TeNeS ユーザーマニュアル

リリース ***0.1***

東京大学物性研究所

2019 年 **12** 月 **04** 日

目次

第 1 章

TeNeS とは？

1.1 概要

TeNeS はテンソルネットワーク法に基づく多体量子状態計算のためのオープンソースのプログラムパッケージです。本バージョンでは、格子上で定義された量子スピン模型の基底状態の波動関数を評価することができ、磁化や相関関数などの物理量を評価できます。あらかじめ定義された模型・格子に対しては、ユーザーが簡単に入力ファイルを作成するためのツールがあり、気軽に体験することができます。バージョン 1.0 では、ユーザーが簡単な入力ファイルから模型・格子を自分で定義できるツールの作成と、ボソン系への拡張も行う予定です。これにより、様々な二次元量子スピン、ボソン系を取り扱うことができるようになる予定です。

1.2 開発者

TeNeS は以下のメンバーで開発しています。

- ver 0.1
  - 大久保 毅 (東京大学大学院 理学系研究科)
  - 森田 悟史 (東京大学 物性研究所)
  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 加藤 岳生 (東京大学 物性研究所)
  - 川島 直輝 (東京大学 物性研究所)

1.3 バージョン履歴

- ver. 0.1: 2019/12/04 にリリース。

1.4 ライセンス

本ソフトウェアのプログラムパッケージおよびソースコード一式は GNU General Public License version 3（GPL v3）に準じて配布されています。

1.5 コピーライト

本ソフトウェアの一部は 2019 年度 東京大学物性研究所 ソフトウェア高度化プロジェクトの支援を受け開発されており、その著作権は東京大学が所持しています。

第 2 章

インストール方法

2.1 ダウンロード

TeNeS のソースコードは GitHub page からダウンロードできます。git がインストールされている環境で、以下のコマンドを打つとダウンロードが開始されます。

```
$ git clone https://github.com/issp-center-dev/TeNeS
```

2.2 必要なライブラリ・環境

TeNeS をコンパイルするには以下のライブラリ・環境が必要です。

1. C++11 compiler
2. CMake (>=2.8.14)
3. MPI と ScaLAPACK

TeNeS は以下のライブラリに依存していますが、自動でダウンロードおよびビルドがされます。

1. mptensor
2. cpptoml
3. sanitizers-cmake

ScaLAPACK については自身でインストールする必要があります。Mac で homebrew を利用されている方は、

```
brew install scalapack
```

でインストールすることが可能です。それ以外の方は、ScaLAPACK のホームページを参照の上、インストールをしてください。

また、入力ファイル作成ツール `tenes_simple` の使用には以下のライブラリ・環境が必要です。

1. Python (バージョン 3 以上での稼働を確認)
2. numpy
3. toml

2.3 インストール

1. 以下の手順に従ってビルドを行います。

```
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_PREFIX=<path to install to> ../
$ make
```

上記のコマンドで build/src ディレクトリに実行ファイル tests が作成されます。<path to install to> のデフォルト値は /usr/local です。

2. 次にインストールを実行します。

:: $ make install

実行ファイル tenes が <path to install to>/bin にインストールされます。

コンパイラの指定

CMake では自動でコンパイラを検出してビルドを行います。コンパイラを指定したい場合には, 以下のようにオプションを追加してください。

```
$ cmake -DCMAKE_CXX_COMPILER=<path to your compiler> ../
```

Python インタープリタの指定

CMake では自動で Python インタープリタを検出してビルドを行います。指定したい場合には, 以下のようにオプションを追加してください。

```
$ cmake -DPYTHON_EXECUTABLE=<path to your interpreter> ../
```

---

mptensor の指定

ユーザーが事前にインストールした mptensor を使用したい場合には、以下のようにオプションを追加してください。

```
$ cmake -DMPTENSOR_ROOT=<path to mptensor> ../
```

