Tutorial of TeNeS with MateriApps Live

University of Tokyo, Tsuyoshi Okubo

Tensor Netwok Solver (TeNeS)

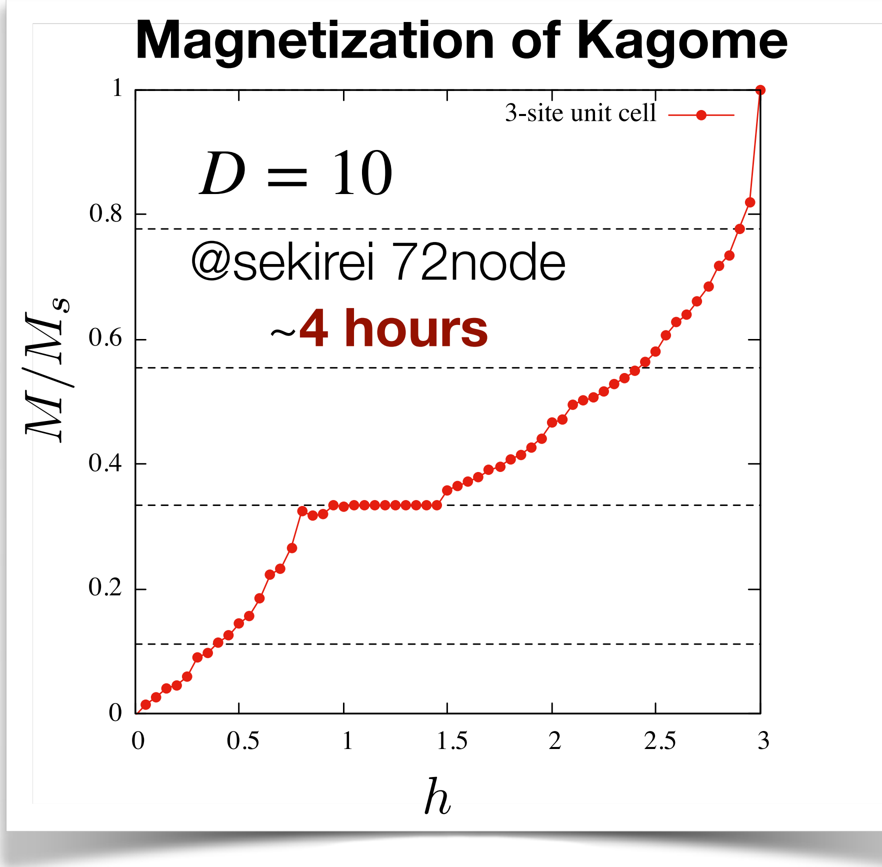
Y. Motoyama, T. Okubo, et al., Comput. Phys. Commun. 279, 108437 (2022).

TeNeS

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

Ground state calculation of 2d quantum models by iTPS

- ☑ Tensor optimization by the imaginary time evolution
- ☑ Massively parallelized by MPI/OpenMP
  - ☑ parallelization of tensor operations by mptensor (Morita)
- ☑ Easy calculations for quantum spins or bosons
- ☑ Easy access to standard 2d lattices
  - ☑ In principle, we can use any 2d lattices



Developpers:

- Tsuyoshi Okubo: Core algorithms
- Satoshi Morita: Related libraries and tools
- Yuichi Motoyama: Main programs
- Kazuyoshi Yoshimi: User tests, tutorials, and project management
- Takeo Kato: User tests and tutorials
- Naoki Kawashima: Project leader

Features of TeNeS

- We can calculate the ground states of various two-dimensional quantum spin models.
    - We can also deal with Bose Hubbard model.
    - So far, we do not support fermions.
- We represent a quantum state using square lattice iTPS
    - Models on other 2d lattices are mapped onto the square lattice.
- Contraction of iTPS is done by CTM environment.
- iTPS is optimized through imaginary time evolution.
- MPI/OpenMP hybrid parallelization supported by mptensor.
    - https://github.com/smorita/mptensor

Features of TeNeS: Models

- Various standard models are already defined in TeNeS, and one can easily simulate them.
  - General spin-S spin models

