The simple mode of DSQSS/DLA

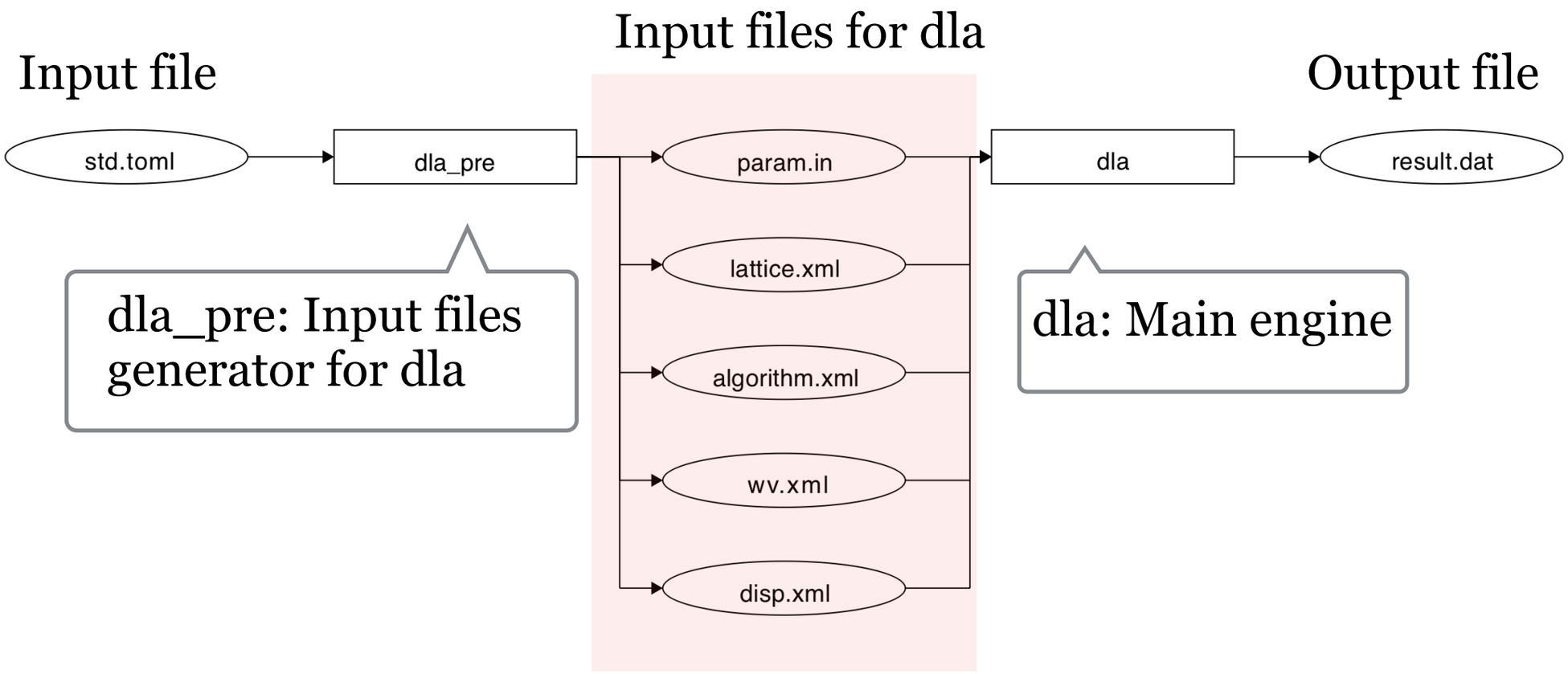
ISSP, UTokyo Kazuyoshi Yoshimi
2022-12-14 @ ISSP

Sample files are available from
https://www.pasums.issp.u-tokyo.ac.jp/dsqss/doc-presentation.

The simple mode of DSQSS/dla

- Users can simulate of a predefined model
  on a predefined lattice from one text file (std.toml file).

Schematic calculation flow of dla



Detail of input file

1. [parameter] section
A table specifying simulation parameters such as the inverse temperature.

2. [hamiltonian] section
A table specifying information of Hamiltonian.

3. [lattice] section
A table specifying information of lattice.

4. [kpoints] section
A table specifying information of wavevectors.

5. [algorithm] section
A table specifying algorithm for calculating scattering probability of wormheads

See details in Sec. 4.1 「Simple mode of DSQSS/DLA」.

Detail of input file

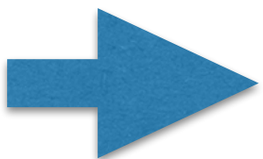
1. [parameter] section
A table specifying simulation parameters such as the inverse temperature.

2. [hamiltonian] section
A table specifying information of Hamiltonian.

3. [lattice] section
A table specifying information of lattice.

4. [kpoints] section
A table specifying information of wave vectors.

