HΦ Tutorial

University of Tokyo

Nov 26, 2021

CONTENTS

CHAPTER

ONE

TUTORIAL FOR CALCULATIONS AT ZERO TEMPERATURE

1.1 Spin 1/2 Dimer

Let's solve the following spin 1/2 dimer model (2-site Heisenberg model).

$$H = J\mathbf{S}_0 \cdot \mathbf{S}_1 \tag{1.1}$$

The input file (`samples/tutorial_1.1/stan1.in`) is as follows:

```
L=2
model = "Spin"
method = "FullDiag"
lattice = "chain"
J = 0.5
2Sz = 0
2S  = 1
```

It should be noted that the `L=2` chain has two sites connected by *two* bonds with each other, and so we let `J=0.5` to simulate the dimer model with $J = 1$.

You can execute HPhi as follows

```
HPhi -s stan.in
```

1.1.1 Check the energy

Please check whether the energies are given as follows.

$E_{\mathrm{min}} = -3/4$ (singlet)

$E_{\mathrm{max}} = 1/4$ (triplet)

1.1.2 Check S dependence

By changing 2S=1 in stan.in, you can treat spin-S dimer (eg. 2S=2 means S=1, see `samples/tutorial_1.1/stan2.in`). Please check whether the energies are given as follows.

$E_{\mathrm{min}} = -S(S+1)$

$E_{\mathrm{max}} = S^2$

1.1.3 Add magnetic field H

By adding H in stan.in, selecting model as "SpinGC", and deleting "2Sz=0" you can examine the effects of the magnetic field in the dimer model. An example of the input file (`samples/tutorial_1.1/stan3.in`) is as follows:

```
L=2
model = "SpinGC"
method = "FullDiag"
lattice = "chain"
J = 0.5
2S  = 1
H   = 2
```

Please check whether the ground state becomes polarized state (Sz=1).

1.1.4 Try to use Lanczos method

By selecting method as "Lanczos" you can perform the Lanczos calculations. An example of the input file (`samples/tutorial_1.1/stan4.in`) is as follows:

```
L=2
model = "SpinGC"
method = "Lanczos"
lattice = "chain"
J = 0.5
2S  = 1
H   = 2
```

Please check the Lanczos method reproduces the energy (energy is output in ***output/zvo_energy.dat**).

This is just a pedagogical example. By changing H = 20 (very large magnetic field), please examine what will happen. This calculation may **fail**! Please think why the Lanczos method fails for large magnetic field.

1.1.5 Try to use LOBCG method

LOBCG is locally optimal block conjugate gradient method. By selecting method as "CG", you can perform the LOBCG calculations. An example of the input file (`samples/tutorial_1.1/stan5a.in`) is as follows:

```
L=2
model = "SpinGC"
method = "CG"
lattice = "chain"
J = 0.5
2S  = 1
H   = 2
```

Please check the LOBCG method reproduces the energy (energy is output in ***output/zvo_energy.dat**).

This is just a pedagogical example. By changing H = 20 (very large magnetic field), please examine what will happen. In contrast to the Lanczos method, LOBCG will work well ! Please think why the LOBCG method works well for the large magnetic field.

Please also check the excited states can be correctly obtained by using LOBCG method. (Please compare the energies obtained by the full diagonalization.) An example of the input file (`samples/tutorial_1.1/stan5b.in`) is as follows:

```
L=2
model = "SpinGC"
method = "CG"
lattice = "chain"
J = 0.5
2S  = 1
H   = 2
exct = 4
```

Here, exct represents the number of excited states, which are obtained by the LOBCG method.

1.2 Hubbard Dimer

Let's solve the following the Hubbard dimer model.

$$H = -t\sum_{\sigma}(c^{\dagger}_{0\sigma}c_{1\sigma} + \text{H.c.}) + U(n_{0\uparrow}n_{0\downarrow} + n_{1\uparrow}n_{1\downarrow}) \tag{1.2}$$

The input file (`samples/tutorial_1.2/stan1.in`) is as follows:

```
model = "Hubbard"
method = "FullDiag"
lattice = "chain"
L=2
t = -0.5
U = 4
2Sz = 0
nelec = 2
```

You can execute HPhi as follows

```
HPhi -s stan.in
```

1.2.1 Check the energy

For the Hubbard dimer at half filling with total Sz=0, energies are given as follows:

$E = 0, U, \frac{U}{2} \times (1 \pm \sqrt{(1 + (4t/U)^2)})$

For example, by taking $U = 4, t = -1$, the energies are given as follows:

$E = -0.828427, 0, 4, 4.828427$

It is note that simple mathematical calculations can be done using:

```
bc -l
```

on the terminal.

1.2.2 Try to use LOBCG method

The input file (`samples/tutorial_1.2/stan2.in`) is as follows:

```
model = "Hubbard"
method = "CG"
lattice = "chain"
L=2
t = -0.5
U = 4
2Sz = 0
nelec = 2
exct = 4
```

Please check whether LOBCG method correctly reproduces the energies including the excited states.

1.3 Hubbard Trimer

Let's solve the following the Hubbard trimer model (Hubbard model on a triangle).

$$H = -t\sum_{\sigma}(c^{\dagger}_{0\sigma}c_{1\sigma} + c^{\dagger}_{1\sigma}c_{2\sigma} + c^{\dagger}_{2\sigma}c_{0\sigma} + \text{H.c.}) + U\sum_{i}(n_{i\uparrow}n_{i\downarrow}) \tag{1.3}$$

The input file (`samples/tutorial_1.3/stan1.in`) is as follows:

```
model = "Hubbard"
method = "FullDiag"
lattice = "chain"
L = 3
t = -1
U = 4
2Sz = 0
nelec = 2
```

Note that the filling is not half filling.

You can execute HPhi as follows

```
HPhi -s stan.in
```

1.3.1 Ferromagnetic ground state

For the Hubbard model on a triangle with one hole, it is known that the **perfect ferromagnetism** becomes ground state. Please check that.

If you want know the mechanism of the ferromagnetism, please see **Hal Tasaki, Kotai Butsuri, Vol. 31, 173 (1996)**. This is one of the simplest example of the Nagaoka's ferromagnetism.