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

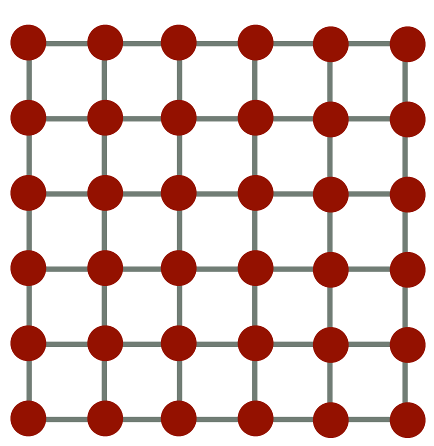
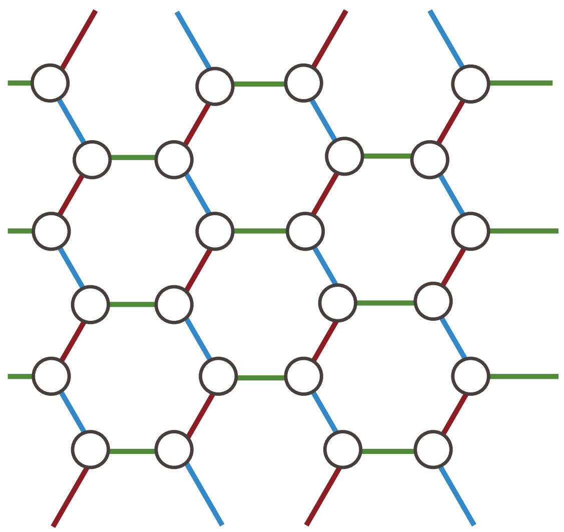
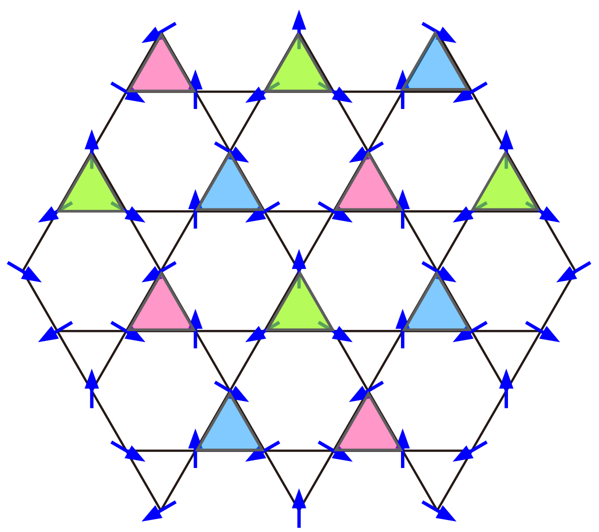
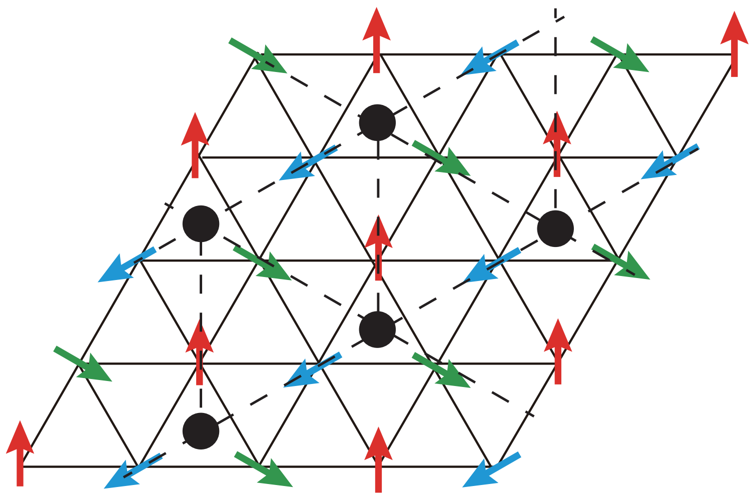
  - Bose Hubbard model (with particle number cutoff