第 3 章

使用方法

TeNeS では事前に所定のフォーマットに従い模型や演算順などを定義するための複数の入力ファイルを作成する必要があります。入力ファイルの作成をしやすいように、

- あらかじめ定義された格子模型に対して, 簡単な入力ファイルから tenes を実行するための入力ファイルを生成するツール tenes_simple.py
- 所定のフォーマットに従い、自分で格子模型などを定義した入力ファイルを作成し実行することで、tenes を実行するための入力ファイルを生成するツール tenes_standard.py (ver. 1.0 で実装予定)

が用意されています。上記以外の模型や格子に対応させたい場合には、 tenes の入力ファイルを直接作成することで対応できます。 tenes の入力ファイルの詳細については、ファイルフォーマット をご覧ください。

以下、それぞれのツールの使用方法について記載し、最後に tenes の使用方法について説明します。

3.1 tenes_simple の使用方法

tenes_simple は定義済みの模型、格子に対する tenes の入力ファイルを生成するツールです。

```
$ ./tenes_simple --help
usage: tenes_simple [-h] [-o OUTPUT] input

Simple input generator for TeNeS

positional arguments:
  input                 Input TOML file

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Output TOML file
```

- 引数として入力ファイル名を取ります。

- コマンドラインオプションは、以下の通りです。
  - help
    - ヘルプメッセージの表示
  - output
    - 出力ファイル名
    - デフォルトは input.toml
    - 入力ファイル名と同じファイル名にすることはできません

3.2 tenes_standard.py の使用方法

tenes_standard.py は ver.1.0 で作成予定です。ver. 1.0 は 2020 年度中の公開を目指し開発中です。

3.3 tenes の使用方法

tenes の実行は以下のように行うことができます。

```
$ tenes --help
TeNeS: TEnsor NEtwork Solver for 2D quantum lattice system

  Usage:
    tenes [--quiet] <input_toml>
    tenes --help
    tenes --version

  Options:
    -h --help       Show this help message.
    -v --version    Show the version.
    -q --quiet      Do not print any messages.
```

- 引数として入力ファイル名を取ります。
- コマンドラインオプションは、以下の通りです。
  - help - ヘルプメッセージの表示
  - version - バージョン情報の表示
  - quiet - 標準出力に何も書き出さないようにします

```
tenes input.toml
```

第 4 章

チュートリアル

4.1 横磁場イジングモデル

ここでは横磁場イジングモデルに対して、横磁場を変化させた場合の計算例について紹介します。入力ファイルの変数 G を用いることで横磁場の大きさを調整することが可能です。例えば、横磁場が 0 の場合には、

```
[parameter]
[parameter.tensor]
D  = 2     # tensor_dim
CHI  = 10  # env_dim

[parameter.simple_update]
num_step = 1000
tau = 0.01

[parameter.ctm]
iteration_max = 10

[lattice]
type = "square lattice"
L_sub = [2,2]

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

とします (Jz = −1.0 なので、 G=0 では強磁性状態になります)。入力ファイルを simple.toml とした場合、

```
$ tenes_simple simple.toml
$ tenes input.toml
```

を実行することで計算が開始されます。計算を実行すると、

```
Start simple update
Start calculating observables
Start updating environment
Start calculating local operators
Save site observables to output/site_obs.dat
Start calculating energy
Save energy to output/energy.dat
Start calculating NN correlation
Save NN correlation to output/neighbor_obs.dat
Save elapsed times to output/time.dat