1.3.2 Effects of transfer integrals

Please the effects of the sign of the transfer integrals. **For example, what happens if you take t = 1 ?**

Another interesting example is by changing the transfer integrals between site 0 and site 2. Following an example of the **trans.def**

```
========================
NTransfer      12
========================
========i_j_s_tijs======
========================
  1     0     0     0         -1.000000000000000          0.000000000000000
  0     0     1     0         -1.000000000000000          0.000000000000000
  1     1     0     1         -1.000000000000000          0.000000000000000
  0     1     1     1         -1.000000000000000          0.000000000000000
  2     0     0     0         -2.000000000000000          0.000000000000000
  0     0     2     0         -2.000000000000000          0.000000000000000
  2     1     0     1         -2.000000000000000          0.000000000000000
  0     1     2     1         -2.000000000000000          0.000000000000000
  2     0     1     0         -1.000000000000000          0.000000000000000
  1     0     2     0         -1.000000000000000          0.000000000000000
  2     1     1     1         -1.000000000000000          0.000000000000000
  1     1     2     1         -1.000000000000000          0.000000000000000
```

Using this transfer integrals, please examine the U dependence of the ground state. Is there phase transition between singlet ground state and the perfect ferromagnetism ?

1.4 Heisenberg chain (zero temperature)

Let's solve the following spin 1/2 Heisenberg model on the chain.

$$H = J \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j \tag{1.4}$$

The input file (`samples/tutorial_1.4/stan1.in`) for 16-site Heisenberg model is as follows:

```
L       = 16
model   = "Spin"
method  = "CG"
lattice = "chain"
J = 1
2Sz = 0
2S  = 1
```

You can execute HPhi as follows

```
HPhi -s stan.in
```

1.4.1 Check the energy

Please check whether the energies are given as follows.

$$E_0 = -7.142296 \tag{1.5}$$

1.4.2 Obtaining the excited state

By adding **exct=2**, you can obtain the 2 low-energy states (`samples/tutorial_1.4/stan2.in`). Please check the energies.

$$\begin{aligned} E_0 &= -7.142296 \\ E_1 &= -6.872107 \end{aligned} \tag{1.6}$$

1.4.3 Size dependence of the spin gap

The spin gap at finite system size is defined as $\Delta = E_1 - E_0$. For 16-site, we obtain $\Delta \sim 0.2701$.

Please examine how Δ behaves as a function of system size L (`samples/tutorial_1.4/stan3.in` for L=20). (available system size on PC may be L=24)

1.4.4 Haldane gap

By performing a similar calculations for S=1 system, please examine how Δ behaves as a function of system size L (`samples/tutorial_1.4/stan4.in`). It is known that the finite spin gap exists even in the thermodynamic limit ($L = \infty$). This spin gap is often called Haldane gap.

1.5 J1-J2 Heisenberg model

Here, we solve the $J_1 - J_2$ Heisenberg model on the square lattice, which is a canonical example of the frustrated magnets. Its Hamiltonian is defined as

$$\mathcal{H} = J_1 \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j + J_2 \sum_{\langle\langle i,j \rangle\rangle} \mathbf{S}_i \cdot \mathbf{S}_j, \tag{1.7}$$

where J_1 (J_2) represents the nearest (next-nearest) neighbor interactions.

An input file (`samples/tutorial_1.5/stan1.in`) for treating $J_1 - J_2$ Heisenberg model is given as

```
model = "Spin"
method = "CG"
lattice = "square"
L = 4
W = 4
J = 1
J' = 1
2S = 1
2Sz = 0
exct = 4
```

Here, J (J') represents J_1 (J_2).

1.5.1 Calculations of spin structure factors for ground state

First, we calculate the spin structure factors, which are defined as

$$S(\mathbf{q}) = \frac{1}{N_{\mathrm{s}}} \sum_{i,j} \mathbf{S}_i \cdot \mathbf{S}_j \tag{1.8}$$

To calculate $S(\mathbf{q})$, it is necessary to prepare a proper input file for two-body Green functions. By using a python script **MakeGreen.py** (`samples/tutorial_1.5/MakeGreen.py`), you can generate **greentwo.def** for calculating $S(\mathbf{q})$. To use **MakeGreen.py** , an input file for specifying lattice geometry (**input.txt**, `samples/tutorial_1.5/input.txt`) is necessary, whose form is given as follows

```
Lx 4
Ly 4
Lz 1
orb_num 1
```

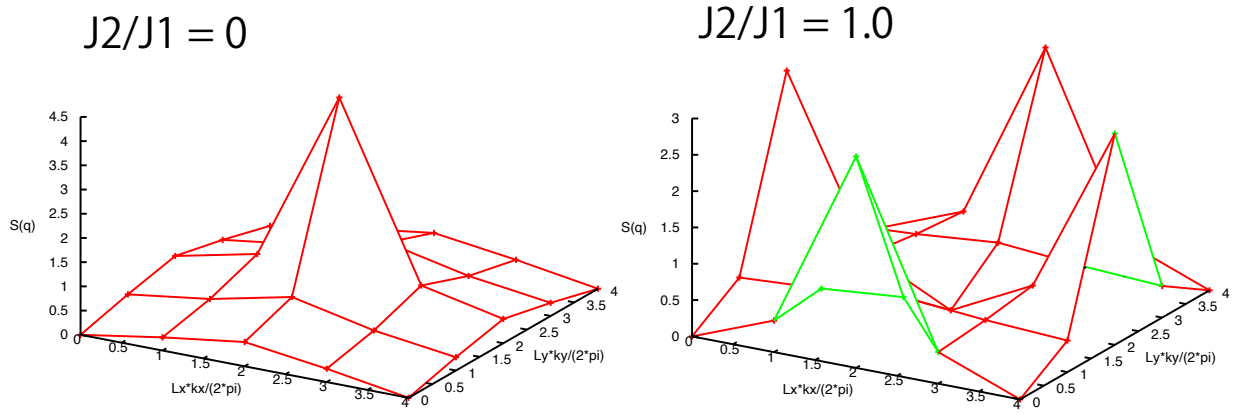
Here, Lx (orb_num) represents the length of the x direction (number of orbitals).

By using a python script **CalcSq.py** (`samples/tutorial_1.5/CalcSq.py`), you can calculate $S(\mathbf{q})$ from **output/zvo_cisajscktalt_eigen0.dat**.

Procedure for calculating and visualizing $S(\mathbf{q})$ is given as follows ::

1. HPhi -sdrt stan.in
2. pyhton3 MakeGreen.py (input.txt is necessary)
3. HPhi -e namelist.def
4. python3 CalcSq.py (input.txt is necessary)
5. You can obtain **Sq_eigen0.dat** !!
6. plot "Sq_eigen0.dat" u 1:2:3 (using gnuplot)

Following the procedure, please see how $S(\mathbf{q})$ changes by changing J'. As an example, we show $S(\mathbf{q})$ for J1/J2=0 and 1 below.