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

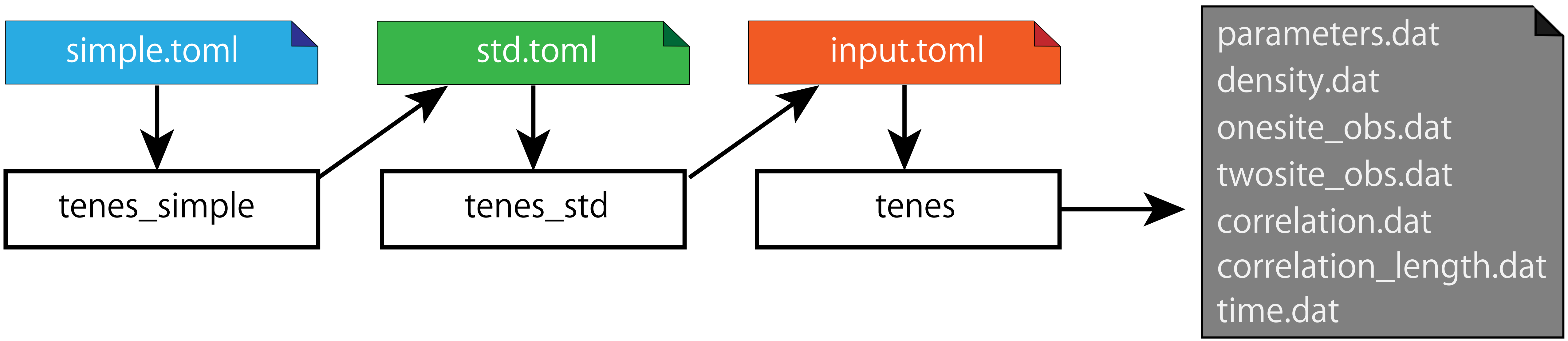
- We can also simulate general models by defining the Hamiltonian matrix elements.

Features of TeNeS: Lattices

- Various standard lattices are already defined in TeNeS
  - Square lattice
  - Honeycomb lattice
  - Kagome lattice
  - Triangular lattice
- We can deal with further neighbor interactions on these lattices.

Structure of TeNeS



- tenes_simple (simple.toml → std.toml)
  - This makes Hamiltonian and lattice information from the input for predefined models.
- tenes_std (std.toml → input.toml)
  - This makes ITE operators for the give Hamiltonian and lattice.
- tenes (input.toml → results)
  - This calculates the ground state by ITE, and output expectation values of it.

How to use TeNeS

- We use **MateriApps Live**, a Linux virtual environment, for this tutorial.
  - https://github.com/cmsi/MateriAppsLive
  - We can use
    - **Virtual box (For windows and intel Mac)** https://github.com/cmsi/MateriAppsLive/wiki/GettingStartedOVA-en
    - **Docker (For Mac)** https://github.com/cmsi/MateriAppsLive/wiki/GettingStartedDocker-en
- TeNeS is already installed in the virtual environment!

Tutorials for the simple mode

Here we do a simulation for predefined models by the simple mode.

Before the simulation, we update TeNeS in MALive.

```
$ sudo apt update
$ sudo apt install tenes
```

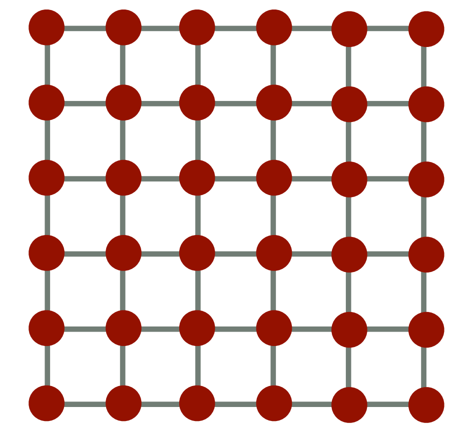
*The password is `live`.