Energy = -0.5
Local operator 0 = 0.5
Local operator 1 = 1.90794709356e-11

time simple update = 3.21127
time full update   = 0
time environmnent  = 0.875561
time observable    = 0.132412
```

のように計算が実行されます。最初に各プロセスの実行状況が表示されます。計算終了後、 Energy と局在演算子 Local operator 0 (<Sz>), Local operator 1 (<Sx>) がそれぞれ出力されます。最後に time でどの程度計算時間がかかったか出力されます (単位は秒)。計算終了後は output フォルダに energy.dat, parameters.dat, time.dat, neighbor_obs.dat, site_obs.dat がそれぞれ出力されます。各出力ファイルの詳細は、ファイルフォーマットをご覧ください。<Sz> の値は、 site_obs.dat の 0 0 成分もしくは標準出力の Local operator 0 に続く値から抽出することが可能で、G をパラメータとして 0.1 刻みで 0-2.0 まで振り、得られた結果を下図に表示します。

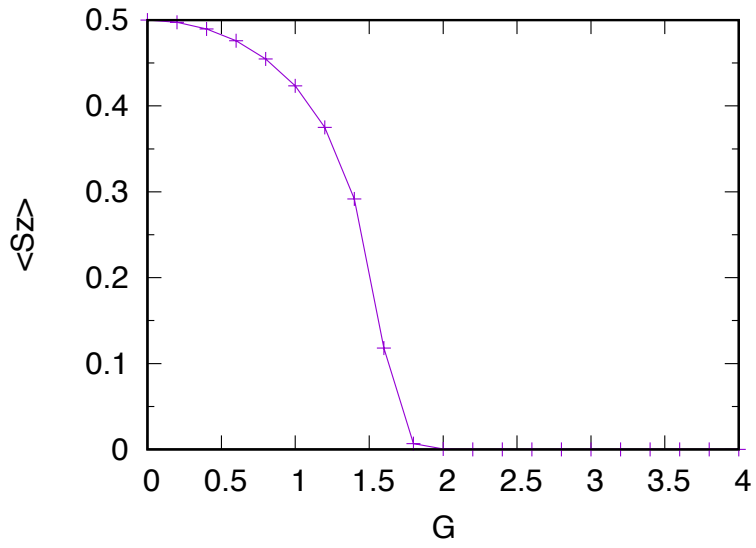
なお、サンプルスクリプトの例として、sample/01_transverse_field_ising の tutorial_example.py , tutorial_read.py があります。実行は、

```
$ python tutorial_example.py
```

でできます (MacBook2017, 1.4 GHz Intel Core i7 で数分程度で計算が全て終了します)。得られた結果は

```
$ python tutorial_read.py
```

とすることで, 標準出力に、G, エネルギー、 <Sz> 、 <Sx> が出力されます。



図から `G` が大きくなるにつれ、 `<Sz>` が `0.5` から徐々に小さくなり最終的には 0 になることがわかります。

第 5 章

ファイルフォーマット

5.1 `tense_simple` の入力ファイル

- ファイルフォーマットは TOML 形式
- `model`, `parameter`, `lattice`, `observable`, `correlation` の 5 つのセクションを持ちます。 なお、`observable`, `correlation` セクションは、エキスパートモードの入力ファイルと共通です。
  - 将来的にはファイル名を指定することで分割可能にする予定です。

5.1.1 `parameter` セクション

`parameter` セクションの内容は、そのまま出力の `parameter` セクションにコピーされます。

また、simple update と full update における虚時間発展演算子の虚時間刻み幅をサブセクション `simple_update`, `full_update` で指定できます。

名前	説明	型	デフォルト
`tau`	虚時間発展演算子における虚時間の刻み幅	実数	0.01

以下は `tenes` の入力ファイルと共通のパラメータになります。

parameter.tensor

名前	説明	型	デフォルト
D	中心テンソルがもつ virtual ボンドの次元	整数	2
CHI	角転送行列の virtual ボンドの次元	整数	4
save_dir	最適化後のテンソルを書き込むディレクトリ	文字列	""
load_dir	初期テンソルを読み込むディレクトリ	文字列	""

- save_dir - 最適化後のテンソルをこのディレクトリ以下に保存します- 空文字列の場合は保存しません
- load_dir - 各種テンソルをこのディレクトリ以下から読み込みます- 保存したときと同じ並列度である必要があります- 空文字列の場合は読み込みません

parameter.simple_update

名前	説明	型	デフォルト
num_step	simple update の回数	整数	0
lambda_cutoff	simple update でゼロとみなす平均場の cutoff	実数	1e-12

parameter.full_update

名前	説明	型	デフォルト
num_step	full update の回数	整数	0
env_cutoff	full update で環境テンソルを計算する際にゼロとみなす特異値の cutoff	実数	1e-12
inverse_precision	full update で擬似逆行列を計算する際にゼロとみなす特異値の cutoff	実数	1e-12
convergence_epsilon	full update で truncation の最適化を行う際の収束判定値	実数	1e-12
iteration_max	full update で truncation の最適化を行う際の iteration の最大回数	整数	1000
gauge_fix	テンソルのゲージを固定するかどうか	真偽値	true
fastfullupdate	Fast full update にするかどうか	真偽値	true

parameter.ctm

名前	説明	型	デフォルト
projector_cutoff	CTM の projector を計算する際にゼロとみなす特異値の cutoff	実数	1e-12
convergence_epsilon	CTM の収束判定値	実数	1e-10
iteration_max	CTM の収束 iteration の最大回数	整数	100
projector_corner	CTM の projector 計算で 1/4 角のテンソルのみを使う	真偽値	true
use_rsvd	SVD を 乱択 SVD で置き換えるかどうか	真偽値	false
rsvd_oversampling_factor	乱択 SVD 中に計算する特異値の数の、最終的に用いる数に対する比率	実数	2.0

parameter.random

名前	説明	型	デフォルト
seed	テンソルの初期化や乱択 SVD に用いる疑似乱数生成器のシード	整数	11

MPI 並列において、各プロセスは seed にプロセス番号を足した数を実際のシードとして持ちます。

例