1.5.2 Calculations of spin structure factors for excited states

By changing exct in stan.in, you can obtain several excited states. For those excited states, by changing **max_num=1** in CalcSq.py as, for example, **max_num=4**, you can obtain $S(\mathbf{q})$ for the excited states. As a practice for editing file, please try to edit CalcSq.py manually (open CalcSq.py, finding variable max_num =1, and changing max_num = 4).

Please see how $S(\mathbf{q})$ in the excited states changes by changing J'. For example, please check what is the nature of the first excited state J'=0,0.5, and 1.

1.6 How to use Expert mode

If you prepare input files, you can perform calculations for arbitrary Hamiltonians with any one-body potentials and the two-body interactions. By taking spin 1/2 system as an example, we explain how to prepare input files. For spin 1/2 system, we prepare simple python scripts (`samples/tutorial_1.6/MakeDef.py`) that can generate the input files for general Hamiltonians, which are defined as

$$\mathcal{H} = \sum_{i,j} J_{i,j}^{\alpha,\beta} \mathbf{S}_i^{\alpha} \mathbf{S}_j^{\beta}. \tag{1.9}$$

Note tat `samples/tutorial_1.6/read.py` and `samples/tutorial_1.6/hphi_io.py` are necessary for **MakeDef.py**. To use *MakeDef.py*, it is necessary to prepare two input files, **input.txt** and **pair.txt** (examples of them are available in `samples/tutorial_1.6`).

In **input.txt**, two parameters **Ns** (number of sites) and **exct** (number of excited states) are specified.

Below is an example of **input.txt** for 2 site Heisenberg model

```
Ns 2
exct 2
```

In **pair.txt**, you specify the interaction terms in the form

$$i \quad j \quad \alpha \quad \beta \quad J_{i,j}^{\alpha,\beta} \tag{1.10}$$

Below is an example of **pair.txt** for 2 site Heisenberg model

```
0 1 x x 0.5
0 1 y y 0.5
0 1 z z 0.5
```

You can also specify the non-diagonal interaction as

```
0 1 x x 0.5
0 1 y y 0.5
0 1 z z 0.5
0 1 x y 0.5
0 1 x z 0.5
0 1 y z 0.5
```

Note that interaction terms must be specified for **(x,y), (x,z), (y,z)** and **(y,x), (z,x), (z,y) cannot be used**.

1.6.1 Exercise

By changing **pair.txt** and **input.txt**, you can treat your favorite models. For example, please try to make input files for the **Kitaev model** on the honeycomb lattice. We note that the **Kitaev model** can be used in the Standard mode.

Another example is the **XY model** on the chain. In the standard model, you can also treat **XY model** by omitting "CoulombInter coulombinter.def" and "Hund hund.def" in namelist.def.

1.7 Use eigenvectors

In this tutorial, we will study how to read the eigenvectors. In the standard mode, setting `EigenVecIO = "Out"` makes HPhi to write the calculated eigenvectors as `output/zvo_eigenvec_[index]_rank_[rank].dat`, where `[index]` is the index of the states (e.g., the ground state has `[index] = 0`) and `[rank]` is the rank of the process. In the MPI parallelization with Npara processes, HPhi splits the whole Hilbert space into the Npara blocks and each process treats one of them. The file format is described in the reference manual . For example, the following python function (`samples/tutorial-1.7/read_gs.py`) reads the vector:

```
def read_gs(*, index=0, rank=0):
    import numpy as np
    from os.path import join
    from struct import unpack

    filename = join("output",
                    "zvo_eigenvec_{}_rank_{}.dat".format(index,
                                                         rank))
    with open(filename, "rb") as f:
        f.read(4)
        nelems = unpack("L", f.read(8))[0]
        ret = np.zeros(nelems, dtype=np.complex)
        f.read(16)
        for i in range(nelems):
            re = unpack("d", f.read(8))[0]
            im = unpack("d", f.read(8))[0]
            ret[i] = np.complex(re, im)
        return ret
```

1.7.1 Exercise

Check the orthogonality of the eigenvectors calculated by the LOBCG method by calculating the norm and the inner product of some of the eigenvectors.

Hint : In the standard mode, the `exct` keyword controls the number of eigenvectors to be calculated.

Solution : See `samples/tutorial-1.7/solution.py`. This script firstly generates the input file for calculating the ground state and the first excited state of the $L = 8$ AFH chain with the LOBCG method, next invokes `HPhi`, then reads the vectors, and finally calculates the norms and the inner product.

CHAPTER

TWO

TUTORIAL FOR FINITE-TEMPERATURE CALCULATIONS

2.1 Heisenberg chain (finite temperatures)

Here, we study the finite-temperature properties of spin 1/2 Heisenberg model on the chain.

$$H = J \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j \tag{2.1}$$

The input file (`samples/tutorial_2.1/stan1.in`) for 12-site Heisenberg model is as follows:

```
L       = 12
model   = "Spin"
method  = "FullDiag"
lattice = "chain"
J = 1
2Sz = 0
2S  = 1
```

You can execute HPhi as follows

```
HPhi -s stan.in
```

2.1.1 Full diagonalization

After executing the full diagonalization, all the eigen energies are output in **output/Eiegenvalue.dat**. By using the python script (**``samples/tutorial_2.1/Finite.py``**), you can obtain the temperature dependence of the energy and the specific heat.

You can execute **Finite.py** as follows

```
python3 Finite.py
```

Then, you can obtain **FullDiag.dat** as follows

```
0.000100  -5.3873909174000003    0.0000000000000000
0.000150  -5.3873909174000003    0.0000000000000000
0.000200  -5.3873909174000003    0.0000000000000000
0.000250  -5.3873909174000003    0.0000000000000000
0.000300  -5.3873909174000003    0.0000000000000000
```

The 1st row represents temperature, 2nd row represents the energy, and the 3rd row represents the specific heat defined by $C = (\langle E^2 \rangle - \langle E \rangle^2)/T^2$.

2.1.2 TPQ method (Sz=0)

By selecting method as "TPQ", you can perform the finite-temperature calculations using the TPQ method.

The input file (`samples/tutorial_2.1/stan2.in`) for 12-site Heisenberg model is as follows:

```
L       = 12
model   = "Spin"
method  = "TPQ"
lattice = "chain"
J = 1
2Sz = 0
2S  = 1
```

After performing the TPQ calculations, results are output in **output/SS_rand*.dat**. By using the python script (``**samples/tutorial_2.1/AveSSrand.py**``), you can obtain the temperature dependence of physical quantities such as the energy and the specific heat.

