DSQSS Documentation

Release 2.0.0

DSQSS developers

May 29, 2019

Contents

CHAPTER 1

About DSQSS

1.1 Overview

DSQSS is a program package for solving quantum many-body problems defined on lattices. It is based on the quantum Monte Carlo method in Feynman's path integral representation. It covers a broad range of problems written by flexible input files that define arbitrary unit cells in arbitrary dimensions, and arbitrary matrix elements of the interactions among arbitrary number of degrees of freedom.

For example, you can perform finite temperature calculation of XXZ spin model by specifying parameters such as dimension, size of lattice, anisotropic coupling constants, length of spin, strength of magnetic field, and temperature. You can calculate Bose-Hubbard model as well as quantum spin model. PMWA (Parallel Multi Worm Algorithm) suits for large-scale non-trivial parallel calculation by domain parallelization.

DSQSS consists of the following subpackages.

- serial (dla)
  1. Input files generator: `dla_pre`
  2. Hamiltonian file generator: `dla_hamgen`
  3. Lattice file generator: `dla_latgen`
  4. Algorithm file generator: `dla_alg`
  5. QMC engines (directed loop algorithm): `dla`
- non-trivial parallel (pmwa)
  1. Input files generator: `pmwa_pre`
  2. Lattice file generator: `lattgene_P`
  3. QMC engines (multiworm): `pmwa_H` (XXZ spin model), `pmwa_B` (Hard core Boson)

1.2 Developers

2018/10/19

- Yasuyuki Kato (Univ. of Tokyo)
- Naoki Kawashima (ISSP)
- Kota Sakakura (NEC)
- Takafumi Suzuki (Hyogo Univ.)
- Kenji Harada (Kyoto Univ.)
- Akiko Masaki (Hitachi Research Laboratory)
- Yuichi Motoyama (ISSP)
- Kazuyoshi Yoshimi (ISSP)

1.3 Collaborators

2018/10/19

- Tsuyoshi Okubo (Univ. of Tokyo)
- Takeo Kato (ISSP)

1.4 License

- GNU General Public License (GPL)

The users are kindly requested to acknowledge the usage of this software in their publication, if any, based on the software, and let the developers know its reference information.

Acknowledgment Sample

Numerical results in the present paper were obtained by the quantum Monte Carlo program DSQSS(https://github.com/qmc/dsqss/wiki). This package is distributed under GNU General Public License version 3 (GPL v3) or later.

1.5 Acknowledgment

Development of this software was and is being supported by K-computer project, post-K computer projects and "Project for advancement of software usability in materials science" by ISSP.

1.6 Contact

Write topics to GitHub's issues or send the e-mail to the following mailing list *dsqss-dev@issp.u-tokyo.ac.jp*

CHAPTER 2

How to install

2.1 Requirements

- (Optional) MPI (essential for PMWA)
- python 2.7 or 3.4+
  - numpy
  - scipy
  - toml
  - pip (essential for `make install`)

2.2 Download

- Download zip file

  The latest version of DSQSS can be obtained from https://github.com/issp-center-dev/dsqss/releases

- Use git

  Type the following command:

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

2.3 Directory structure

```
|-- CMakeLists.txt
|-- LICENSE
|-- README.md
|-- config/
```

(continues on next page)

(continued from previous page)

```
|-- doc/
|-- sample/
|   |-- dla/
|   `-- pmwa/
|-- src/
|   |-- common/
|   |-- dla/
|   |-- pmwa/
|   `-- third-party/
|-- test/
|   |-- dla/
|   |-- pmwa/
|   `-- tool/
`-- tool/
    |-- cmake/
    |-- dsqss/
    `-- setup.py
```

2.4 Install

The installation of DSQSS can be done by the following procedures. In the following, we assume that you are in the root directory of dsqss.

```
$ mkdir dsqss.build && cd dsqss.build
$ cmake ../ -DCMAKE_INSTALL_PREFIX=/path/to/install/to
$ make
```

Replace `/path/to/install/to` with the directory path where you want to install dsqss. It is noted that the default install directory is set as `/usr/local/bin` .

Note: By default, CMake usually sets `/usr/bin/c++` as a C++ compiler for building DSQSS. If you want to use another C++ compiler like `icpc` (Intel compiler,) you should tell it to CMake by using `-DCMAKE_CXX_COMPILER` option:

```
$ cmake ../ -DCMAKE_CXX_COMPILER=icpc
```

For Intel compiler, DSQSS offers another option to set compiler options besides compiler itself as following:

```
$ cmake ../ -DCONFIG=intel
```

For details, see https://github.com/issp-center-dev/HPhi/wiki/FAQ .

Each binary files for dsqss will be made in `src` and `tool` directories. To check whether the binary files are correctly made or not, please type the following command:

```
$ make test
```

After seeing that all tests are passed, type the following command to install files:

```
$ make install
```

This command installs executable files into the `bin` directory, the sample files into the `share/dsqss/VERSION/samples` directory, and the python package `dsqss` into the `lib` directory under the install path before you set. One

configuration file for setting environment variables to perform DSQSS commands will also be installed as `share/dsqss/dsqssvar-VERSION.sh` . Before invoke DSQSS commands, please load this file by `source` command as

```
$ source share/dsqss/dsqssvar-VERSION.sh
```

In the remaining, it is assumed that DSQSS is installed and this configuration file is loaded.

CHAPTER 3

Tutorial of DSQSS/DLA

3.1 What's DSQSS/DLA?

DSQSS/DLA is a solver package based on the world line Monte Carlo method with the directed loop algorithm. This can simulate many quantities, e.g. magnetization and susceptibility, of any model on any lattice if free from negative sign problem. This comes with some utility tools which generate input files for some models and lattices; Heisenberg model, Bose-Hubbard model, hypercubic lattice, and triangular lattice.

3.2 Energy calculation of the antiferromagnetic Heisenberg dimer by DSQSS/DLA

This tutorial gives how to use DSQSS/DLA through a calculation the energy of the $S = 1/2$ antiferromagnetic Heisenberg dimer $\mathcal{H} = -J\vec{S}_1 \cdot \vec{S}_2$.