```
[parameter]
[parameter.tensor]
D  = 4     # tensor_dim
CHI  = 16  # env_dim

[parameter.simple_update]
num_step = 1000

[parameter.full_update]
num_step = 1

[parameter.ctm]
iteration_max = 5
```

5.1.2 `lattice` セクション

計算する格子を指定します。正方格子 (square lattice) と 蜂の巣格子 (honeycomb lattice) が定義されています。

名前	説明	型	デフォルト
`type`	格子名 (square lattice もしくは honeycomb lattice)	文字列	–
`L_sub`	ユニットセルの大きさ	整数もしくは 2 つの整数からなるリスト	–

ユニットセルは `Lx` かける `Ly` の大きさをもつ長方形の形をしています。`L_sub` として 2 つの整数からなるリストを渡した場合、はじめの要素が `Lx` に、もう片方が `Ly` になります。3 つ以上の要素からなるリストを渡した場合にはエラー終了します。`L_sub` として整数を渡した場合、 `Lx` と `Ly` とが等しくなります。

ユニットセル内のサイトは 0 から順番に番号付けされます。 x 方向から順に並びます。

`L_sub = [2,3]` としたときの例:

```
y
^      4 5
|      2 3
.->x   0 1
```

正方格子 square lattice

ボンドは水平方向 (0) と垂直方向 (1) の 2 種類あります（下図の – と | に対応)。

`L_sub = 2` のときのユニットセルは次の通り:

```
0   1
|   |
2 - 3 - 2
|   |
0 - 1 - 0
```

蜂の巣格子 honeycomb lattice

ユニットセルの大きさ (`L_sub` の各要素) は偶数でなければなりません。

ボンドは x (0), y (1), z (2) の 3 種類あります（それぞれ、下図の –, ~, | に対応)。偶数番のサイトは右 (x)、左 (y)、上 (z) に伸びるボンドを持ち、奇数番のサイトは左 (x)、右 (y)、下 (z) に伸びるボンドを持ちます。

L_sub = 2 のときのユニットセルは次の通り:

