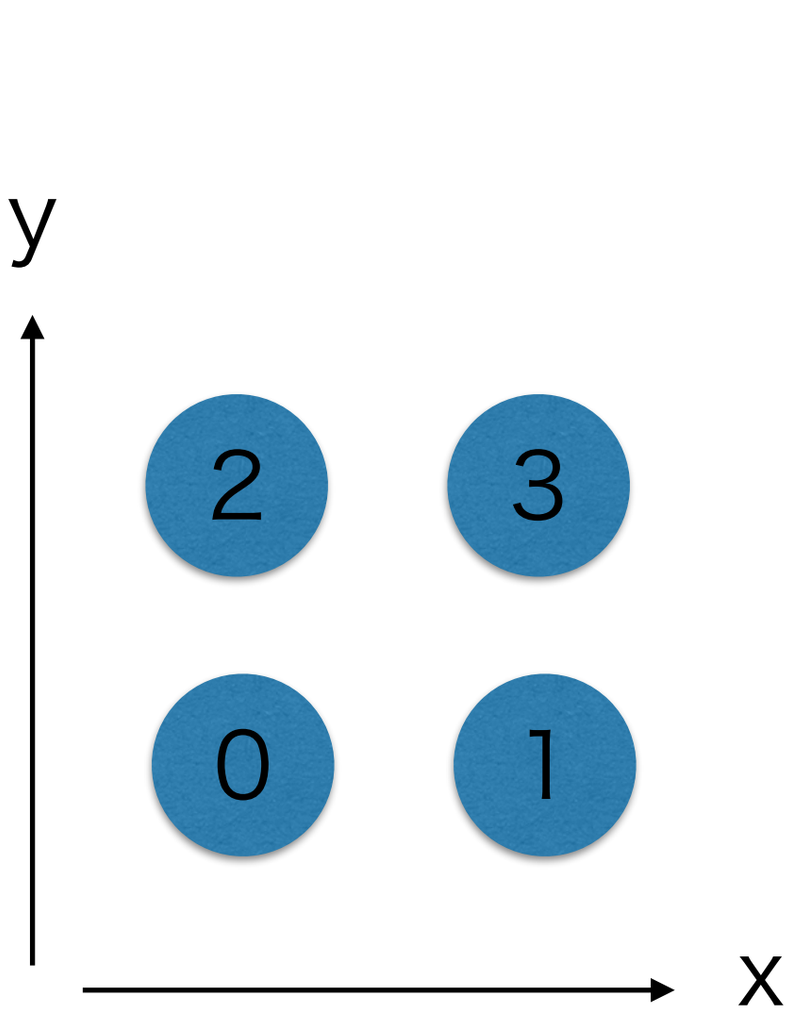
output ディレクトリには
サイトごとの物理量など、
もう少し詳細な出力ファイルが生成される