DSQSS/DLA calculation has the following three parts:

1. Prepare the input files
2. Perform QMC calculation
3. Analyze the result

3.2.1 Prepare the input files

DSQSS/DLA requires the following input files:

1. Parameter file
2. lattice file
3. algorithm file

`dla_pre` is a utility tool to generate these files from one textfile such as the following (sample/dla/01_spindimer/std.toml)

```
[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

[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
```

Give this file to `dla_pre` as

```
$ dla_pre std.toml
```

This generates the following four files: a parameter file `param.in`, a lattice file `lattice.xml`, an algorithm file `algorithm.xml`.

3.2.2 Perform QMC calculation

Once input files are prepared, you can perform a quantum Monte Carlo calculation based on the directed loop algorithm (dla) by DSQSS/DLA.

```
$ dla param.in
```

You can perform random number parallelization by using MPI.[1]

```
$ mpiexec -np 4 dla param.in
```

By the above command, the total number of Monte Carlo samples times by four (equals to the number of process) and the obtained statistical error is expected to reduce to half (equals to the inverse square root of the number of process).

3.2.3 Analyze the result

The result of the calculation is written into a text file `sample.log`. Since the energy per site is recorded with a name `ene`, you can for example draw this by the `grep` command by the following.

```
$ grep ene sample.log
R ene = -3.74380000e-01 5.19493985e-03
```

[1] After finishing DSQSS/DLA, the OpenMPI on macOS may say an error message, `No such file or directory (errno 2)`. You can ignore this error safely. If you're annoyed by it, please put an extra option `--mca shmem posix` to `mpiexec`.

The two figures stand for the expectation value and the statistical error, respectively. The result value is compatible with the exact solution, $-3|J|/8 = -0.375|J|$, within the statistical error.

3.3 Magnetic Susceptibility of antiferromagnetic spin chains

In this tutorial, we will calculate the temperature dependence of the magnetic susceptibility for two kinds of antiferromagnetic spin chains with the local spin length $S = 1/2, 1$.

The following Python script (sample/dla/02_spinchain/exec.py) performs DSQSS/DLA work-flow for each parameter (combination of spin length and temperature) automatically.

```
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()
```

Before executing this script, `source` a configuring file `dsqssvars-VERSION.sh` in order to set environment variables (replace `VERSION` with the version of DSQSS, e.g., *2.0.0*.)

```
$ source $DSQSS_INSTALL_DIR/share/dsqss/dsqssvars-VERSION.sh
$ python exec.py
```

The result of $S = 1/2, 1$ will be written to `xmzu_1.dat` and `xmzu_2.dat`, respectively (Fig. 3.1). The $S = 1/2$ chain is gapless and so the magnetic susceptibility remains finite at absolute zero (note that in the simulation the finite size effect opens energy gap). On the other hand the magnetic susceptibility of the $S = 1$ chain drops to zero at finite temperature due to the spin gap.

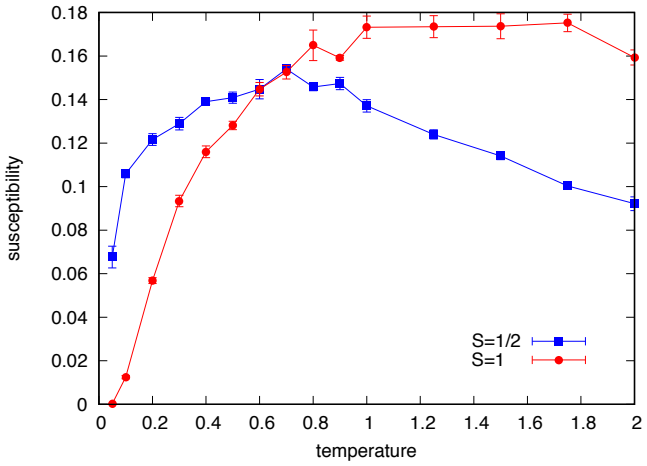

Fig. 3.1: Temperature v.s. susceptibility curves for $S = 1/2$ (blue) and $S = 1$ (red) antiferromagnetic Heisenberg chain.

3.4 Number density of the hardcore Bosons on a square lattice

In this tutorial, we will calculate the chemical potential dependence of the number density of the hardcore Bose-Hubbard model with the nearest neighbor repulsive on a 8×8 square lattice.

The following Python script (sample/dla/03_bosesquare/exec.py) performs DSQSS/DLA work-flow for each parameter (chemical potential) automatically.

```
import subprocess

from dsqss.parameter 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]
```

(continues on next page)

(continued from previous page)

```
    subprocess.call(cmd)
    res = Results(ofile)
    output.write('{} {}\n'.format(mu, res.to_str(name)))
output.close()
```

Before executing this script, `source` a configuring file `dsqssvars-VERSION.sh` in order to set environment variables (replace `VERSION` with the version of DSQSS, e.g., *2.0.0*.)

```
$ source $DSQSS_INSTALL_DIR/share/dsqss/dsqssvars-VERSION.sh
$ python exec.py
```

The result is written to `amzu.dat` (Fig. 3.2). You can see a density plateau around $\mu = 6$. In this region, a checker board solid phase due to repulsive interaction appears.

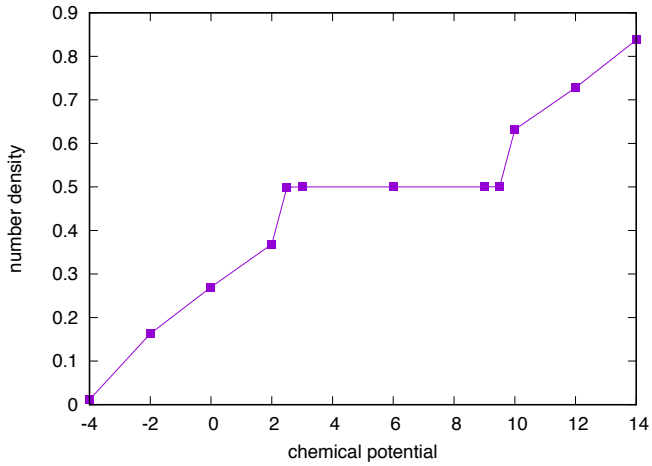

Fig. 3.2: Chemical potential dependence of number density of repulsive hardcore bosons.