```
0   1
    |
2 ~ 3 - 2
|
0 - 1 ~ 0
```

5.1.3 model セクション

計算する模型を指定します。スピン系 (spin) が定義済みです。

名前	説明	型	デフォルト
type	模型の種類	文字列	–

模型の種類によって相互作用などのパラメータ名が変わります。

スピン系 spin

スピン系

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

名前	説明	型	デフォルト
S	局所スピンの大きさ	実数	0.5
Jx	交換相互作用 J の x 成分	実数もしくは実数のリスト	1.0
Jy	交換相互作用 J の y 成分	実数もしくは実数のリスト	1.0
Jz	交換相互作用 J の z 成分	実数もしくは実数のリスト	1.0
BQ	双二次相互作用 B	実数もしくは実数のリスト	0.0
h	縦磁場 h	実数	0.0
G	横磁場 Γ	実数	0.0
D	オンサイトスピン異方性 D	実数	0.0

交換相互作用および双二次相互作用としてリストを与えることで、格子ボンドの種類ごとに相互作用の大きさを変えることができます。リストの要素数が格子ボンドの種類より少ない場合、足りない分は指定された最後の要素で埋められます。

5.1.4 observable セクション

tenes_simple ではデフォルトでは、物理量測定に使われる局所物理量として、 S^z と S^x が定義されます。 より詳細な物理量測定は、 tenes の入力ファイルで指定する observable セクションと共通のフォーマットで指定することができます。詳細は、*tenes の入力ファイル* の observable セクションをご覧ください。

5.1.5 correlation セクション

tenes_simple では相関関数 C = <A(0)B(r)> はデフォルトでは計算されません。相関関数を計算したい場合は、 tenes の入力ファイルで指定する correlation セクションと共通のフォーマットで指定することができます。詳細は、*tenes の入力ファイル* の correlation セクションをご覧ください。

5.2 tenes の入力ファイル

- ファイルフォーマットは TOML 形式
- parameter, lattice, evolution, observable, correlation の 5 つのセクションを持ちます。
  - 将来的にはファイル名を指定することで分割可能にする予定です。

5.2.1 parameter セクション

更新回数など、種々のパラメータを記述します。このセクションのみ、各々のパラメータにデフォルト値が存在します。サブセクションとして tensor, simple_update, full_update, ctm, random を持ちます。

parameter.tensor

名前	説明	型	デフォルト
D	中心テンソルがもつ virtual ボンドの次元	整数	2
CHI	角転送行列の virtual ボンドの次元	整数	4
save_dir	最適化後のテンソルを書き込むディレクトリ	文字列	""
load_dir	初期テンソルを読み込むディレクトリ	文字列	""

- save_dir - 最適化後のテンソルをこのディレクトリ以下に保存します- 空文字列の場合は保存しません
- load_dir - 各種テンソルをこのディレクトリ以下から読み込みます- 保存したときと同じ並列度である必要があります- 空文字列の場合は読み込みません

parameter.simple_update

名前	説明	型	デフォルト
num_step	simple update の回数	整数	0
lambda_cutoff	simple update でゼロとみなす平均場の cutoff	実数	1e-12

parameter.full_update

名前	説明	型	デフォルト
num_step	full update の回数	整数	0
env_cutoff	full update で環境テンソルを計算する際にゼロとみなす特異値の cutoff	実数	1e-12
inverse_precision	full update で擬似逆行列を計算する際にゼロとみなす特異値の cutoff	実数	1e-12
convergence_epsilon	full update で truncation の最適化を行う際の収束判定値	実数	1e-12
iteration_max	full update で truncation の最適化を行う際の iteration の最大回数	整数	1000
gauge_fix	テンソルのゲージを固定するかどうか	真偽値	true
fastfullupdate	Fast full update にするかどうか	真偽値	true

parameter.ctm

名前	説明	型	デフォルト
projector_cutoff	CTM の projector を計算する際にゼロとみなす特異値の cutoff	実数	1e-12
convergence_epsilon	CTM の収束判定値	実数	1e-10
iteration_max	CTM の収束 iteration の最大回数	整数	100
projector_corner	CTM の projector 計算で 1/4 角のテンソルのみを使う	真偽値	true
use_rsvd	SVD を 乱択 SVD で置き換えるかどうか	真偽値	false
rsvd_oversampling_factor	乱択 SVD 中に計算する特異値の数の、最終的に用いる数に対する比率	実数	2.0

parameter.random

名前	説明	型	デフォルト
seed	テンソルの初期化や乱択 SVD に用いる疑似乱数生成器のシード	整数	11

MPI 並列において、各プロセスは seed にプロセス番号を足した数を実際のシードとして持ちます。

例