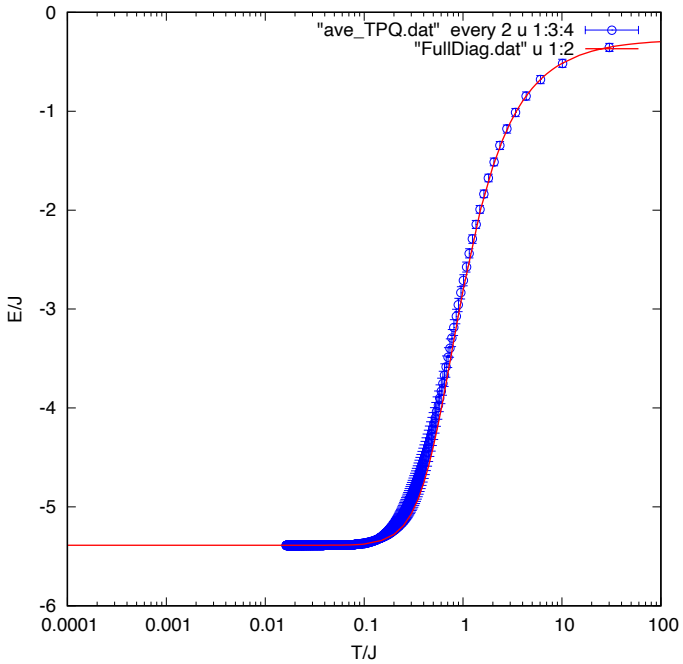
Then, you can obtain **ave_TPQ.dat** as follows

```
# temperature T    err of T  Energy E    error of E  specific heat C  err of C
30.17747           0.02022   -0.35489    0.04044     0.00264          8.1126-05
15.10875           0.01016   -0.43500    0.04065     0.01068          0.0003182
10.08598           0.00681   -0.51592    0.04086     0.02423          0.0007030
7.574693           0.00513   -0.59754    0.04106     0.04335          0.0012311
6.067978           0.00412   -0.67978    0.04123     0.06808          0.0019053
```

Using gnuplot, you can directly compare the two results

```
plot "FullDiag.dat" u 1:2 w l,"ave_TPQ.dat" u 1:3:4 w e
```

You can see the following output image.



2.1.3 TPQ method (susceptibility)

By using the TPQ method, it is also possible to calculate the spin susceptibility by performing the calculations for all Sz sectors.

The input file (`samples/tutorial_2.1/stan3.in`) for 12-site Heisenberg model is as follows:

```
L       = 12
model   = "SpinGC"
method  = "TPQ"
lattice = "chain"
J = 1
2S  = 1
```

Here, note that "model = Spin" is changed to "model = SpinGC" and "Sz = 0" is omitted.

After performing the TPQ calculations, temperature dependence of several physical quantities such as number of particel N and total Sz are output in **output/Flct_rand*.dat**. By using the python script (**``samples/tutorial_2.1/AveFlct.py``**), you can obtain the temperature dependence of physical quantities such as the energy and the spin susceptibility χ. Note that χ is defined as

$$\begin{aligned} \chi &= \frac{\langle m_z^2 \rangle - \langle m_z \rangle^2}{T} \\ m_z &= \sum_i S_i^z. \end{aligned} \tag{2.2}$$

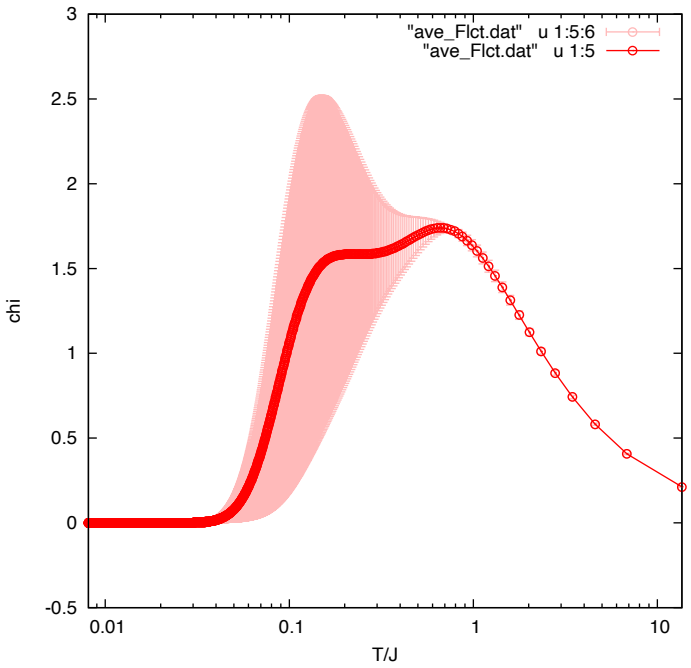
Then, you can obtain **ave_Flct.dat** as follows

```
# temperature T    err of T  m_z         error of m_z susceptibility chi  err of chi
13.5876            0.00695   -0.00688    0.00955      0.21243             0.00184
6.83615            0.00373   -0.00783    0.01067      0.40632             0.00414
4.58603            0.00278   -0.00894    0.01251      0.58234             0.00694
3.46113            0.00239   -0.01023    0.01474      0.74129             0.01021
2.78624            0.00220   -0.01171    0.01713      0.88407             0.01383
```

Using gnuplot, you can see the temperature dependence of χ

```
se log x
se colors classic
se xlabel "T/J"
se ylabel "chi"
plot    "ave_Flct.dat"   u 1:5:6 w e lc rgb "#FFBBBB" ps 1 pt 6,\
        "ave_Flct.dat"   u 1:5 w lp lt 1 ps 1 pt 6
```

You can see the following output image.



2.2 Kitaev cluster (finite temperatures)

In this subsection, we will examine the convergence of the sampling in the TPQ method. Here, we study a 12 site cluster of a typical magnetically frustrated magnet, the Kitaev model, whose heat capacity shows non-trivial temperature dependence.

The Kitaev model is a kind of the Ising model with bond-orientation dependent Ising interactions as follows:

$$H = K \sum_{\gamma=x,y,z} \sum_{\langle i,j \rangle_\gamma} S_i^\gamma S_j^\gamma, \tag{2.3}$$

where the index $\gamma = x, y, z$ specifies the bond direction of the nearest-neighbor site pair $\langle i, j \rangle_\gamma$.