CHAPTER 4

User's manual of DSQSS/DLA

4.1 Simple mode of DSQSS/DLA

A simple mode of DSQSS/DLA is the simplest workflow for DSQSS/DLA. In this mode, users can simulate of a predefined model on a predefined lattice from one text file. Fig. 4.1 shows a workflow of the simple mode.

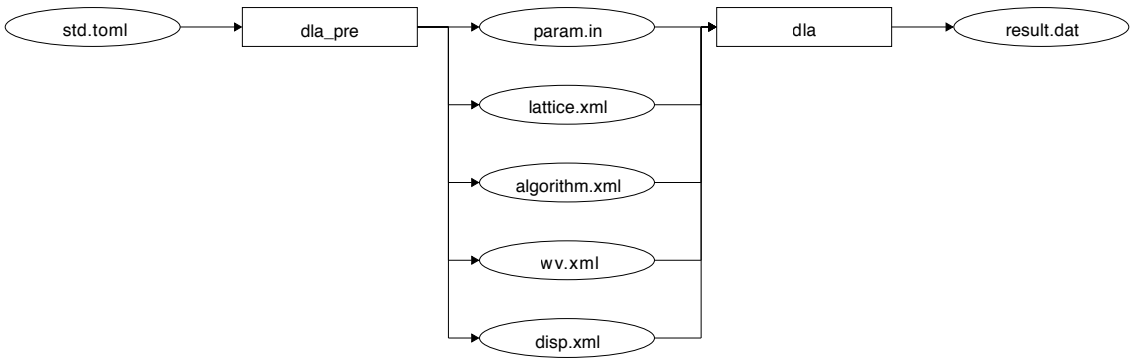

Fig. 4.1: Simple mode of DSQSS/DLA. Ellipses are files and rectangles are tools.

4.1.1 Simple mode file `std.toml`

A simple mode file is a textfile written in TOML format. This is used as an input file of several tools such as `dla_pre`.

4.1.1.1 `parameter`

A table specifying simulation parameters such as the inverse temperature. This table is used in `dla_pre` and `dla_pgen`.

Keys in the `parameter` table are following:

name	type	default	description
beta	double	–	Inverse temperature.
npre	int	1000	The number of Monte Carlo steps in the pre-calculation phase where the number of creation trials of a pair of worms in one Monte Carlo sweep is defined.
ntherm	int	1000	The number of Monte Carlo sweeps to thermalize the system.
ndecor	int	1000	The number of Monte Carlo sweeps to reduce autocorrelation time between two preceding sets.
nmcs	int	1000	The number of Monte Carlo sweeps to calculate mean values of observables.
nset	int	10	The number of Monte Carlo sets.
simulationtime	double	0.0	Simulation time in second.
seed	int	198212240	The seed of the random number generator.
nvermax	int	10000	The maximum number of vertices.
nsegmax	int	10000	The maximum number of world-line segments.
algfile	string	algorithm.xml	The filename of an algorithm file.
latfile	string	lattice.xml	The filename of a lattice file.
wvfile	string	–	A wavevector XML file. If it is an empty string, observables as functions of wavevector will not be calculated.
dispfile	string	–	A relative coordinates XML file. If it is an empty string, observables as functions of relative coordinates will not be calculated.
outfile	string	sample.log	The name of the main result file.
sfoutfile	string	sf.dat	The name of the structure factor result file.
cfoutfile	string	cf.dat	The name of the real space temperature Green's function output file.
ckoutfile	string	ck.dat	The name of the momentum space temperature Green's function output file.

- About simulationtime
  - When simulationtime > 0.0
  - DSQSS/DLA loads the checkpoint file and resumes the simulation if the checkpoint file exists. - If not, DSQSS/DLA starts a new simulation.
  - After the time specified by "simulationtime" (in seconds) has elapsed, DSQSS/DLA saves the state of the simulation into the checkpoint file and halts the simulation.
  - The name of the checkpoint file is that of the main result file with a suffix ".cjob".
  - When simulationtime <= 0.0
    - The checkpoint file is ignored. DSQSS/DLA never saves nor loads it.

4.1.1.2 `lattice`

A table specifying information of lattice. This is used in `dla_pre` and `dla_latgen`.

Keys in the `lattice` table are following:

name	type	default	description
lattice	string	–	The type of lattice.
dim	int	–	Dimension.
L	list(int) or int	–	The size of the lattice. Specified by an integer or a list of integers. If the number of elements is less than the dimension, missing elements are filled by the last element of the given list.
bc	list(bool) or bool	true	The boundary condition of the lattice. Specified by a boolean or a list of booleans. `true` means the periodic boundary condition, `false` means the open boundary condition.

Available lattices are following.

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

triangular A two dimensional triangular lattice.

honeycomb A two dimensional honeycomb lattice.

kagome A two dimensional kagome lattice.

4.1.1.3 `hamiltonian`

A table specifying information of Hamiltonian. This table is used in `dla_pre` and `dla_hamgen`.

Keys in the `hamiltonian` table are following:

name	type	default	description
model	string	–	The type of the model. 'spin' means XXZ spin model and 'boson' means Bose-Hubbard model.
M	int	1	For XXZ model twice as the length of the local spin, $M = 2S$, and for Bose-Hubbard model the cutoff of the number of particles on a site.

XXZ model

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

has the following parameters:

name	type	default	description
Jz	list(float) or float	0.0	The exchange interaction. Positive for ferromagnetic and negative for antiferromagnetic.
Jxy	list(float) or float	0.0	The exchange interaction. Positive for ferromagnetic and negative for antiferromagnetic.
D	list(float) or float	0.0	The onsite uniaxial anisotropy.
h	list(float) or float	0.0	The magnetic field.