5. [algorithm] section
A table specifying algorithm for calculating scattering probability of wormheads



See details in Sec. 4.1 「Simple mode of DSQSS/DLA」.

Detail of input file (1)
Hamiltonian

XXZ model for arbitrary S

$$\mathcal{H} = -\sum_{\langle i,j \rangle} \left[J_z S_i^z S_j^z + \frac{J_{xy}}{2}(S_i^+ S_j^- + S_i^- S_j^+) \right] + D\sum_i (S_i^z)^2 - h\sum_i S_i^z$$

The value of S on a site can be specified by the keyword M (=$2S$) in the input file.

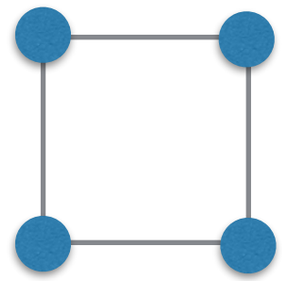
Bose-Hubbard model

$$\mathcal{H} = -\sum_{\langle i,j \rangle} \left[t(b_i^\dagger b_j + \text{h.c.}) + V n_i n_j \right] + \sum_i \left[\frac{U}{2} n_i(n_i - 1) - \mu n_i \right]$$

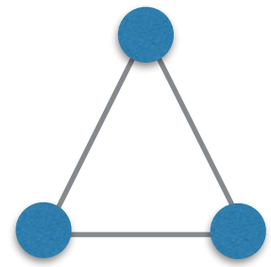
The cutoff of the number of particles on a site can be specified by the keyword M in the input file.

Detail of input file (2) Lattice

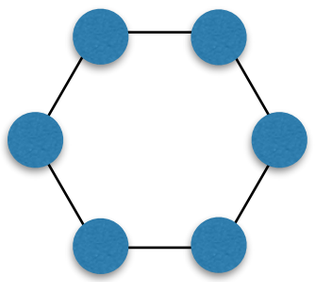
1. hypercubic
A hyper cubic lattice with arbitrary dimension.
By using `bc`, users can generate ladder or slab lattices.



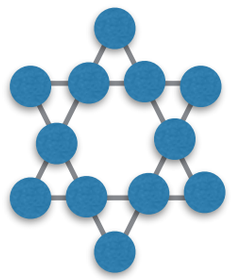
2. triangular
A two dimensional triangular lattice.



3. honeycomb
A two dimensional honeycomb lattice.



4. kagome
A two dimensional kagome lattice.



More complex lattice / Hamiltonian → Standard mode

Detail of measurements (1)

Output in main output file indicated by keyword “output”

`sign`

The sign of the weights.

$$\sum_i W_i / \sum_i |W_i$$

`anv`

The mean number of the vertices.

$$\frac{\langle N_v \rangle}{N_s}$$

`ene`

The energy density (energy per site)

$$\epsilon \equiv \frac{1}{N_s}(E_0 - T\langle N_v \rangle)$$

`spe`

The specific heat

$$C_V \equiv \frac{\partial \epsilon}{\partial T}$$

Detail of measurements (2)

Output in main output file indicated by keyword “output”

`xmx`

The transverse susceptibility

`amzu`

The “magnetization” (uniform, $\tau = 0$).

$$\langle m^z \rangle \text{ , where } m^z \equiv \frac{1}{N_s} \sum_i^{N_s} M_i^z$$

`bmzu`

The “magnetization” (uniform, average over τ). $\langle \tilde{m}^z \rangle$.

`smzu`

The structure factor (uniform).

$$S^{zz}(\vec{k}=0) \equiv \frac{1}{N_s} \sum_{i,j} e^{i\vec{k}\cdot(\vec{r}_i - \vec{r}_j)} \left[\left\langle M_i^z M_j^z \right\rangle - \left\langle M_i^z \right\rangle \left\langle M_j^z \right\rangle \right] \Big|_{\vec{k}=0} = N_s \left[\left\langle (m^z)^2 \right\rangle - \langle m^z \rangle^2 \right]$$

`xmzu`

The longitudinal susceptibility (uniform).

$$\chi^{zz}(\vec{k}=0, \omega=0) \equiv \frac{\partial \langle \tilde{m}^z \rangle}{\partial h} = \beta N_s \left[\left\langle \left(\tilde{m}^z \right)^2 \right\rangle - \langle \tilde{m}^z \rangle^2 \right]$$

Detail of measurements (3)

Output in main output file indicated by keyword “output”

`amzsK`

The “magnetization” (“staggered”, $\tau = 0$)

$$\langle m_s^z \rangle \text{ where } m_K^z \equiv \frac{1}{N_s} \sum_i^{N_s} M_i^z \cos\left(\vec{k} \cdot \vec{r_i}\right) .$$

K is an index of wavevector k specified in the wavevector XML file.

`bmzu`

The “magnetization” (“staggered”, average over τ). $\langle \tilde{m}_K^z \rangle$.