2.2.1 Lattice structure and eigen states of the Kitaev cluster

A 12 site cluster of the Kitaev model is generated by the following standard input file (`samples/tutorial_2.2/stan1.in`):

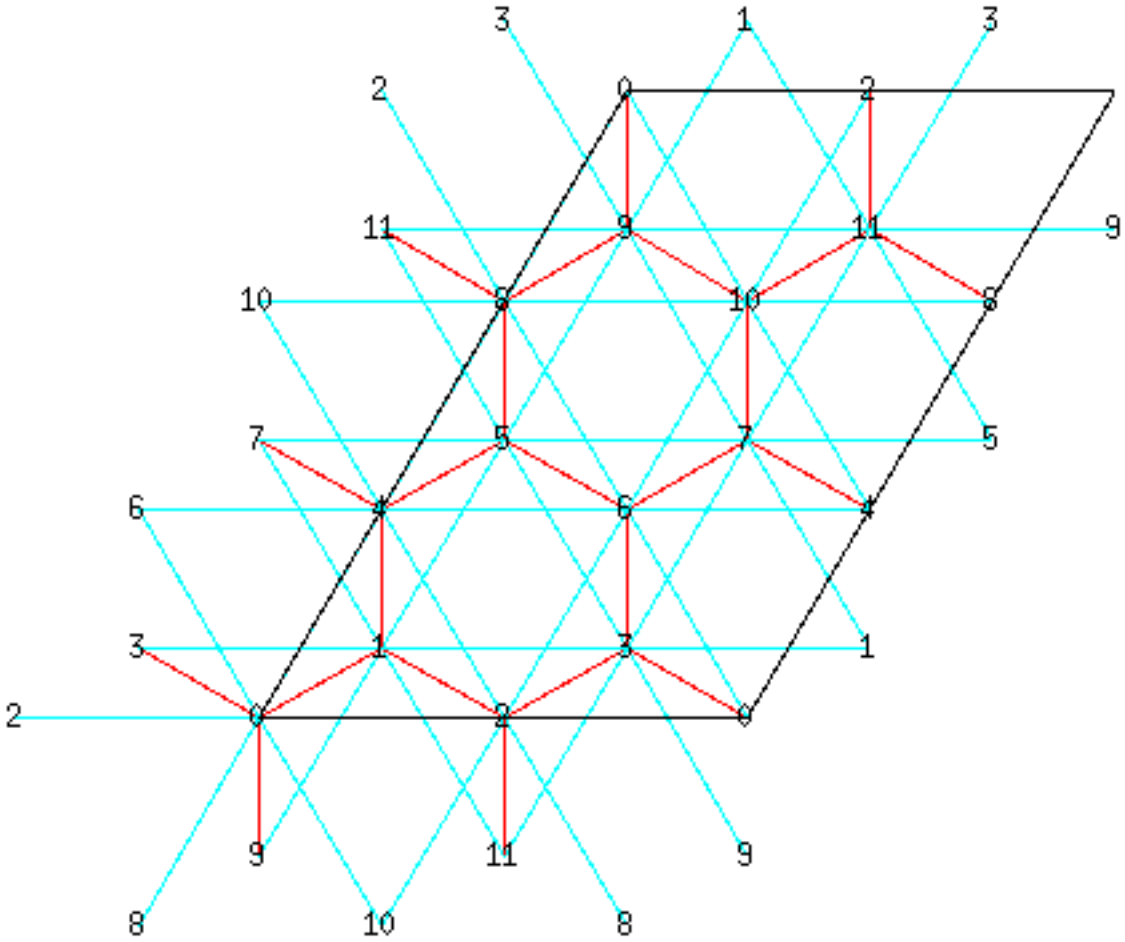
```
W = 2
L = 3
model = "SpinGC"
method = "CG"
lattice = "Honeycomb"
J0x = -1.0
J0y =  0.0
J0z =  0.0
J1x =  0.0
J1y = -1.0
J1z =  0.0
J2x =  0.0
```

(continues on next page)

(continued from previous page)

```
J2y =  0.0
J2z = -1.0
2S=1
exct = 8
```

You can visualize the cluster by plotting **lattice.gp** as shown in the figure below. Here the x bond ($\langle i,j\rangle_x$) connects, for example, the 0th and 1st sites ($i=0, j=1$), while the y bond ($\langle i,j\rangle_y$) connects, for example, the 1st and 2nd sites ($i=1, j=2$). The :math: z bond ($\langle i,j\rangle_z$) connects the 1st and 4th sites for example ($i=1, j=4$).



By using the above standard input, we will obtain **zvo_energy.dat** in your **output** directory:

```
State 0
 Energy  -2.4500706750607750
 Doublon  0.0000000000000000
 Sz  0.0000000000333962

State 1
 Energy  -2.4500706750607750
 Doublon  0.0000000000000000
 Sz  -0.0000000000081852
```

(continues on next page)

(continued from previous page)

```
State 2
 Energy  -2.4500706750607804
 Doublon  0.0000000000000000
 Sz  -0.0000000000025177

State 3
 Energy  -2.4500706750607777
 Doublon  0.0000000000000000
 Sz  -0.0000000000018603

State 4
 Energy  -2.3427788601414870
 Doublon  0.0000000000000000
 Sz  0.0000000000982415

State 5
 Energy  -2.3427788601414870
 Doublon  0.0000000000000000
 Sz  -0.0000000000217449

State 6
 Energy  -2.3427788601414830
 Doublon  0.0000000000000000
 Sz  -0.0000000000488295

State 7
 Energy  -2.3427788601414847
 Doublon  0.0000000000000000
 Sz  0.0000000001057668

State 8
 Energy  -2.3237385032276898
 Doublon  0.0000000000000000
 Sz  -0.0647000851410590

State 9
 Energy  -2.3237385032277440
 Doublon  0.0000000000000000
 Sz  0.4412235001577003
```

2.2.2 Convergence of the microcanonical TPQ

Then, we will examine the convergence of the microcanonical TPQ (mTPQ) method introduced by S. Sugiura and A. Shimizu, which is implemented in H Φ for **method = "TPQ"**.