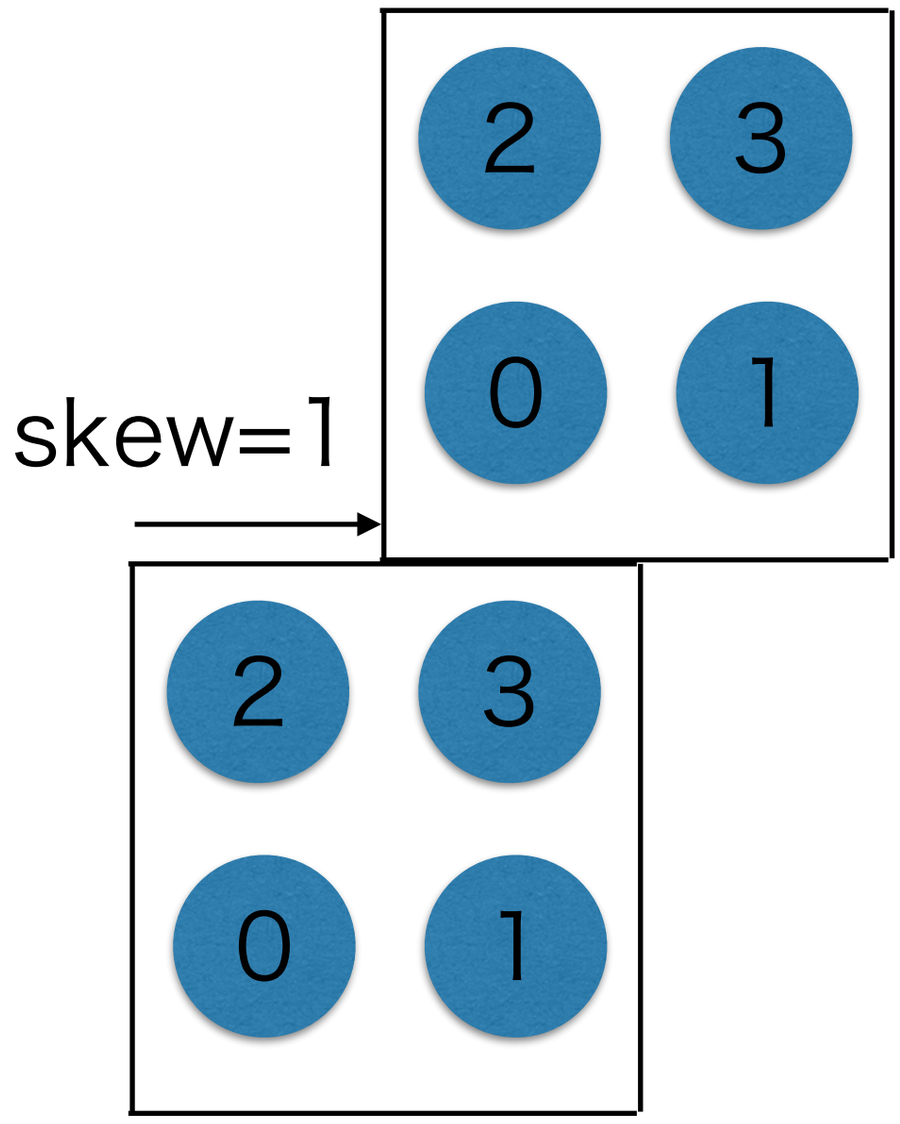
TeNeS のつかいかた・発展編

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

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 ψ はサイトごとの初期状態 ψ_i の直積状態 $\left|\Psi\right\rangle = \bigotimes_i \left|\Psi_i\right\rangle$

ψ_i は、initial_state 配列の要素を前から a_0, a_1, ..., a_{d-1} とすると、

$$\left|\Psi_i\right\rangle \propto \sum_{k}^{d-1} a_k \left|k\right\rangle$$

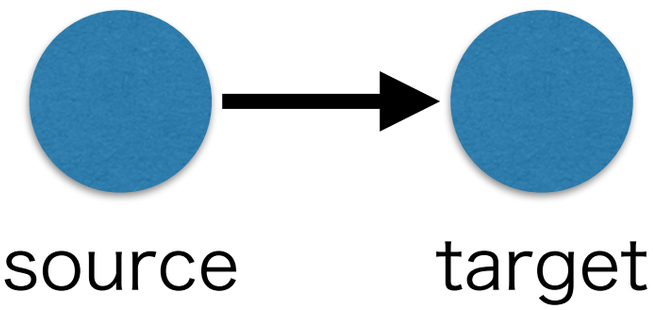
standard mode: ボンドハミルトニアン

TeNeS が扱うハミルトニアンはボンドハミルトニアン（2サイトハミルトニアン）の和

（磁場などのサイトハミルトニアンも近くのボンドハミルトニアンに取り入れる）

$$\mathcal{H} = \sum_{i,j} \mathcal{H}_{ij}$$

ボンドは source サイトと target サイトの組であると考える



ボンドハミルトニアンは、その行列要素と作用するボンドで規定する

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

source, target が同じ番号のテンソルになるのは禁忌

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

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

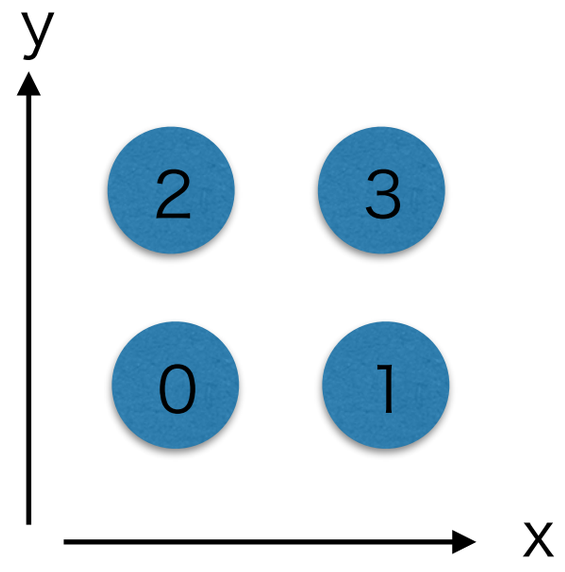
```
[[hamiltonian]]
dim = [2, 2]    # 作用するボンド [source, target] の取りうる状態数の対
bonds = """     # 作用するボンドの集合（1行1ボンド）
0 1 0           # 1列目: ユニットセル内の source の番号
1 1 0           # 2列目: source からみた target の x 座標（変位）
2 1 0           # 3列目: source からみた target の y 座標（変位）
3 1 0
0 0 1
1 0 1
2 0 1
3 0 1
"""
```

右のユニットセルの時

０１０は０番と右隣(1) (x+=1, y+=0)

１０１は１番と上隣(3) (x+=0, y+=1)

１１０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 = """       # 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
"""
```

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

0 1 1 0 0.25 0.0 は　$\langle 10 \,|\, \mathcal{H}_b \,|\, 01 \rangle = 0.5$

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2サイト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 = 0        # 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      # 1,2列目: 作用前後の状態
1 1 -0.5 0.0     # 3,4列目: 要素の実部・虚部
"""
```

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

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2サイト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 = [0, 0]     # 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0 番の1サイト演算子
    S_i^z S_j^z  # 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）
```