Bose-Hubbard model

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

has the following parameters:

name	type	default	description
t	list(float) or float	0.0	The hopping parameter.
V	list(float) or float	0.0	The offsite interaction. Positive for repulsion and negative for attraction.
U	list(float) or float	0.0	The onsite interaction. Positive for repulsion and negative for attraction.
mu	list(float) or float	0.0	The chemical potential.

4.1.1.4 `kpoints`

A table specifying information of wavevectors. This table is used in `dla_pre` and `dla_wvgen`.

Keys in the `kpoints` table are following:

name	type	default	description
ksteps	list(int) or int	0	Increments of wavenumber. If 0, half of lattice size instead of 0 is set.

4.1.1.5 `algorithm`

A table specifying algorithm for calculating scattering probability of wormheads. This table is used in `dla_pre`.

Keys in the `algorithm` table are following:

name	type	default	description
kernel	string	'suwa todo'	Algorithm for calculating the scattering probability of wormheads.

Available `kernel` s are following:

`suwa todo` Rejection minimized algorithm without detailed balance condition (irreversible) proposed by Suwa and Todo. (H. Suwa and S. Todo, PRL 105, 120603 (2010))

`reversible suwa todo` Rejection minimized algorithm with detailed balance condition (reversible) proposed by Suwa and Todo. (arXiv:1106.3562)

`heat bath` Heat bath method (Gibbs sampler).

`metropolice` Metropolice-Hasting algorithm.

4.2 Standard input files for DSQSS/DLA

In the standard model of DSQSS/DLA, users can define their own models and lattices. Of course, they can be combined with predefined ones. Fig. 4.2 shows a workflow of the standard mode.

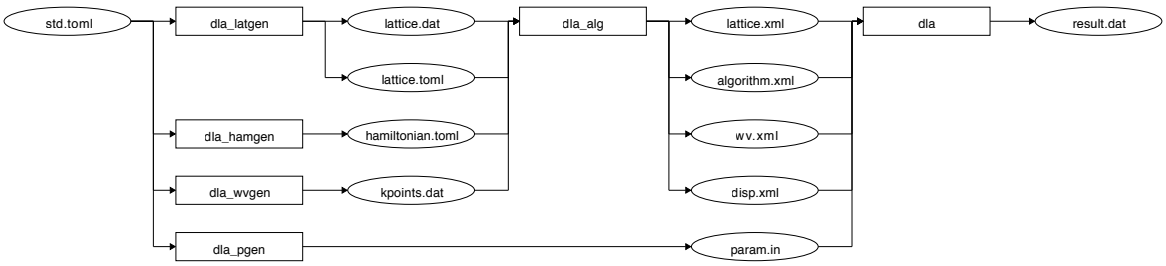

Fig. 4.2: Standard mode of DSQSS/DLA. Ellipses are files and rectangles are tools.

4.2.1 List of files

lattice.dat	lattice datafile.
lattice.toml	lattice TOML file.
hamiltonian.toml	Hamiltonian TOML file.
kpoints.dat	Wavevector file.

4.2.2 Lattice datafile

A lattice datafile is a textfile describing information of space (lattice), such as the number of sites and links connecting some sites. This file is used as an input file of `dla_alg`.

From # mark to the end of the line is a comment and ignored. A blank line is also ignored. A "list" is written as space separated elements in one line like `2 2`.

The lattice datafile includes the five sections, `name`, `lattice`, `directions`, `sites`, `interactions`.

`name` A string denoting the name of lattice. This is just a comment or a marker, and will not be used.

`lattice` The section specifying general information of lattice.

- **First line**
  - An integer denoting dimension of the lattice.
- **Second line**
  - A list of integers denoting the size of the lattice.
- **Third line**
  - A list of integers denoting the boundary condition of the lattice. `1` means the periodic boundary condition and `0` means the open one.
- **Rest**
  - A list of figures denoting the primitive translation vector $\vec{e}^i$. The first integer means the index of the vector i and the rest floating point numbers mean coordinates of the vector in Cartesian, e^i_d.

`directions` The section specifying directions of bonds (two-body interactions.)

- **First line**
  - An integer denoting the number of the directions.
- **Rest**

  - A list of figures denoting the a direction. The first integer means the index of the direction and the rest floating point numbers mean coordinates. The coordinates is represented by using the basis specified in `lattice`.

`sites` The section specifying sites.

- **First line**
  - An integer denoting the number of the sites.
- **Rest**
  - A list of figures denoting a site. The first two integers mean the index and the type of the site, respectively. The rest floating point numbers mean the coordinates in the basis specified in `lattice`.

`interactions` The section specifying interactions.

- **First line**
  - An integer denoting the number of the interactions.
- **Rest**
  - A list of $N =$ involved sites $+ 5$ integers. The details are following.

column	description
1	The index of the interaction.
2	The type of the interaction.
3	The number of involved sites.
4 ... (N-2)	The indices of the involved sites.
N-1	If the bond crosses the boundary, this is 1. Otherwise, 0.
N	The index of the direction.

The following is an example describing 4×4 square lattice:

```
name
2 dimensional hypercubic lattice

lattice
2 # dim
4 4 # size
1 1 # 0:open boundary, 1:periodic boundary
0 1.0 0.0 # latvec_0
1 0.0 1.0 # latvec_1

directions
2 # ndirections
# id, coords...
0 1.0 0.0
1 0.0 1.0

sites
16 # nsites
# id, type, coord...
0 0 0 0
1 0 1 0
2 0 2 0
3 0 3 0
4 0 0 1
```

(continues on next page)