By using the following standard input file, you can generate 64 mTPQ samples (`samples/tutorial_2.2/stan2a.in`):

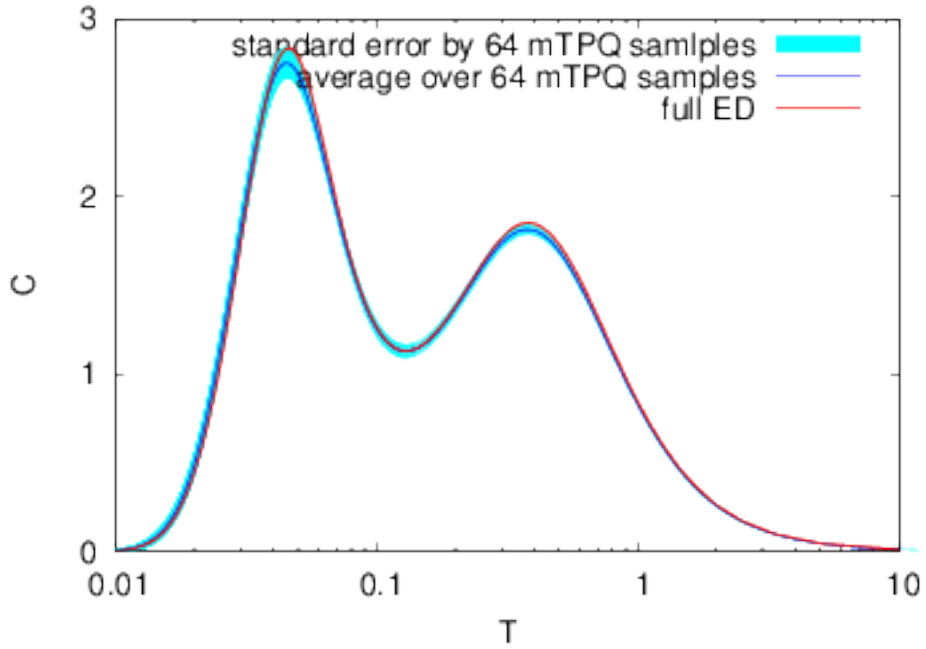
```
W = 2
L = 3
model = "SpinGC"
method = "TPQ"
lattice = "Honeycomb"
J0x = -1.0
J0y =  0.0
```

(continues on next page)

(continued from previous page)

```
J0z =  0.0
J1x =  0.0
J1y = -1.0
J1z =  0.0
J2x =  0.0
J2y =  0.0
J2z = -1.0
2S=1
Lanczos_max = 3000
LargeValue = 4.0
NumAve = 64
```

Here, you may use larger **Lanczos_max** and **LargeValue** than those in the default setting.



Then, you can obtain the above temperature dependence of the heat capacity. The average of C over the 64 mTPQ samples is shown by the blue curve while C obtained by the full diagonalization is shown by the red curve. The grey belt shows the standard deviation $\sqrt{\sigma^2}$, and the cyan belt shows the standard error $\sigma_E = \sqrt{\sigma^2/N_{\rm sample}}$, where $N_{\rm sample}$ is the sample size ($N_{\rm sample} = 64$). Although the systematic errors in the estimation of temperatures exist in the mTPQ formalism, the exact hat capacity (the red curve in the above figure) is within the random sample distribution (estimated by $\sqrt{\sigma^2}$ and shown by the grey belt).

For obtaining the results by the full diagonalization, please use the following input file ((`samples/tutorial_2.2/stan2b.in`))

```
W=2
L=3
model = "SpinGC"
method = "fulldiag"
lattice = "Honeycomb"
J0x = -1.0
J0y = 0.0
J0z = 0.0
J1x = 0.0
```

(continues on next page)

(continued from previous page)

```
J1y = -1.0
J1z = 0.0
J2x = 0.0
J2y = 0.0
J2z = -1.0
2S=1
```

Note that it takes several minute because the dimension of matrix is large (dimension is 4900). As in the Heisenberg chain, by using the python script (**(``samples/tutorial_2.2/Finite.py``)**), you can obtain the temperature dependence of the energy and the specific heat.

CHAPTER

THREE

TUTORIALS FOR REAL-TIME EVOLUTION

3.1 U quench in Hubbard model

Let's solve the following U quench in 2D Hubbard model at half filling.

$$H(\tau) = -t \sum_{\langle i,j\rangle,\sigma} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{H.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow} + U_{\text{quench}} h(\tau) \sum_{i} n_{i\uparrow} n_{i\downarrow} \tag{3.1}$$

$h(\tau)$ means the step function. The input files (`samples/tutorial_3.1/stan1.in` and `samples/tutorial_3.1/stan2.in`) are as follows

```
stan1.in

model = "Hubbard"
method = "CG"
lattice = "square"
a0W =  2
a0L =  2
a1W =  2
a1L = -2
2Sz = 0
t = 1.0
U = 4.0
nelec = 8
EigenvecIO = "out"
```

```
stan2.in

model = "Hubbard"
method = "Time-Evolution"
lattice = "square"
a0W =  2
a0L =  2
a1W =  2
a1L = -2
2Sz = 0
t = 1.0
U = 4.0
nelec = 8
PumpType = "Quench"
Uquench = -8
EigenvecIO = "in"
dt = 0.01
lanczos_max = 1000
```

You can execute HPhi as follows

```
HPhi -s stan1.in
HPhi -s stan2.in
```

3.1.1 Check norm and energy

Unitary dynamics of the norm of a wavefunction should be conserved during the real-time evolution. Using gnuplot, check the dynamics of the norm for this problem

```
plot "output/Norm.dat" u 1:2 w l
```

In sudden U-quench simulations, the total energy should be conserved for $\tau > 0$. Using gnuplot, check whether the energy is conserved during the real-time evolution

```
plot "output/SS.dat" u 1:2 w l
```

3.1.2 Dynamics of double occupation

The double occupation $D = \sum_i \langle n_{i\uparrow} n_{i\downarrow} \rangle$ is not conserved because $[D, H] \neq 0$. You can check the dynamics of D by executing the following command on gnuplot

```
plot "output/SS.dat" u 1:4 w l
```

3.2 Dynamical phase transition in 1D transverse-field Ising model

Dynamical phase transition (DPT) is a phase transition due to nonequilibrium process such as sudden quench. In this section, we introduce the DPT in 1D transverse Ising model, which is one of the famous examples for the DPT. The details are found in this paper (M. Heyl, A. Polkovnikov, and S. Kehrein, Phys. Rev. Lett. **110**, 135704 (2013)).

$$H = J \sum_{\langle i,j \rangle} S_i^z S_j^z + \Gamma \sum_i S_i^x \tag{3.2}$$

3.2.1 Ground state

First, you need to obtain the initial state for simulations of the real-time dynamics. Please make the following input file (`samples/tutorial_3.2/stan1.in`)

```
model = "SpinGC"
method = "CG"
lattice = "chain"
L = 12
Jz = -1.0
Gamma = 0.1
h = 1e-5
EigenvecIO = "out"
```

and run the following command

```
HPhi -s stan1.in
```

Check the total magnetization along z-direction $M_z = \sum_i \langle S_i^z \rangle$ in "output/zvo_energy.dat".

Question: If a longitudinal magnetic field h in stan1.in becomes 0, what happens?