`smzs`

The structure factor (“staggered”).

$$S^{zz}(\vec{k}) = N_s \left[\left\langle (m_K^z)^2 \right\rangle - \left\langle m_K^z \right\rangle^2 \right]$$

`xmzs`

The longitudinal susceptibility (“staggered”).

$$\chi^{zz}(\vec{k}, \omega = 0) = \beta N_s \left[\left\langle (\tilde{m}_K^z)^2 \right\rangle - \langle \tilde{m}_K^z \rangle^2 \right]$$

Detail of measurements (4)

Output in main output file indicated by keyword “output”

`wi2`

The winding number.

$$W^2 = \sum_{d=1}^{D} L_d^2 \left\langle W_d^2 \right\rangle$$

`rhos`

The superfluid density.

$$\rho_s = \frac{W^2}{2DV\beta}$$

`rhof`

The superfluid fraction.

$$\frac{\rho_s}{\langle m^z \rangle}$$

`comp`

The compressibility.

$$\frac{\chi^{zz}(\vec{k} = 0, \omega = 0)}{\langle \tilde{m}^z \rangle^2}$$

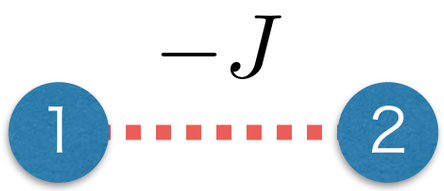
`time`

The time in a Monte Carlo sweep (in seconds.)

Tutorial: Heisenberg dimer (1)

- **Sample of simple mode file (sample/dla/01_spindimer/std.toml)**
  - **S=1/2 antiferromagnetic Heisenberg dimer**

$$\mathcal{H} = -J\boldsymbol{S}_1 \cdot \boldsymbol{S}_2$$




```
[hamiltonian]
model = "spin"
M =  1                  # S=1/2
Jz = -1.0               # coupling constant, negative for AF
Jxy = -1.0              # coupling constant, negative for AF
h = 0.0                 # magnetic field

[lattice]
lattice = "hypercubic" # hypercubic, periodic
dim = 1                 # dimension
L = 2                   # number of sites along each direction
bc = false              # open boundary
```

[hamiltonian] section

Specify information of Hamiltonian

[lattice] section

Specify information of lattice

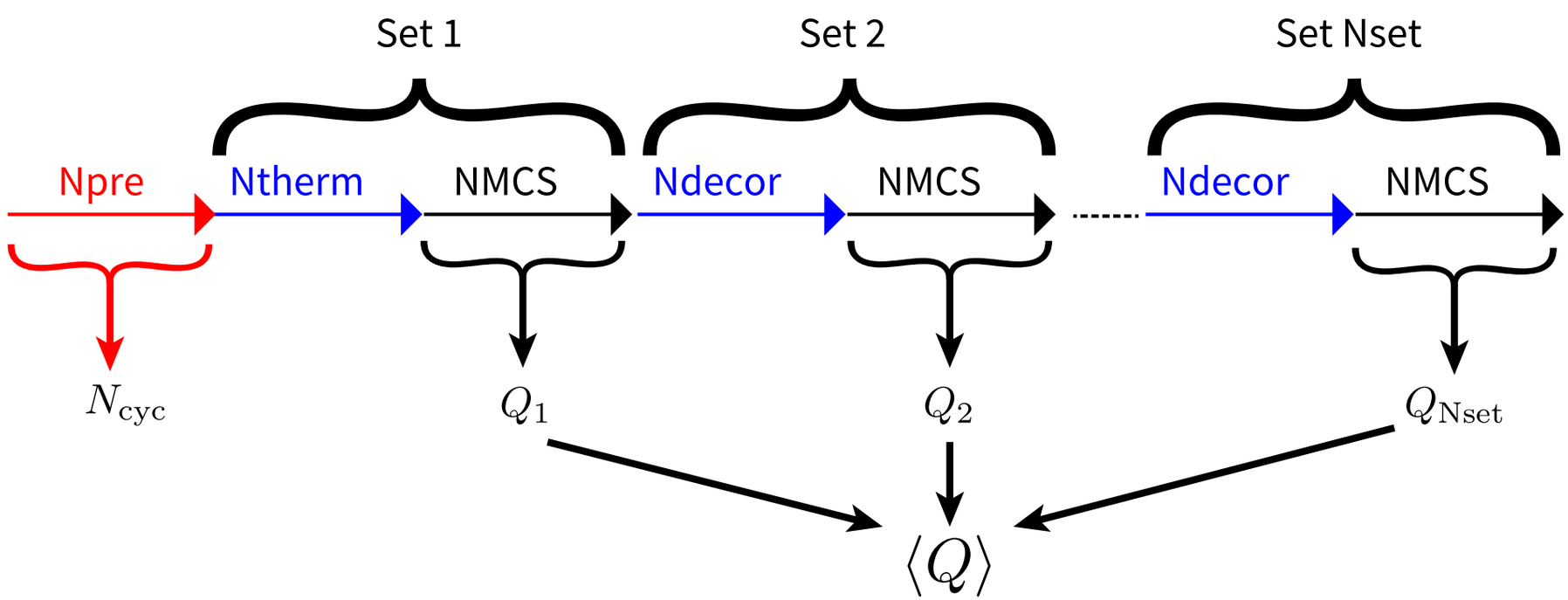
Tutorial: Heisenberg dimer (2)