(continued from previous page)

```
5 0 1 1
6 0 2 1
7 0 3 1
8 0 0 2
9 0 1 2
10 0 2 2
11 0 3 2
12 0 0 3
13 0 1 3
14 0 2 3
15 0 3 3

interactions
32 # nints
# id, type, nbody, sites..., edge_flag, direction
0 0 2 0 1 0 0
1 0 2 0 4 0 1
2 0 2 1 2 0 0
3 0 2 1 5 0 1
4 0 2 2 3 0 0
5 0 2 2 6 0 1
6 0 2 3 0 1 0
7 0 2 3 7 0 1
8 0 2 4 5 0 0
9 0 2 4 8 0 1
10 0 2 5 6 0 0
11 0 2 5 9 0 1
12 0 2 6 7 0 0
13 0 2 6 10 0 1
14 0 2 7 4 1 0
15 0 2 7 11 0 1
16 0 2 8 9 0 0
17 0 2 8 12 0 1
18 0 2 9 10 0 0
19 0 2 9 13 0 1
20 0 2 10 11 0 0
21 0 2 10 14 0 1
22 0 2 11 8 1 0
23 0 2 11 15 0 1
24 0 2 12 13 0 0
25 0 2 12 0 1 1
26 0 2 13 14 0 0
27 0 2 13 1 1 1
28 0 2 14 15 0 0
29 0 2 14 2 1 1
30 0 2 15 12 1 0
31 0 2 15 3 1 1
```

4.2.3 Lattice TOML file `lattice.toml`

A lattice TOML file is a TOML formatted file defining a lattice by using an unitcell and primitive translation vectors. This is used as an input file of `dla_alg`.

This has two tables, `parameter` and `unitcell`.

`parameter`

A table denoting general information of the lattice.

name A string denoting the name of lattice.

dim

An integer denoting the dimension of lattice.

L An array of integers denoting the size of lattice.

bc An array of booleans denoting the boundary condition of lattice. `true` means the periodic boundary condition and `false` means the open one.

basis An two-dimensional array (array of array) of floating point numbers denoting the primitive translation vectors.

unitcell A table describing a unitcell.

sites An array of tables denoting sites in one unitcell. One table corresponding one site.

siteid An integer denoting the local index of the site in one unitcell.

type An integer denoting the type of the site.

coord An array of floating point numbers denoting the coordinate of site in one unitcell.

bonds An array of tables denoting bonds in one unitcell. One table corresponding one bond.

bondid An integer denoting the local index of the bond in one unitcell.

type An integer denoting the type of the bond.

source A table denoting an end (source site) of the bond.

siteid An integer denoting the local index of the source site.

target A table denoting the other end (target site) of the bond.

siteid An integer denoting the local index of the target site.

offset An array of integers denoting the relative coordinate of the unitcell where the target site belongs from the unitcell where the source site belongs.

The following is an example describing two dimensional square lattice.

```
[parameter]
name = "square lattice"
dim = 2
L = [4,4]
bc = [true, true]
basis = [[1,0], [0,1]]

[unitcell]

[[unitcell.sites]]
siteid = 0
type = 0
coord = [0,0]

[[unitcell.bonds]]
bondid = 0
type = 0
source = { siteid = 0 }
target = { siteid = 0, offset = [1,0] }
[[unitcell.bonds]]
```

(continues on next page)

(continued from previous page)

```
bondid = 1
type = 0
source = { siteid = 0 }
target = { siteid = 0, offset = [0,1] }
```

4.2.4 Hamiltonian TOML file

A Hamiltonian TOML file is a TOML formatted file describing the local Hamiltonian. This file is used as an input file of `dla_alg`. A utility tool `dla_hamgen` generates Hamiltonian file for common models, XXZ spin model and Bose-Hubbard model.

`name` A string denoting the name of the Hamiltonian.

`sites` An array of tables denoting the site Hamiltonians. One table corresponds to one type of site hamiltonian.

- **`type`** An integer denoting the type of site Hamiltonian.
- **`N`** An integer denoting the number of states of the local degree of freedom. For example, for $S = 1/2$ spin this is 2.
- **`values`** An array of floating point numbers denoting the elements of local basis such as S^z.
- **`elements`** An array of tables denoting the elements of the Hamiltonian matrix. One table corresponds to one element.
  - **`istate`** An integer denoting the index of the initial state (before the Hamiltonian acts on.)
  - **`fstate`** An integer denoting the index of the final state (after the Hamiltonian acts on.)
  - **`value`** A floating point number denoting the matrix element.
- **`sources`** An array of tables denoting the elements of the source Hamiltonian matrix, which is introduced to create and annihilate worm heads. One table corresponds to one element.
  - **`istate`** An integer denoting the index of the initial state (before the Hamiltonian acts on.)
  - **`fstate`** An integer denoting the index of the final state (after the Hamiltonian acts on.)
  - **`value`** A floating point number denoting the matrix element.

`interactions` An array of tables denoting the many-body interaction Hamiltonians. One table corresponds to one type of interaction.

- **`type`** An integer denoting the type of interaction.
- **`nbody`** An integer denoting the number of involved sites.
- **`N`** An array of integers denoting the number of local states on the involved sites.
- **`elements`** An array of tables denoting the elements of the Hamiltonian matrix. One table corresponds to one element.
  - **`istate`** An array of integers denoting the index of the initial state (before the Hamiltonian acts on.)
  - **`fstate`** An array of integers denoting the index of the final state (after the Hamiltonian acts on.)
  - **`value`** A floating point number denoting the matrix element.

The following is an example describing $S = 1/2$ antiferromagnetic Heisenberg spin model

```
name = "S=1/2 XXZ model"
[[sites]]
id = 0
N = 2
values = [-0.5, 0.5]
[[sites.elements]]
istate = 0
fstate = 0
value = 0.5

[[sites.elements]]
istate = 1
fstate = 1
value = -0.5

[[sites.sources]]
istate = 0
fstate = 1
value = 0.5

[[sites.sources]]
istate = 1
fstate = 0
value = 0.5


[[interactions]]
id = 0
nbody = 2
N = [ 2, 2]
[[interactions.elements]]
istate = [ 0, 0]
fstate = [ 0, 0]
value = 0.25

[[interactions.elements]]
istate = [ 0, 1]
fstate = [ 0, 1]
value = -0.25

[[interactions.elements]]
istate = [ 0, 1]
fstate = [ 1, 0]
value = 0.5

[[interactions.elements]]
istate = [ 1, 0]
fstate = [ 1, 0]
value = -0.25

[[interactions.elements]]
istate = [ 1, 0]
fstate = [ 0, 1]
value = 0.5

[[interactions.elements]]
istate = [ 1, 1]
```

(continues on next page)

(continued from previous page)

```
fstate = [ 1, 1]
value = 0.25
```

4.2.5 Wavevector datafile

A wavevector datafile is a textfile describing wavevectors

$$\vec{k}^{(i)} = \sum_{d=1}^{D} k_d^{(i)} \vec{g}_d,$$

where $\vec{g}$ is the set of the reciprocal vectors.