```
[parameter]
[parameter.tensor]
D  = 4     # tensor_dim
CHI  = 16  # env_dim

[parameter.simple_update]
num_step = 1000

[parameter.full_update]
num_step = 1

[parameter.ctm]
iteration_max = 5
```

5.2.2 lattice セクション

「ユニットセル」の情報を記述します。ユニットセルは Lx かける Ly の大きさをもつ長方形の形をしています。

名前	説明	型
L_sub	ユニットセルの大きさ	整数または整数のリスト

L_sub として 2 つの整数からなるリストを渡した場合、はじめの要素が Lx に、もう片方が Ly になります。3 つ以上の要素からなるリストを渡した場合にはエラー終了します。L_sub として整数を渡した場合、 Lx と Ly とが等しくなります。

ユニットセル内のサイトは 0 から順番に番号付けされます。 x 方向から順に並びます。

L_sub = [2,3] としたときの例:

```
y
^     4 5
|     2 3
.->x  0 1
```

ボンドの情報は evolution や observable で与えられます。

5.2.3 evolution セクション

simple update, full update で使う虚時間発展演算子を記述します。

名前	説明	型
matrix	虚時間発展演算子の行列表現	文字列のリスト
simple_update	simple update における、虚時間発展演算子のインデックスと作用するボンドの順番	文字列
full_update	full update における、虚時間発展演算子のインデックスと作用するボンドの順番	文字列

matrix

- ひとつの文字列がひとつの行列を意味します。
- 列は 1 つ以上の空白で区切られ、行は 1 つ以上の改行で区切られます。
- 定義した順番がそのまま行列の番号に対応し、 *_update での指定で使われます (0-origin)。

simple_update, full_update

- 1 行が 1 回の演算子作用を表します。
- 各行は `int int char int` の４つのフィールドからなります。
  1. ボンドがつながるサイト
  2. ボンドがつながるサイト
  3. 横方向 (h) か縦方向 (v) か
  4. 演算子番号 (0-origin)

例

```
[evolution]
simple_update = """
0 1 h 0
3 2 h 0
2 3 h 0
1 0 h 0
0 2 v 0
3 1 v 0
2 0 v 0
1 3 v 0
"""

full_update = """
0 1 h 0
3 2 h 0
2 3 h 0
1 0 h 0
0 2 v 0
3 1 v 0
2 0 v 0
1 3 v 0
"""

matrix = [
"""
0.9975031223974601 0.0 0.0 0.0
0.0 1.0025156589209967 -0.005012536523536887 0.0
0.0 -0.005012536523536888 1.0025156589209967 0.0
0.0 0.0 0.0 0.9975031223974601
"""
]
```

5.2.4 observable セクション

物理量測定に関する諸々を記述します。

名前	説明	型
local_operator	サイト演算子 (ex. Sz)	文字列のリスト
hamiltonian	ボンドハミルトニアン	文字列のリスト
hamiltonian_bonds	ボンドハミルトニアンの種類と作用するボンドの組	文字列

local_operator, hamiltonian

evolution.matrix と同様。定義した順番がそのまま演算子・ハミルトニアンのインデックスに対応します。

hamiltonian_bonds

evolution.simple_update と同様。

例

```
[observable]
local_operator = [
"""
  0.5  0.0
  0.0 -0.5
""",
"""
  0.0 0.5
  0.5 0.0
""",
]

hamiltonian_bonds = """
0 1 h 0
3 2 h 0
2 3 h 0
1 0 h 0
0 2 v 0
3 1 v 0
2 0 v 0
1 3 v 0
"""

hamiltonian = [
```

(continues on next page)

(前のページからの続き)

```
"""
  0.25   0.0    0.0     0.0
  0.0   -0.25   0.5     0.0
  0.0    0.5   -0.25    0.0
  0.0    0.0    0.0     0.25
""",
]
```

5.2.5 correlation セクション

相関関数 C = <A(0)B(r)> を指定するセクション

名前	説明	型
r_max	相関関数の距離 r の最大値	整数
operators	相関関数を測る演算子 A, B の番号	整数のリストのリスト

演算子は observable セクションで指定したものが用いられます。

例