- **Sample of simple mode file (sample/dla/01_spindimer/std.toml)**

```
[parameter]
beta = 100          # inverse temperature
nset = 5            # set of Monte Carlo sweeps
npre = 10           # MCSteps to estimate hyperparameter
ntherm = 10         # MCSweeps for thermalization
nmcs = 100          # MCSweeps for measurement
seed = 31415        # seed of RNG
```

[parameter] section

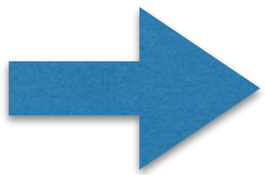
Specify simulation parameters



Tutorial: Heisenberg dimer (3)

- **Run the dla_pre**

  $ dla_pre std.toml



output

param.in, lattice.xml, algorithm.xml

- **Run the dla**

  $ dla param.in

Random number parallelization

$ mpiexec -np 4 dla param.in

Standard output (log)

```
>>> The program is being run with MPI mode.( N_PROC = 1 )
+++++++++ input data +++++++++
RUNTYPE = 0
...
+++++++++ input data +++++++++
Determining hyperparameter NCYC : 51
Start main calculation.
1 / 5 done. [Elapsed: 0.014687 sec. ETR: 0.058748 sec.]
2 / 5 done. [Elapsed: 0.195772 sec. ETR: 0.293658 sec.]
3 / 5 done. [Elapsed: 0.333755 sec. ETR: 0.222503 sec.]
4 / 5 done. [Elapsed: 0.471529 sec. ETR: 0.117882 sec.]
5 / 5 done. [Elapsed: 0.616055 sec. ETR: 0 sec.]
```

ETR: Estimated Time Remaining

Tutorial: Heisenberg dimer (4)

Output file: sample.log

```
I N_PROC = 1
P D      =          1
P L      =          2
...
P SIMULATIONTIME   =     0.000000
R sign = 1.00000000e+00 0.00000000e+00
R anv = 2.49300000e+01 2.96344394e-01
R ene = -3.74300000e-01 2.96344394e-03
R spe = 8.16400000e-02 1.47017825e+00
R som = 8.16400000e+00 1.47017825e+02
R len = 4.00127485e+00 2.92088648e-02
R xmx = 1.00031871e-02 7.30221620e-05
...
R time = 1.37698000e-06 1.20496230e-07
I [the maximum number of segments]         = 165
I [the maximum number of vertices]         = 84
I [the maximum number of reg. vertex info.] = 1
```

P <name> = <value>
Parameters read from the input files.

R <name> = <mean> <error>
Results of observables.
<mean> : the expected value
<error> : the statistical error of <mean>.

I <text> = <value>
Other information.

Main Results are written in [R] section.

- **Analyze the result**

  $ grep ene sample.log
  R ene = -3.74300000e-01 2.96344394e-03

Ground state energy
-3.75 (exact)

Tutorial: Magnetic Susceptibility of antiferromagnetic spin chains (1)

- **sample/dla/02_spinchain**
  - **S=1/2, 1 antiferromagnetic spin chains**

$$\mathcal{H} = -J \sum_{i=1}^{30} \boldsymbol{S}_i \cdot \boldsymbol{S}_{i+1}$$

```
[hamiltonian]
model = "spin"
M =  1                # S=1/2
Jz = -1.0             # coupling constant, negative for AF
Jxy = -1.0            # coupling constant, negative for AF
h = 0.0               # magnetic field

[lattice]
lattice = "hypercubic" # hypercubic, periodic
dim = 1               # dimension
L = 30                # number of sites along each direction
```

For S=1, M = 2

- **Calculate magnetic susceptibility at each beta**

Tutorial: Magnetic Susceptibility of antiferromagnetic spin chains (2)