Then, we copy samples to our directory.

```
$ cd
$ cp -r /usr/share/tenes .
$ cd tenes/sample/
```

Sample 01: Transverse field Ising model

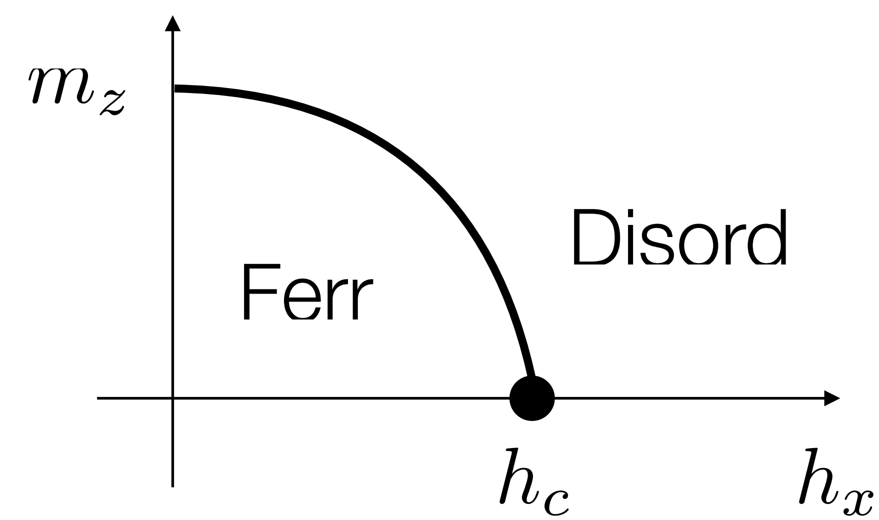
Transverse field Ising model on the square lattice



$$\mathcal{H} = J_z \sum_{\langle i,j \rangle} S_i^z S_i^z - h_x \sum_i S_i^x$$

Move to the sample directory, and see the input file.

```
$ cd 01_transverse_field_ising
$ cat simple.toml
```




```
[parameter]
. . . skipped some lines . . .
[lattice]
type = "square lattice" # Type of lattice
L = 2                   # X length of unit cell
W = 2                   # Y length of unit cell
virtual_dim = 2         # Bond dimension of bulk tensors
initial = "ferro"       # Initial condition

[model]
type = "spin" # Type of model
Jz = -1.0     # Jz SzSz
Jx = 0.0      # Jx SxSx
Jy = 0.0      # Jy SySy
hx = 0.0      # hx Sx
```

Sample 01: Transverse field Ising model

Firstly just run a simulation by the original input (J_z = -1, h_x = 0).

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

We obtain `std.toml`.

We obtain `input.toml`.

Main calculation

```
Onesite observables per site:
  Sz          = 0.5 0
  Sx          = -1.28526e-13 0
Twosite observables per site:
  hamiltonian = -0.5 0
  SzSz        = 0.5 0
  SxSx        = -1.73749e-18 0
  SySy        = 1.73749e-18 0
    Save elapsed times to output/time.dat
Wall times [sec.]:
  all             = 1.36678
  simple update = 1.28468
  full update     = 0
  environment   = 0.0438204
  observable      = 0.0285048
```

We obtained the ferromagnetic state correctly.

Sample 01: Transverse field Ising model

Let's turn on the transverse field. Here we set $h_x = 3$.

```
$ mv output output_hx0
$ vi simple.toml
```

Move previous outputs

Edit the input file. (You can use any editors)

```
[parameter]
. . . skipped some lines . . .
[lattice]
type = "square lattice" # Type of lattice
L = 2                   # X length of unit cell
W = 2                   # Y length of unit cell
virtual_dim = 2         # Bond dimension of bulk tensors
initial = "ferro"       # Initial condition

[model]
type = "spin" # Type of model
Jz = -1.0     # Jz SzSz
Jx = 0.0      # Jx SxSx
Jy = 0.0      # Jy SySy
hx = 0.0      # hx Sx
```

Change the value from 0.0 to 3.0.

Sample 01: Transverse field Ising model

Run a simulation by the new input ($J_z = -1$, $h_x = 3$).

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

We obtain `std.toml`.

We obtain `input.toml`.