From # mark to the end of the line is a comment and ignored. A blank line is also ignored. A "list" is written as space separated elements in one line like `2 2`.

The wavevector datafile includes the two sections, `dim` and `kpoints`.

`dim` An integer denoting the dimension of the lattice.

`kpoints` A section describing wavevectors.

- **First line**
  - An integer denoting the number of wavevectors.
- **Rest**
  - A list of integers denoting the wavevector. The first integer means the index of the wavevector. The rest integers means the coordinates of the wavevector, k_d.

Wavevectors are represented by using the reciprocal vectors $\vec{g}$. When the coordinate of a lattice site is $\vec{r} = \sum r_d \vec{e}_d$ and the wavevector is $\vec{k} = \sum k_d \vec{g}_d$, the innerproduct of them is $\vec{r} \cdot \vec{k} = \sum_d 2\pi r_d k_d / L_d$, where L_d is the length of the lattice along d th dimension.

The inner product between the coordinate $\vec{r} = \sum r_d \vec{e}_d$ and the wavevector $\vec{k} = \sum k_d \vec{g}_d$ is $\vec{r} \cdot \vec{k} = \sum_d 2\pi r_d k_d / L_d$, where L_d is the size of lattice along d th dimension.

The following is an example of two dimensional case.

```
dim
2

kpoints
3
0 0 0
1 2 0
2 4 0
```

4.3 Input files for DSQSS/DLA

4.3.1 The list of input files

qmc.inp	Parameter list for the simulation, e.g., the number of Monte Carlo sets.
lattice.xml	Definition of the lattice.
algorithm.xml	Definition of the algorithm (e.g., scattering rate of a worm).
wavevector.xml	Indication of wave vectors for structure factors. (optional)
displacement.xml	Indexing directions between all the sites. (optional)

4.3.2 Parameter file

The parameter file is a plain-text file with the following format,

- One line stands for one parameter by the key-value style, `<name> = <value>`.
- Cases are insensitive except for file names.
- White spaces and blank lines are ignored.
- Parts between "#" symbol and the linebreak are ignored as comments.

The list of parameters are the following,

name	type	default	description
beta	double	–	Inverse temperature.
npre	int	1000	The number of Monte Carlo steps in the pre-calculation phase where the number of creation trials of a pair of worms in one Monte Carlo sweep is defined.
ntherm	int	1000	The number of Monte Carlo sweeps to thermalize the system.
ndecor	int	1000	The number of Monte Carlo sweeps to reduce autocorrelation time between two preceding sets.
nmcs	int	1000	The number of Monte Carlo sweeps to calculate mean values of observables.
nset	int	10	The number of Monte Carlo sets.
simulationtime	double	0.0	Simulation time in second.
seed	int	198212240	The seed of the random number generator.
nvermax	int	10000	The maximum number of vertices.
nsegmax	int	10000	The maximum number of world-line segments.
algfile	string	algorithm.xml	The filename of an algorithm file.
latfile	string	lattice.xml	The filename of a lattice file.
ntau	int	10	The number of the discretization of the imaginary time for calculating some observables as functions of imaginary time.
wvfile	string	–	A wavevector XML file. If it is an empty string, observables as functions of wavevector will not be calculated.
dispfile	string	–	A relative coordinates XML file. If it is an empty string, observables as functions of relative coordinates will not be calculated.
outfile	string	sample.log	The name of the main result file.
sfoutfile	string	sf.dat	The name of the structure factor result file.
cfoutfile	string	cf.dat	The name of the real space temperature Green's function output file.
ckoutfile	string	ck.dat	The name of the momentum space temperature Green's function output file.

- About simulationtime
  - When simulationtime > 0.0
    - **DSQSS/DLA loads the checkpoint file and resumes the simulation if the checkpoint file exists.**
      - If not, DSQSS/DLA starts a new simulation.
    - After the time specified by "simulationtime" (in seconds) has elapsed, DSQSS/DLA saves the state of the simulation into the checkpoint file and halts the simulation.
    - The name of the checkpoint file is that of the main result file with a suffix ".cjob".
  - When simulationtime <= 0.0
    - The checkpoint file is ignored. DSQSS/DLA never saves nor loads it.

4.3.3 Lattice XML file `lattice.xml`

A lattice file is a textfile written in XML format. This defines space to be simulation, for example, the number of sites, the connections between sites, and so on. This file can be very complicated, so DSQSS has a utility tool `dla_alg` to generate lattice XML files from simpler files, lattice data files and lattice TOML files.

The lattice file has a unique element named "Lattice". The other elements belong to "Lattice" as children.

Lattice The root element.

Lattice/Comment (Optional) Comment. DSQSS ignores this element.

Lattice/Dimension The dimension of the lattice.

Lattice/LinearSize The size of the lattice in units of the unitcell. This takes space-separated positive integers as many as specified by "Lattice/Dimension".

```
<LinearSize> 3 4 </LinearSize>
# unitcells are arranged in 3x4.
```

Lattice/NumberOfSites The number of sites.

Lattice/NumberOfInteractions The number of the interactions. When all the interactions are two-body ones, this is nothing but the number of bonds.

Lattice/NumberOfSiteTypes The number of site types.

Lattice/NumberOfInteractionTypes The number of interaction types.

Lattice/NumberOfBondDirections The number of bond directions.

Lattice/NumberOfEdgeInteractions The number of bonds connecting sites over the lattice's boundary.