- Sample script (exec.py)

```
import subprocess
from dsqss.dla_pre import dla_pre
from dsqss.result import Results
L = 30
lattice = {"lattice": "hypercubic", "dim": 1, "L": L}
hamiltonian = {"model": "spin", "Jz": -1, "Jxy": -1}
parameter = {"nset": 5, "ntherm": 1000, "ndecor": 1000, "nmcs": 1000}
name = "xmzu"
Ms = [1, 2]
Ts = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0]
for M in Ms:
    output = open("{0}_{1}.dat".format(name, M), "w")
    for i, T in enumerate(Ts):
        ofile = "res_{}_{}.dat".format(M, i)
        pfile = "param_{}_{}.in".format(M, i)
        hamiltonian["M"] = M
        parameter["beta"] = 1.0 / T
        parameter["outfile"] = ofile
        dla_pre(
            {"parameter": parameter, "hamiltonian": hamiltonian, "lattice": lattice},
            pfile,
        )
        cmd = ["dla", "param_{0}_{1}.in".format(M, i)]
        subprocess.call(cmd)
        res = Results(ofile)
        output.write("{} {}\n".format(T, res.to_str(name)))
    output.close()
```

Set parameters

Change beta

Run dla_pre

Run dla

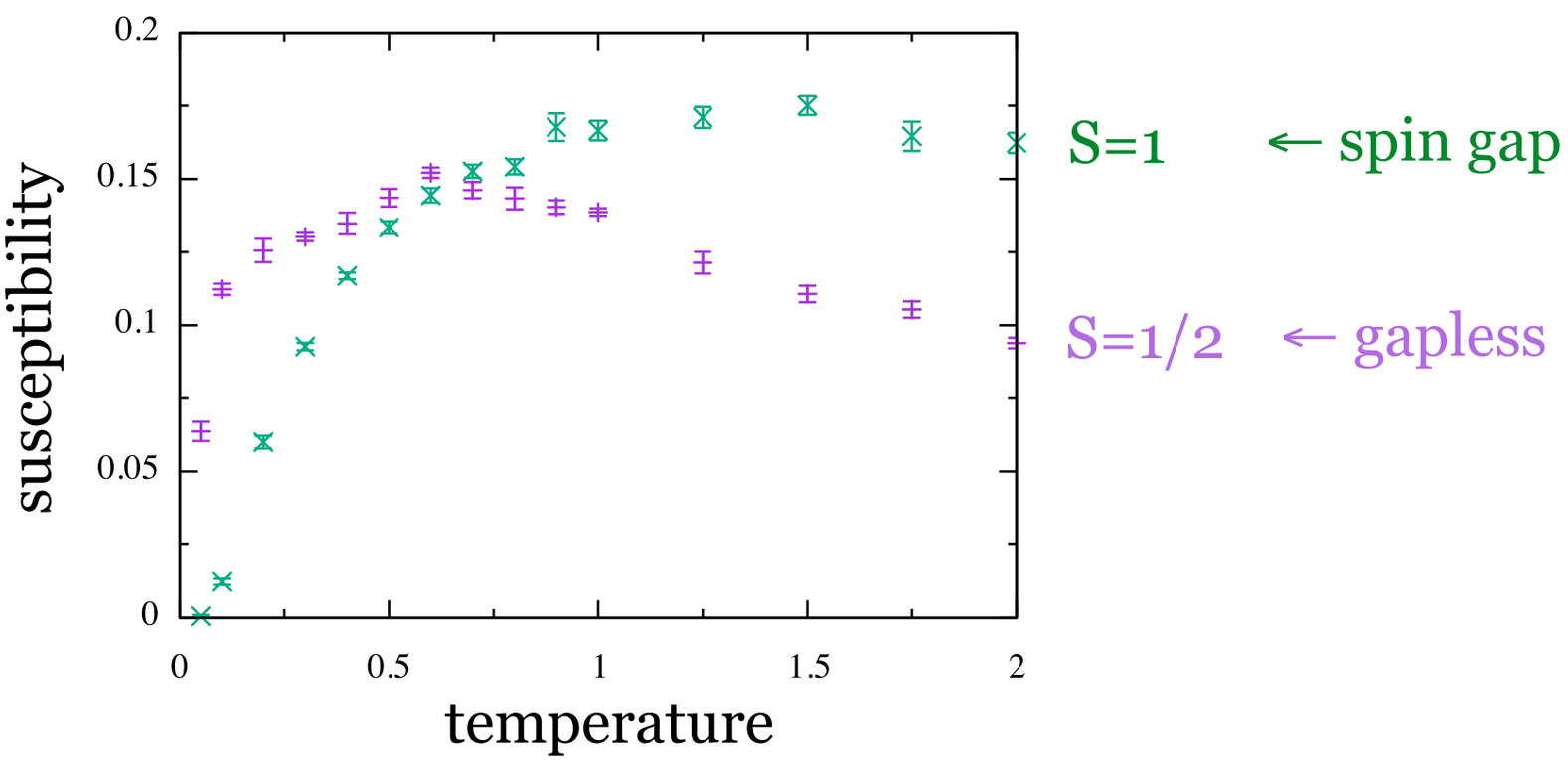
Read and output results

Tutorial: Magnetic Susceptibility of antiferromagnetic spin chains (3)