```
[correlation]
r_max = 5
operators = [[0,0], [0,1], [1,1]]
```

5.3 出力ファイル

計算結果は output ディレクトリ以下に保存されます。

5.3.1 parameters.dat

parameter と lattice のパラメータが出力されます。

5.3.2 energy.dat

サイトあたりのエネルギーが出力されます。

5.3.3 site_obs.dat

- サイト演算子の期待値 $\langle \hat{A}_i^\alpha \rangle$ が出力されます。
- 各行 4 列からなります。
  1. 演算子のインデックス α
  2. サイトのインデックス i
  3. 期待値の実部 $\mathrm{Re}\langle \hat{A}_i^\alpha \rangle$
  4. 期待値の虚部 $\mathrm{Im}\langle \hat{A}_i^\alpha \rangle$

例

```
# $1: op_index
# $2: site_index
# $3: real
# $4: imag

0 0 1.9254946524957336e-02 0.0000000000000000e+00
0 1 -1.9262081413019553e-02 0.0000000000000000e+00
0 2 -1.9524309305592252e-02 0.0000000000000000e+00
0 3 1.9161947763206115e-02 0.0000000000000000e+00
1 0 4.0720606334876879e-01 0.0000000000000000e+00
1 1 -4.0724351173715767e-01 0.0000000000000000e+00
1 2 -4.0725596773873412e-01 0.0000000000000000e+00
1 3 4.0730891879155401e-01 0.0000000000000000e+00
```

5.3.4 neighbor_obs.dat

- サイト演算子の最近接相関 $\langle \hat{A}_i^\alpha \hat{A}_j^\alpha \rangle$ が出力されます。
- 各行 5 列からなります。
  1. 演算子のインデックス α
  2. サイトのインデックス i
  3. サイトのインデックス j
  4. 期待値の実部 $\mathrm{Re}\langle \hat{A}_i^\alpha \hat{A}_j^\alpha \rangle$
  5. 期待値の虚部 $\mathrm{Im}\langle \hat{A}_i^\alpha \hat{A}_j^\alpha \rangle$

```
# $1: op_index
# $2: source_site
# $3: target_site
# $4: real
# $5: imag

0 0 1 -7.0592761506496890e-02 0.0000000000000000e+00
0 0 2 -7.2706845643005127e-02 0.0000000000000000e+00
0 1 0 -7.1328438595739229e-02 0.0000000000000000e+00
0 1 3 -7.1952334925611358e-02 0.0000000000000000e+00
0 2 3 -7.1261036489548304e-02 0.0000000000000000e+00
0 2 0 -7.1973150756101195e-02 0.0000000000000000e+00
0 3 2 -7.0563355823021006e-02 0.0000000000000000e+00
0 3 1 -7.2680375080734049e-02 0.0000000000000000e+00
1 0 1 -1.8594286923710335e-01 0.0000000000000000e+00
1 0 2 -1.8716473167754519e-01 0.0000000000000000e+00
1 1 0 -1.8636038255007659e-01 0.0000000000000000e+00
1 1 3 -1.8676845108636669e-01 0.0000000000000000e+00
1 2 3 -1.8638418190980594e-01 0.0000000000000000e+00
1 2 0 -1.8674757673269352e-01 0.0000000000000000e+00
1 3 2 -1.8597508952501360e-01 0.0000000000000000e+00
1 3 1 -1.8719652291687905e-01 0.0000000000000000e+00
```

5.3.5 correlation.dat

- 相関関数 $C_{ij}^{\alpha\beta}(x,y) \equiv \langle \hat{A}_i^{\alpha}(0,0)\hat{A}_j^{\beta}(x,y)\rangle$ が出力されます。
- 各行 8 列から構成されます。
  1. 左演算子のインデックス α
  2. 左演算子のサイトインデックス i
  3. 右演算子のインデックス β
  4. 右演算子のサイトインデックス j
  5. 右演算子のユニットセルオフセット (x) x
  6. 右演算子のユニットセルオフセット (y) y
  7. 演算子の実部 $\mathrm{Re}C_{ij}^{\alpha\beta}(x,y)$
  8. 演算子の虚部 $\mathrm{Im}C_{ij}^{\alpha\beta}(x,y)$

例

```
# $1: left_op
# $2: left_site
# $3: right_op
# $4: right_site
# $5: offset_x
# $6: offset_y
# $7: real
# $8: imag