Main calculation

```
Onesite observables per site:
  Sz          = 8.05351e-09 0
  Sx          = 0.49251 0
Twosite observables per site:
  hamiltonian = -1.52141 0
  SzSz        = 0.0438808 0
  SxSx        = 0.488707 0
  SySy        = -0.040709 0
    Save elapsed times to output/time.dat
Wall times [sec.]:
  all            = 1.40828
  simple update = 1.28499
  full update    = 0
  environment   = 0.0830286
  observable     = 0.0273423
```

<Sz> became smaller and <Sx> increased.

Sample 01: Transverse field Ising model

Perform a set of calculations by varying the parameter h_x.

Here we have convenient scripts for this purpose.

"tutorial_example.py" This creates new input files for various hx based on simple.toml, and run simulations. *Parameters for iTPS simulation are fixed.

"tutorial_read.py" This corrects relevant outputs from the simulation results.

```
$ mv output output_hx3
$ python3 tutorial_example.py
$ python3 tutorial_read.py
```

Move previous outputs

```
0.0 -5.0000000000000000e-01 5.0000000000000000e-01 -1.28526262481783166e-13
0.2 -5.05004176692116613e-01 4.97482489246812931e-01 5.00836559134375525e-02
0.4 -5.20067358911737165e-01 4.89712356048553898e-01 1.00677102836352519e-01
0.6000000000000001 -5.45345944127035076e-01 4.75974311279677875e-01 1.52332679263363163e-01
0.8 -5.81118489589626419e-01 4.54819709041673070e-01 2.05703615600286410e-01
1.0 -6.27825504636870924e-01 4.23449891449173332e-01 2.61634985373808693e-01
1.2000000000000002 -6.86162453242837089e-01 3.76008716148938216e-01 3.21286581242890168e-01
1.4000000000000001 -7.57303058332144063e-01 2.97866954877537060e-01 3.86024177993127549e-01
1.6 -8.43079240807139141e-01 1.45603092256966299e-01 4.51659225163221101e-01
1.8 -9.37833056836064660e-01 1.29597626028327938e-02 4.75420591969319573e-01
2.0 -1.03342333861389823e+00 8.18829125482502728e-04 4.81205045326086711e-01
2.2 -1.12998527934941073e+00 6.56261154191476533e-05 4.85019486173761227e-01
2.4000000000000004 -1.22722461097822300e+00 6.15231156984653096e-06 4.87737178838292929e-01
2.6 -1.32495042110809980e+00 6.35774448821559990e-07 4.89754086990076054e-01
2.8000000000000003 -1.42303986960010898e+00 7.00131184489289022e-08 4.91298098071581257e-01
3.0 -1.52140952584773581e+00 8.05351277567753402e-09 4.92509565166792285e-01
```