- **Run script**
  $ python3 exec.py

  xmzu_1.dat (S=1/2), xmzu_2.dat (S=1)

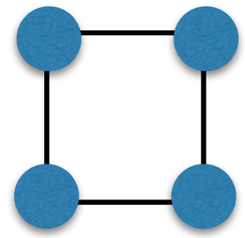
- **Plot results**
  $ gnuplot
  $ plot "./xmzu_1.dat" using 1:2:3 with errorbar, "./xmzu_2.dat" using 1:2:3 with errorbar



Tutorial: Number density of the hardcore Bosons on a square lattice (1)

- **sample/dla/03_bosesquare**
  Hardcore Bose-Hubbard model with the nearest neighbor repulsive on a 8×8 square lattice

$$\mathcal{H} = -t \sum_{\langle i,j \rangle} \left[b_i^\dagger b_j + b_j^\dagger b_i \right] + \sum_{\langle i,j \rangle} V n_i n_j$$



$-t, V$

```
[hamiltonian]
model = "boson"
M =  1                # The cutoff of the number of particles on a site
t = 1                 # The hopping parameter.
V = 3                 # The offsite interaction.

[lattice]
lattice = "hypercubic" # hypercubic, periodic
dim = 2                # dimension
L = [8, 8]             # number of sites along each direction
```

Tutorial: Number density of the hardcore Bosons on a square lattice (2)

- **Sample script (exec.py)**

```
import subprocess

from dsqss.dla_pre import dla_pre
from dsqss.result import Results

V = 3
L = [8, 8]
beta = 10.0

lattice = {"lattice": "hypercubic", "dim": 2, "L": L}
hamiltonian = {"model": "boson", "t": 1, "V": V, "M": 1}
parameter = {"beta": beta, "nset": 4, "ntherm": 100, "ndecor": 100, "nmcs": 100}

name = "amzu"
mus = [-4.0, -2.0, 0.0, 2.0, 2.5, 3.0, 6.0, 9.0, 9.5, 10.0, 12.0, 14.0]

output = open("{}.dat".format(name), "w")
for i, mu in enumerate(mus):
    ofile = "res_{}.dat".format(i)
    pfile = "param_{}.in".format(i)
    hamiltonian["mu"] = mu
    parameter["outfile"] = ofile
    dla_pre(
        {"parameter": parameter, "hamiltonian": hamiltonian, "lattice": lattice}, pfile
    )
    cmd = ["dla", pfile]
    subprocess.call(cmd)
    res = Results(ofile)
    output.write("{} {}\n".format(mu, res.to_str(name)))
output.close()
```

Set parameters

Change chemical potential

Run dla_pre

Run dla

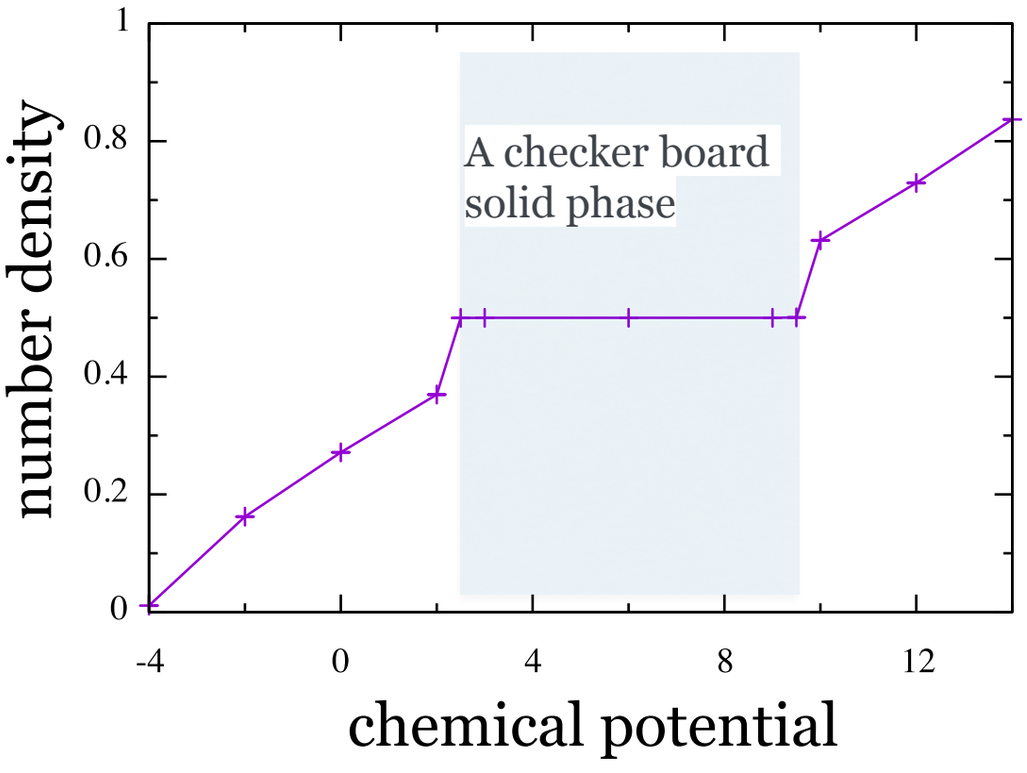
Read and output results

Tutorial: Number density of the hardcore Bosons on a square lattice (3)