Lattice/Basis The basic vectors spanning lattice space.

Lattice/S Site information. "Lattice" should includes this element as many as the number specified by "Lattice/NumberOfSites". This takes two nonnegative integers, "index of site" and "site type". The detail of site type is defined in an algorithm file.

```
<S> 3 0 </S>
# the site with index 3 has the site type of 0.
```

Lattice/I Interaction information. "Lattice" should includes this element as many as the number specified by "Lattice/NumberOfInteractions". This takes space-separated integers, "index of the interaction", "interaction type", "the number of sites involved in the interaction", "indices of involved sites". The details of interaction type,

e.g., the strength, are defined in an algorithm file. The order of the indices of sites should be compatible with the order of sites specified in "Algorithm/Vertex/InitialConfiguration" in the algorithm file.

```
<I> 5 1 2 8 12 </I>
# the interaction with index 5 has the interaction type of 1 and connects 2 sites,
↪ 8 and 12.
```

Lattice/Direction The direction of bonds. This takes "index of the direction" and "coordinates of the direction."

4.3.4 Algorithm XML file `algorithm.xml`

An algorithm file is a textfile written in XML format. This defines the details of interactions, for example, the scattering probability of a worm head. This file can be very complicated, so DSQSS has a utility tool `dla_alg` to generate algorithm files from more simple file, the Hamiltonian file introduced later.

The algorithm file has a unique element named "Algorithm". The other elements belong to "Algorithm" as children.

Algorithm The root element. This has children, "General", "Site", "Interaction", and "Vertex".

Algorithm/Comment (Optional) Comment. DSQSS ignores this.

Algorithm/General General parameters such as the number of site types. This has children, "NSType", "NIType", "NVType", "NXMax", and "WDiag".

```
<Algorithm>
  <General>
    <NSType>  1 </NSType>
    <NIType>  1 </NIType>
    <NVType>  2 </NVType>
    <NXMax>   2 </NXMax>
    <WDiag>   0.25 </WDiag>
  </General>
  ...
</Algorithm>
```

Algorithm/General/NSType The number of site types.

Algorithm/General/NIType The number of interaction types.

Algorithm/General/NVType The number of vertex types.

Algorithm/General/NXMax The maximum number of states on a site. For example, $2S + 1$ for a spin system with local spin S.

Algorithm/General/WDiag A coefficient to measure correlation functions from the length of worms.

Algorithm/Site This defines a site type, for example, the weight of worm heads on a site. This has children "SType", "NumberOfStates", "VertexTypeOfSource", and "InitialConfiguration".

```
<Algorithm>
  ...
  <Site>
    <STYPE> 0 </STYPE>
    <NumberOfStates> 2 </NumberOfStates>
    <LocalStates> -0.5 0.5 </LocalStates>
    <VertexTypeOfSource> 0 </VertexTypeOfSource>
    <InitialConfiguration>
       ...
    </InitialConfiguration>
```

(continues on next page)

(continued from previous page)

```
    <InitialConfiguration>
       ...
    </InitialConfiguration>
  </Site>
  ...
</Algorithm>
```

Algorithm/Site/SType The index of site type.

Algorithm/Site/NumberOfStates The number of states of the site.

Algorithm/Site/LocalStates Mapping from indices of local states to values of states. For example, the z components of the spin operator in the usual spin case.

Algorithm/Site/VertexTypeOfSource The index of the vertex to be inserted here.

Algorithm/Site/InitialConfiguration The process of pair creation/annihilation of worm heads. This has children, "State", "NumberOfChannels", and "Channel"

```
<Algorithm>
  ...
  <Site>
    ...
    <InitialConfiguration>
      <State> 0 </State>
      <NumberOfChannels> 2 </NumberOfChannels>
      <Channel> 0 1 0.5 </Channel>
      <Channel> 1 1 0.5 </Channel>
    </InitialConfiguration>
    ...
  </Site>
  ...
</Algorithm>
```

Algorithm/Site/InitialConfiguration/State The state index of the site without worms (before creation or after annihilation).

Algorithm/Site/InitialConfiguration/NumberOfChannels The number of the channels (result of creation/annihilation).

Algorithm/Site/InitialConfiguration/Channel Channels. This takes two integers and one floating number.

- First figure denotes the direction of the worm head (0 for negative and 1 for positive in the imaginary time direction).
- Second figure denotes the state between worms.
- Third figure denotes the probability of this channel.

If the result has no worm heads, let both the first and the second integers be -1.

Algorithm/Interaction This defines an interaction. This has children, "IType", "VType", "NBody", "EBase", "VertexDensity", and "Sign".

```
<Algorithm>
  ...
  <Interaction>
    <IType> 0 </IType>
    <VType> 1 </VType>
    <NBody> 2 </NBody>
```

(continues on next page)

(continued from previous page)

```
    <EBase> 0.125 </EBase>
    <VertexDensity> 0 0 0.25 </VertexDensity>
    <VertexDensity> 1 1 0.25 </VertexDensity>
    <Sign> 0 1 1 0 -1.0 </Sign>
    <Sign> 1 0 0 1 -1.0 </Sign>
  </Interaction>
  ...
</Algorithm>
```

Algorithm/Interaction/IType The index of the interaction.

Algorithm/Interaction/VType The index of the vertex to be inserted.

Algorithm/Interaction/NBody The number of sites involved in this interaction. An onebody interaction such as the Zeeman term has 1 and a twobody interaction such as the exchange coupling has 2. Three or higher body interaction can be treated.

Algorithm/Interaction/EBase The offset of the local energy. This value does not contribute to the simulation, but to the value of energy in the final result.

Algorithm/Interaction/VertexDensity The density of vertex to be inserted. This takes integers as many as "Algorithm/Interaction/NBody" and one preceding floating number. The integers denote the states of sites (the order should be compatible with the order of sites in "I" of the lattice file). The last floating number represents the density.