Sample 01: Transverse field Ising model

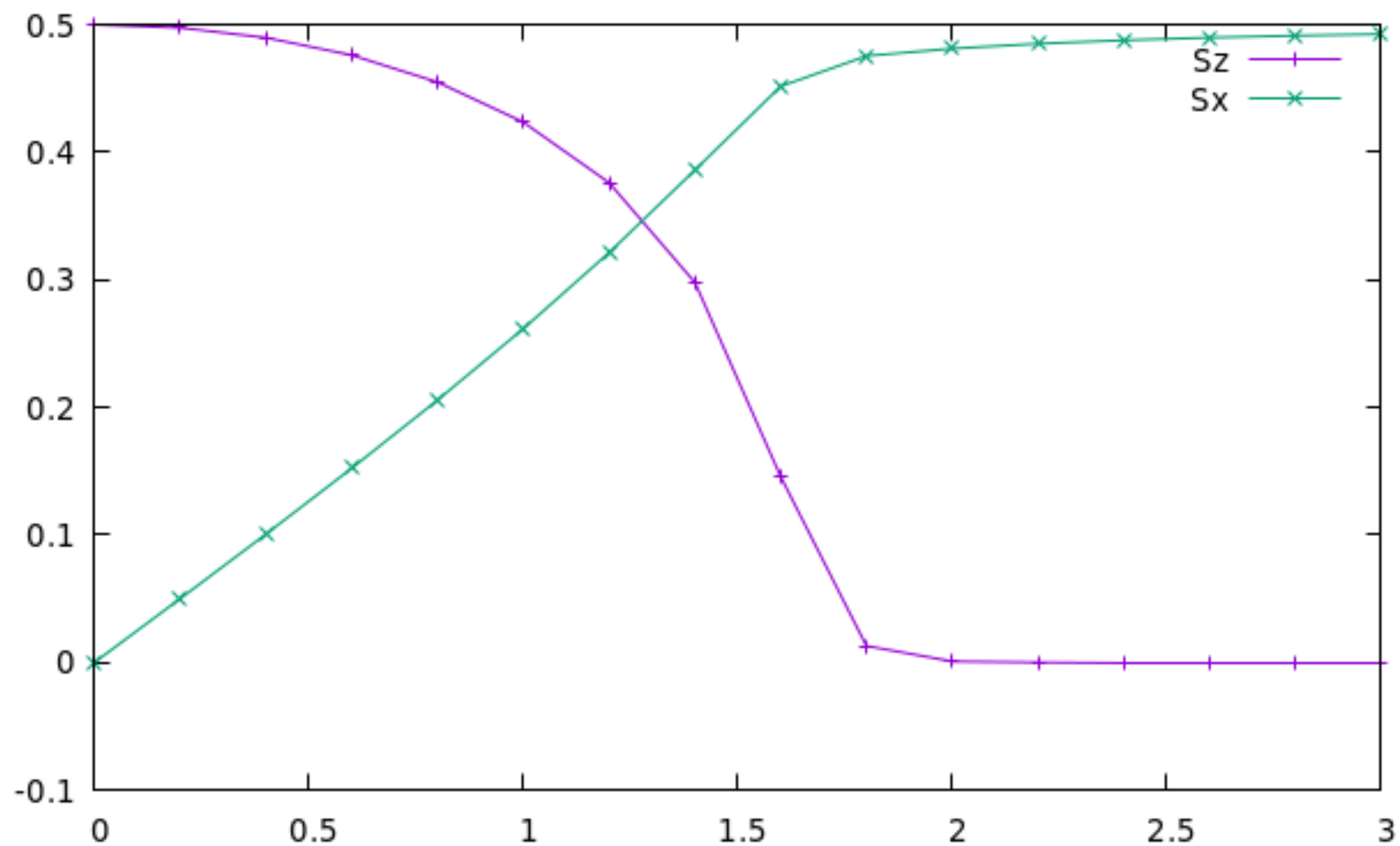
Finally, plot the result.

```
$ python3 tutorial_read.py > result_D2.dat
$ gnuplot
```

```
gnuplot> p "result_D2.dat" u 1:3 ti "Sz" w lp, "" u 1:4 ti "Sx" w lp
```

*Format of “result_D2.dat”

```
# $1: hx
# $2: energy
# $3: sz
# $4: sx
```



Sample 01: Transverse field Ising model

Exercise: Try simulations with larger bond dimensions.

It can be done by modifying “simple.toml”.

```
[lattice]
type = "square lattice" # Type of lattice
L = 2                   # X length of unit cell
W = 2                   # Y length of unit cell
virtual_dim = 2         # Bond dimension of bulk tensors
initial = "ferro"       # Initial condition
```