- **Sample script (exec.py)**
- **Run script**
  $ python3 exec.py

  → **amzu.dat**
- **Plot results**

$ gnuplot
$ plot "./amzu.dat" using 1:2:3 with errorbar, "" using 1:2 w l lc 1



Tutorial: Number density of the hardcore Bosons on a square lattice (4)

- **Calculate structure factors**
  - **Add following keywords.**

```
[parameter]
wvfile = “wave.out” # A wavevector XML file.
[kpoints]
ksteps = [4, 4]        #Increments of wavenumber.
                       #If 0, half of lattice size instead of 0 is set.
```

L=[8, 8]
K = 0: (0,0)
K= 1: (π,0)
K= 2: (0,π)
K= 3: (π, π)

`xmzsK`

The longitudinal susceptibility

$$\chi^{zz}(\vec{k}, \omega = 0) = \beta N_s \left[\left\langle (\tilde{m}_K^z)^2 \right\rangle - \langle \tilde{m}_K^z \rangle^2\right]$$

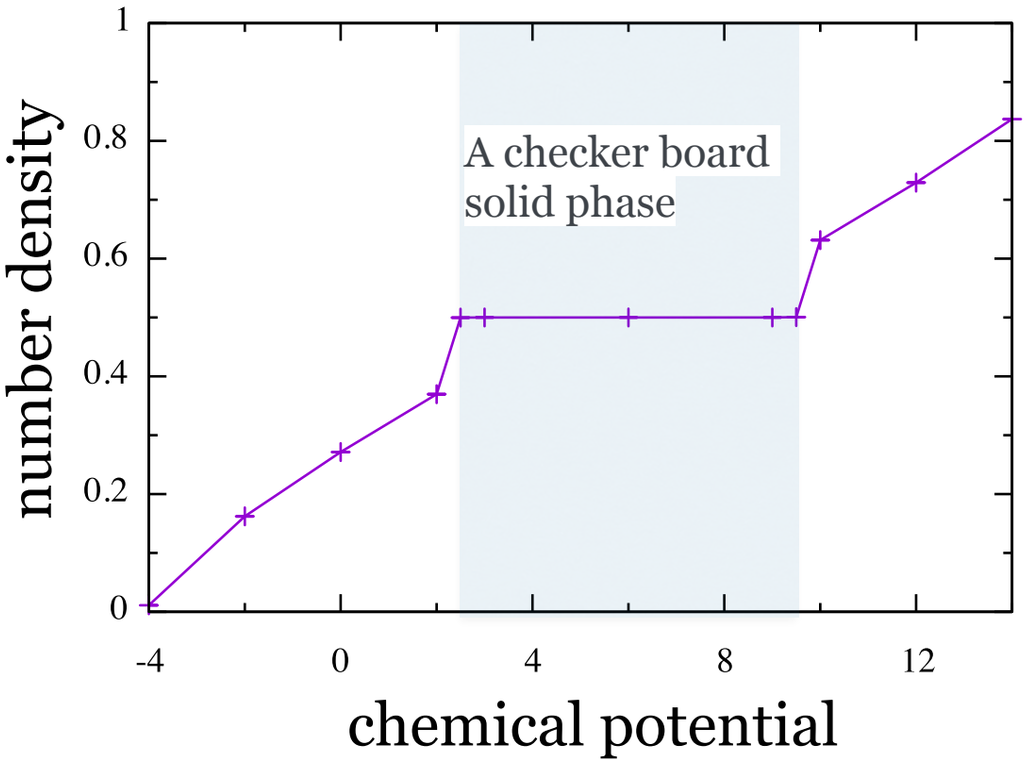
where $$m_K^z = \frac{1}{N_s} \sum_i^{N_s} M_i^z \cos(\boldsymbol{k} \cdot \boldsymbol{r}_i)$$