0 0 0 1 0 0 -7.0592761506496792e-02 0.0000000000000000e+00
0 0 0 0 1 0 1.1966884322676102e-02 0.0000000000000000e+00
0 0 0 1 1 0 -2.4308622932000586e-03 0.0000000000000000e+00
0 0 0 0 2 0 7.4272919452849631e-04 0.0000000000000000e+00
0 0 0 1 2 0 -4.3879481941688542e-04 0.0000000000000000e+00
0 0 0 2 0 0 -7.2706845643005114e-02 0.0000000000000000e+00
0 0 0 0 0 1 1.2333984574662128e-02 0.0000000000000000e+00
0 0 0 2 0 1 -2.5011118624440735e-03 0.0000000000000000e+00
0 0 0 0 0 2 7.5460780658739152e-04 0.0000000000000000e+00
0 0 0 2 0 2 -4.4773455996967955e-04 0.0000000000000000e+00
1 0 1 1 0 0 -1.8594286923710324e-01 0.0000000000000000e+00
（中略）
1 3 1 1 0 3 -1.6587424589146155e-01 0.0000000000000000e+00
```

5.3.6 time.dat

計算時間が出力されます。

第 6 章

参考文献

TeNeS で使用しているアルゴリズムについて紹介している資料については以下のものがあります。アルゴリズムの詳細を知りたい方はぜひご覧ください。

- テンソルネットワークに関する講義資料
  - スライド：https://www.issp.u-tokyo.ac.jp/public/caqmp2019/slides/808L_Okubo.pdf
  - youtube のビデオ：https://www.youtube.com/watch?v=EL9iGwoqZes&list=PLyL_XwLqTTm6roebw6MgIK0ljrN8cSQUX
- Simple update に関する論文
  - H. C. Jiang, Z. Y. Weng, and T. Xiang, Phys. Rev. Lett. **101**, 090603 (2008).
- Full update に関する論文（CTMRG による環境の評価を含む）
  - J. Jordan, R. Orús, G. Vidal, F. Verstraete, and J. I. Cirac, Phys. Rev. Lett. **101**, 250602 (2008).
  - R. Orús and G. Vidal, Phys. Rev. B **80**, 094403 (2009).
  - P. Corboz, T. M. Rice, and M. Troyer, Phys. Rev. Lett. **113**, 046402 (2014).
- Fast Full update, Gauge fixing について
  - Ho N. Phien, Johann A. Bengua, Hoang D. Tuan, Philippe Corboz, and Román Orús Phys. Rev. B **92**, 035142 (2015).
- 参考になるレビュー
  - Román Orús, Annals of Physics, **349**,117 (2014).
  - Román Orús, Nature Reviews Physics, **1**, 538 (2019).

第 7 章

謝辞

TeNeS の開発は，文部科学省ポスト「京」萌芽的課題 1 「基礎科学のフロンティア ― 極限への挑戦」及びポスト「京」重点課題 7 「次世代の産業を支える新機能デバイス・高性能材料の創成」の一環として実施されました．また, 東京大学物性研究所 ソフトウェア高度化プロジェクト (2019 年度) の支援も受け開発されました。この場を借りて感謝します。

第 8 章

お問い合わせ

TeNeS に関するお問い合わせはこちらにお寄せください。

- バグ報告

  TeNeS のバグ関連の報告は GitHub の Issues で受け付けています。

  バグを早期に解決するため、報告時には次のガイドラインに従ってください。

  - 使用している TeNeS のバージョンを指定してください。
  - インストールに問題がある場合には、使用しているオペレーティングシステムとコンパイラの情報についてお知らせください. また, cmake および make 実行時の入出力と CMakeCache.txt も一緒に記載してください。
  - 実行に問題が生じた場合は, 実行に使用した入力ファイルとその出力を記載してください。

- その他

  研究に関連するトピックなど GitHub の Issues で相談しづらいことを問い合わせる際には, 以下の連絡先にコンタクトをしてください。

  E-mail: tenes-dev__at__issp.u-tokyo.ac.jp (_at_を@に変更してください)