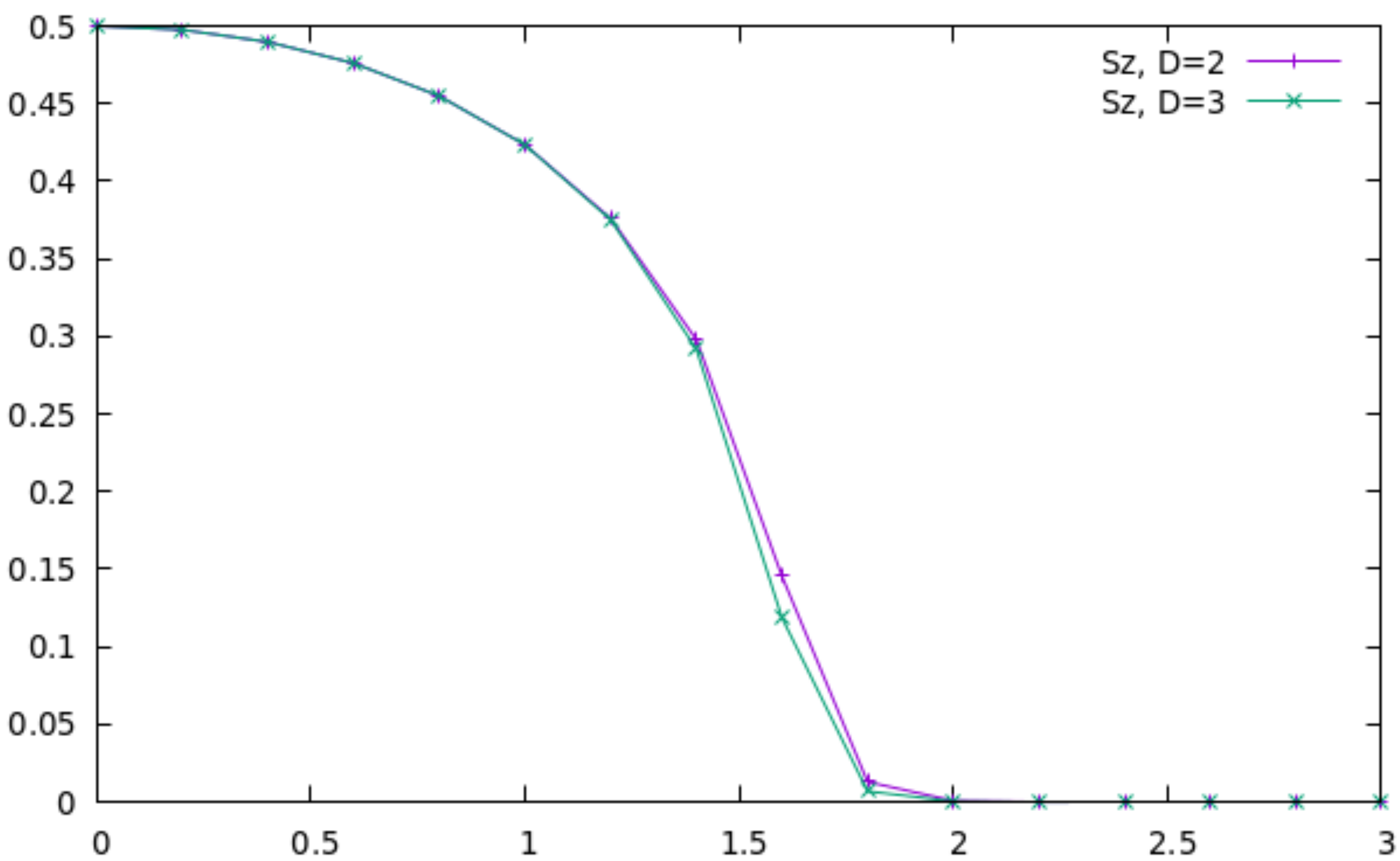
Change this value, e.g., 3.

```
$ python3 tutorial_example.py
$ python3 tutorial_read.py > result_D3.dat
```

Notice: Execution time increases when we increase the bond dimension.

Sample 01: Transverse field Ising model

```
$gnuplot
gnuplot> p "result_D2.dat" u 1:3 ti "Sz, D=2" w lp, "result_D3.dat" u 1:3 ti "Sz, D=3" w lp
```



Smaller D calculations are less accurate in the vicinity of the critical point.

Other samples

- 02_AFH_square
  - Simulation on the square lattice Heisenberg model with S=1/2
- 03_S1_AFH_square
  - Sample input file for the square lattice Heisenberg model with S=1
- 04_Kitaev_honeycomb
  - Sample input file for the honeycomb lattice Kitaev model
- 05_magnetization
  - Simulation of magnetization processes of the Heisenberg model on the square and triangular lattices (recommended)
- 06_hardcore_boson_triangular
  - Simulation of hardcore Bose Hubbard model on the triangular lattice.

To try 02, 05, or 06, please read “README.md” in their directries.

References

GitHub repository:

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

You can find the manual and other information in

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

The paper about TeNeS:

Y. Motoyama, T. Okubo, et al., Comput. Phys. Commun. 279, 108437 (2022).