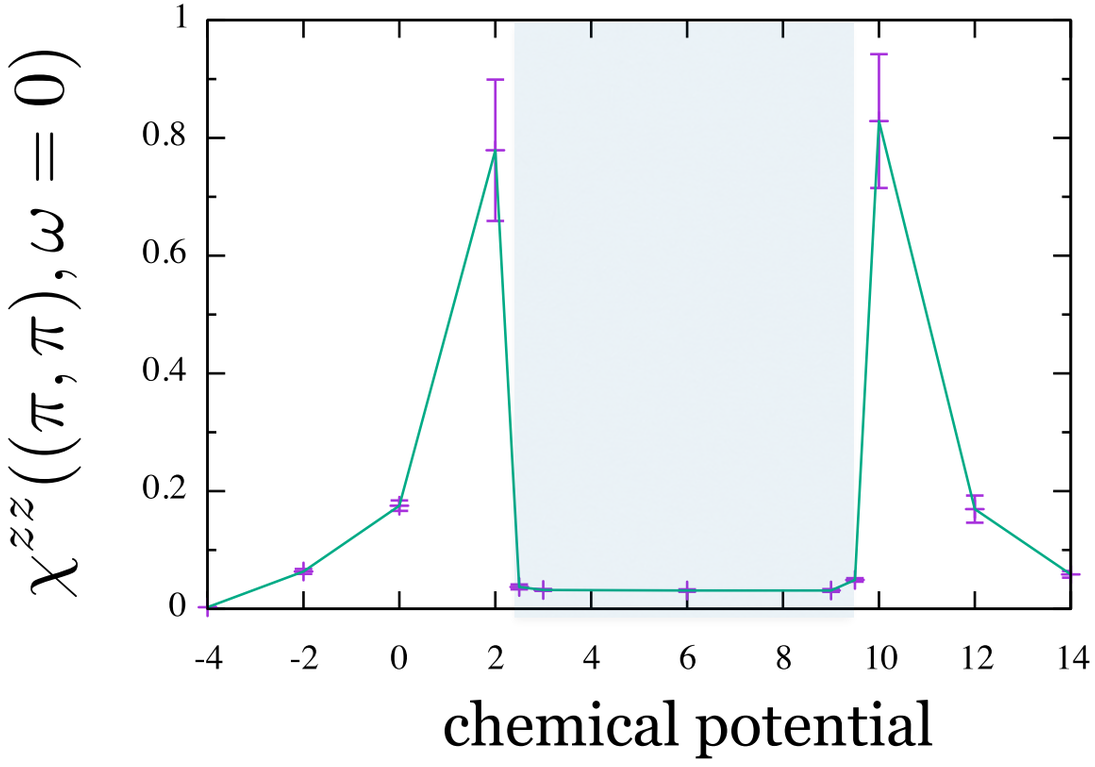
Tutorial: Number density of the hardcore Bosons on a square lattice (5)

- **Modify exec.py**

```
import subprocess

from dsqss.dla_pre import dla_pre
from dsqss.result import Results

V = 3
L = [8, 8]
beta = 10.0

lattice = {"lattice": "hypercubic", "dim": 2, "L": L}
hamiltonian = {"model": "boson", "t": 1, "V": V, "M": 1}
parameter = {"beta": beta, "nset": 4, "ntherm": 100, "ndecor": 100, "nmcs": 100}

name = “smzs3”
mus = [-4.0, -2.0, 0.0, 2.0, 2.5, 3.0, 6.0, 9.0, 9.5, 10.0, 12.0, 14.0]

output = open("{}.dat".format(name), "w")
for i, mu in enumerate(mus):
    ofile = "res_{}.dat".format(i)
    pfile = “param_{}.in”.format(i)
    wvfile = “wave_{}.out”.format(i)
    hamiltonian["mu"] = mu
    parameter["outfile"] = ofile
    parameter[“wvfile"] = wvfile
    dla_pre(
        {"parameter": parameter, "hamiltonian": hamiltonian, "lattice": lattice}, pfile
    )
    cmd = ["dla", pfile]
    # subprocess.call(cmd)
    res = Results(ofile)
    output.write("{} {}\n".format(mu, res.to_str(name)))
output.close()
```

Tutorial: Number density of the hardcore Bosons on a square lattice (6)





Other output files

1. Structure factor (Keyword: sfoutfile in std.tom)

$$S^{zz}(\vec{k},\tau) \equiv \left\langle M^z(\vec{k},\tau)M^z(-\vec{k},0)\right\rangle - \left\langle M^z(\vec{k},\tau)\right\rangle\left\langle M^z(-\vec{k},0)\right\rangle$$

Wave vector k and imaginary time t are specified by the name `C<k>t<t>` as the following:

```
R C0t0 = 1.32500000e-03 1.40929454e-04
R C0t1 = 1.32500000e-03 1.40929454e-04
R C1t0 = 7.35281032e-02 3.18028565e-04
```

2. Real temperature Green's function (Keyword: cfoutfile in std.toml)

$$G(\vec{r}_{ij},\tau) \equiv \left\langle M_i^+(\tau)M_j^-\right\rangle$$

3. Momentum space temperature Green's function
   (Keyword: ckoutfile in std.toml)

$$G(\vec{k},\tau) \equiv \left\langle M^+(\vec{k},\tau)M^-(-\vec{k},0)\right\rangle$$