3.2.2 Γ quench

After obtaining the ground state, you can perform simulations for Γ quench in the 1D transverse-field Ising model. Make the following input file (`samples/tutorial_3.1/stan2.in`)

```
model = "SpinGC"
method = "Time-Evolution"
lattice = "chain"
L = 12
Jz = -1.0
Gamma = 0.2
h = 1e-5
EigenvecIO = "in"
dt = 0.01
lanczos_max = 1000
```

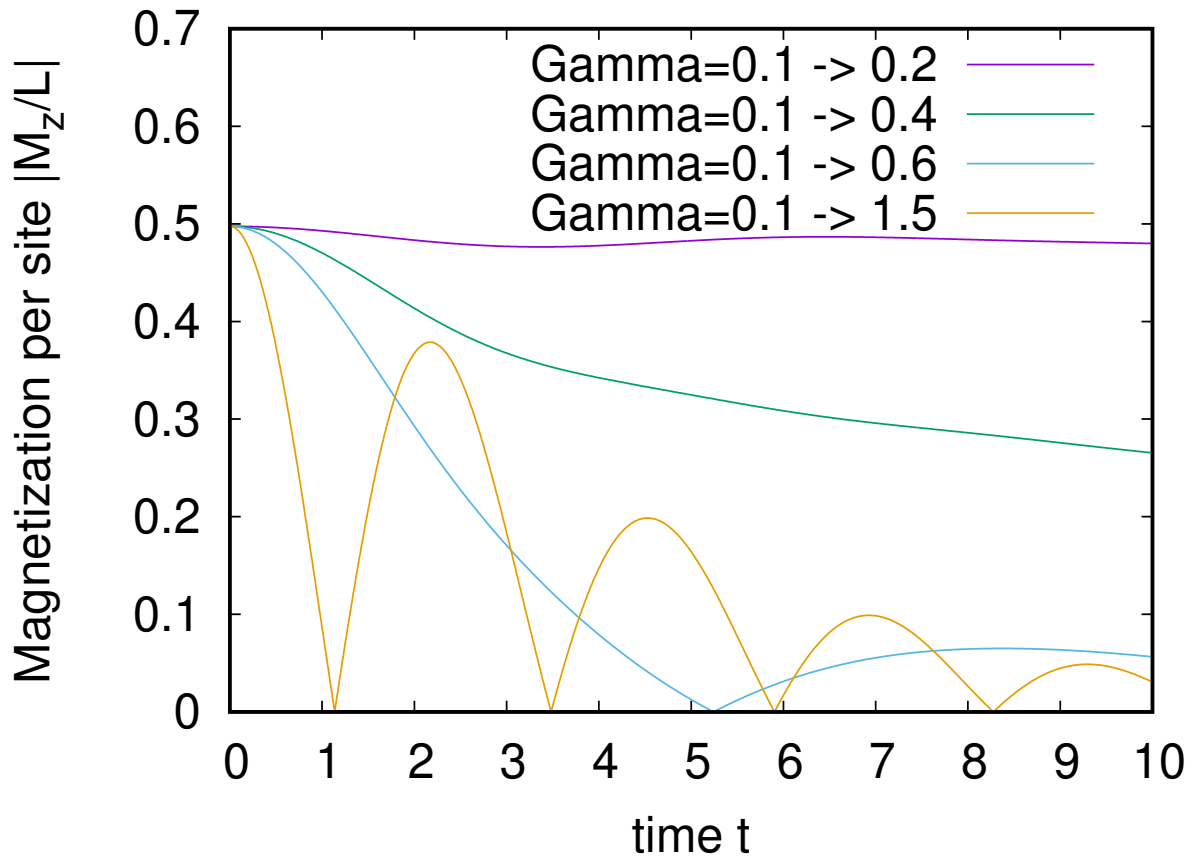
In this case, Γ quench from 0.1 to 0.2 will be performed by executing the following command

```
HPhi -s stan2.in
```

Now you can check the real-time evolution of M_z in "output/Flct.dat". The result is plotted by executing the following command on gnuplot

```
gnuplot
gnuplot> set xlabel "time t"
gnuplot> set ylabel "Magnetization per site |M_z/L|"
gnuplot> p "output/Flct.dat" u 1:(abs($6/12)) w l tit "Gamma=0.1 -> 0.2"
```

One of the features of DPT in this model is that cusp structures in the dynamics of M_z appears at the same intervals. The following figure is an example for several results for Γ quench. You can see the cusp structure for $\Gamma > 0.5$.


Magnetization per site |M_Z/L|
time t
Gamma=0.1 -> 0.2
Gamma=0.1 -> 0.4
Gamma=0.1 -> 0.6
Gamma=0.1 -> 1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1
2
3
4
5
6
7
8
9
10

CHAPTER

FOUR

TUTORIAL FOR CALCULATIONS OF DYNAMICAL PROPERTIES

4.1 Dynamical spin structure factor

Let's solve the following Hubbard model on the square lattice.

$$H = -t \sum_{\langle i,j \rangle, \sigma} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{H.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow} \tag{4.1}$$

The input files (`samples/tutorial_4.1/stan1.in` and `samples/tutorial_4.1/stan2.in`) for 8-site Hubbard model are as follows

```
stan1.in

a0W = 2
a0L = 2
a1W = -2
a1L = 2
model = "hubbard"
method = "CG"
lattice = "square"
t = 1.0
t' = 0.5
U = 4.0
2Sz = 0
nelec = 8
EigenvecIO = "out"
```

```
stan2.in

a0W = 2
a0L = 2
a1W = -2
a1L = 2
model = "hubbard"
method = "CG"
lattice = "square"
t = 1.0
t' = 0.5
U = 4.0
2Sz = 0
nelec = 8
LanczosEPS = 8
CalcSpec = "Normal"
```

(continues on next page)

(continued from previous page)

```
SpectrumType = "SzSz"
SpectrumQW = 0.5
SpectrumQL = 0.5
OmegaMin = -10.0
OmegaMax = 20.0
OmegaIM = 0.2
OmegaOrg = 10.0
```

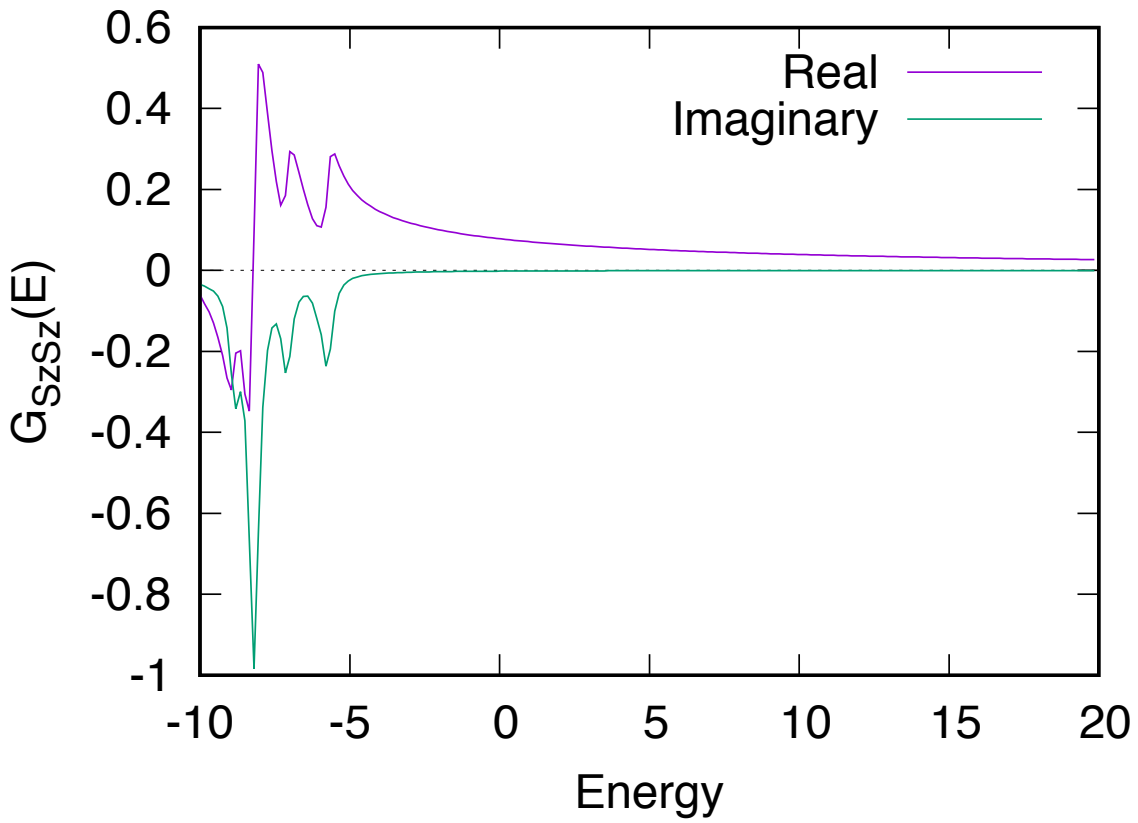
You can execute HPhi as follows

```
HPhi -s stan1.in
HPhi -s stan2.in
```

After finishing calculations, the spectrum $G_{S_zS_z}(\mathbf{Q} \equiv (\pi, \pi), \omega) = \langle S_z(-\mathbf{Q}) \left[H - \omega - \omega_0 + i\eta\right]^{-1} S_z(\mathbf{Q})\rangle$ is outputted in *output/zvo_DynamicalGreen.dat*. Here, $S_z(\mathbf{Q}) = \sum_i e^{i\mathbf{Q}\cdot\mathbf{r}_i} S_z^i$ and the frequency ω moves from -10 to 10, $\omega_0 = 10$, and η is set as 0.2. You can check the result by executing the following command on gnuplot:

```
gnuplot
gnuplot> set xlabel "Energy"
gnuplot> set ylabel "G_{SzSz}(E)"
gnuplot> set xzeroaxis
gnuplot> plot "output/zvo_DynamicalGreen.dat" u 1:3 w l tit "Real", \
> "output/zvo_DynamicalGreen.dat" u 1:4 w l tit "Imaginary"
```

You can see the following output image.



4.2 Hubbard chain (optical conductivity)

Here, we calculate the optical conductivity for the one-dimensional Hubbard model.

The optical conductivity $\sigma(\omega)$ can be calculated from the current-current correlation $I(\omega, \eta)$, which is defined as

$$\begin{aligned} j_x &= i\sum_{i,\sigma}(c^\dagger_{\mathbf{r}_i+\mathbf{e}_x,\sigma}c_{\mathbf{r}_i,\sigma} - c^\dagger_{\mathbf{r}_i,\sigma}c_{\mathbf{r}_i+\mathbf{e}_x,\sigma}), \\ I(\omega,\eta) &= \mathrm{Im}\Big[\langle 0|j_x[H-(\omega-E_0-i\eta)I]^{-1}j_x|0\rangle\Big], \end{aligned} \tag{4.2}$$

where $\mathbf{e}_x$ is the unit translational vector in the x direction. From this the regular part of the optical conductivity is defined as

$$\sigma_{\mathrm{reg}}(\omega) = \frac{I(\omega,\eta)+I(-\omega,-\eta)}{\omega N_s}, \tag{4.3}$$

where N_s is the number of sites.

An input file (`samples/tutorial_4.2/stan1.in`) for 6-site Hubbard model is as follows:

```
model = "Hubbard"
method = "CG"
lattice = "chain"
L = 6
t = 1
U = 10
2Sz = 0
nelec = 6
exct = 1
EigenVecIO  = "out"
```

Scripts for calculating the optical conductivity are available at ``**samples/tutorial_4.2/**``.

By performing the all-in-one script (All.sh),

```
sh ./All.sh
```

you can obtain **optical.dat**. Note that `samples/tutorial_4.2/OpticalSpectrum.py`,``samples/tutorial_4.1/lattice.py``, `samples/tutorial_4.2/lattice.py`, and `samples/tutorial_4.2/input.txt` are necessary.

A way for plotting **optical.dat** is as follows

```
plot "optical.dat" u 1:(-($4+$8)/$1) w l
```

4.3 Spectrum calculation for 12-site one-dimeinsional Heisenberg chain model.

The Hamiltonian is given by

$$H = J\sum_{i=0}^{11}\mathbf{S}_i\cdot\mathbf{S}_{i+1}, \tag{4.4}$$

where

$$\mathbf{S}_0 = \mathbf{S}_{12}. \tag{4.5}$$

The spectrum function can be calculated by following steps.

1. Calculate the ground state.
2. Define excitation operators in the `pair.def` file.
3. Calculate spectrum function.

See the manual for details. To simply do above steps, we prepare the script file `samples/tutorial_4.3/spinchain_example.py` .. in https://github.com/issp-center-dev/HPhi-gallery/tree/master/Spin/HeisenbergSpectrum. In the following, we show the procedure to obtain the specrum function by using the script file.

1. Execute the script file (`spinchain_example.py`)

```
$ python spinchain_example.py
```

2. Plot `spectrum.dat` by gnuplot.

```
$ gnuplot
$ set yrange [0:5]
$ set pm3d map
$ splot "./spectrum.dat" using 1:2:3
```

You can see the following figure, where horizontal and vertical axises correspond to the index of wave vector and frequency, respectively.