Algorithm/Interaction/Sign The sign of the local weight, $\mathrm{Sgn}(\langle f| - \mathcal{H}|i\rangle)$. This takes integers as many as $2\times$ "Algorithm/Interaction/NBody" and one preceding floating number. The integers denote the states of sites before and after applying the local Hamiltonian. The last floating number represents the sign. If the sign is equal to 1.0:, this element (`<Sign> ... </Sign>`) can be omitted.

For example, `<Sign> 0 1 1 0 -1.0 </Sign>` means $\langle 10|\,(-\mathcal{H})\,|01\rangle < 0$.

Algorithm/Vertex This defines a vertex. This has children, "VType", "VCategory", "NBody", "NumberOfInitialConfigurations", and "InitialConfiguration". Vertices belongs to a category specified by "Algorithm/Vertex/VCategory".

```
<Algorithm>
  ...
  <Vertex>
    <VTYPE> 0 </VTYPE>
    <VCATEGORY> 1 </VCATEGORY>
    <NBODY> 1 </NBODY>
    <NumberOfInitialConfigurations> 4 </NumberOfInitialConfigurations>
    <InitialConfiguration>
      ...
    </InitialConfiguration>
    ...
    <InitialConfiguration>
      ...
    </InitialConfiguration>
  </Vertex>
  ...
</Algorithm>
```

Algorithm/Vertex/VType The index of the vertex.

Algorithm/Vertex/VCategory

0. Boundary of imaginary time. Users need not define this.

1. Worm tail.
2. Interaction.

Algorithm/Vertex/NBody The number of sites involved.

Algorithm/Vertex/NumberOfInitialConfigurations The number of initial states.

Algorithm/Vertex/InitialConfiguration

This defines scattering results of a worm head for each initial states. "Algorithm/Vertex" should has this elements as many as the number specified by "Algorithm/Vertex/NumberOfInitialConfigurations". This has children, "State", "IncomingDirection", "NewState", "NumberOfChannels", "Channel".

```
<Algorithm>
  ...
  <Vertex>
    ...
    <InitialConfiguration>
      <State>  1 0 0 1 </State>
      <IncomingDirection> 0 </IncomingDirection>
      <NewState> 0 </NewState>
      <NumberOfChannels> 1 </NumberOfChannels>
      <Channel>    3    0       1.0000000000000000 </Channel>
    </InitialConfiguration>
    ...
  </Vertex>
  ...
</Algorithm>
```

This example represents the following scenario;

- Initial states of bottom-left(0), top-left(0), bottom-right(2), and top-right(3) are 1, 0, 0, and 1, respectively.
- A worm head comes from bottom-left(0) and changes the state of this leg to 0.
- The worm head will be scattered to leg(3) and the state of outgoing leg will be changed to 0 with the probability 1.

Algorithm/Vertex/InitialConfiguration/State The initial states of the legs of the vertex. Since the number of the legs is as twice as the number specified by "Algorithm/Vertex/NBody", say m, this takes $2m$ integers. Legs are in the same order as the corresponding sites. For two legs on the same site, the leg with the smaller imaginary time comes first.

Algorithm/Vertex/InitialConfiguration/IncomingDirection The index of the leg from which a worm head comes.

Algorithm/Vertex/InitialConfiguration/NewState The state of the "Algorithm/Vertex/InitialConfiguration/IncomingDirection" leg after a worm head comes.

Algorithm/Vertex/InitialConfiguration/NumberOfChannels The number of scattering channels (final results).

Algorithm/Vertex/InitialConfiguration/Channel A scattering channel. This takes two integers and one floating number.

- First figure denotes the **index** of the leg where the scattered worm head goes out.
- Second figure denotes the **state** of the leg where the scattered worm head goes out after the scattering.
- Last figure denotes the probability of this channel.

For the special case, the pair-annihilation of worm heads, let both the first and the second integer be -1.

4.3.5 Wavevector XML file `wavevector.xml`

A wavevector XML file is a textfile written in a XML-like format. This defines the wavevectors to calculate several observables: staggered magnetization

$$M^z(\vec{k}) \equiv \frac{1}{N} \sum_i e^{-i\vec{k}\vec{r}_i} \langle M_i^z \rangle ,$$

dynamical structure factor

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

and momentum space temperature Green's function

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

This can be generated from a wavevector datafile via `dla_alg`.

A structure factor file has only one element, "WaveVector", and the other elements are children of this.

WaveVector The root element. This has children, "Comment", "NumberOfSites", "NumberOfWaveVectors" and "RK".

WaveVector/Comment (Optional) Comment. DSQSS ignores this.

WaveVector/NumberOfSites The number of lattice sites.

WaveVector/NumberOfWaveVectors The number of Wavevectors $\vec{k}$.

WaveVector/RK The phase factor $z = \exp \vec{r} \cdot \vec{k}$ for a pair of a wave vector and a site. This takes four figures, "$\mathrm{Re}z$", "$\mathrm{Im}z$", "the index of the site", "the index of the wave vector". "StructureFactor" should has this elements as many as the number specified by "StructureFactor/NumberOfElements".

4.3.6 Relative coordinate XML file `displacement.xml`

A relative coordinate XML file is a textfile written in a XML-like format. This defines relative coordinate between two sites, $\vec{r}_{ij}$, to calculate real space temperature Green's function,

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

This file can be generated by using `dla_alg`.

A relative coordinate XML file has only one element, "Displacements", and the other elements belong to this as children.

Displacements The root element. This has children, "Comment", "NumberOfKinds", "NumberOfSites", and "R".

Displacements/Comment (Optional) Comment. DSQSS ignores this.

Displacements/NumberOfSites The number of lattice sites.

Displacements/NumberOfKinds The number of relative coordinates.

Displacements/R This takes three integers, "the index of the relative coordinate", "the index of the site i", and "the index of the site j". "CorrelationFunction" should has this elements as many as the number specified by "CorrelationFunction/NumberOfKinds".

4.4 Input file generators for DSQSS/DLA

DSQSS/DLA takes several input files; lattice XML file, algorithm XML file, and wavevector XML file, and users can simulate any model on any lattice (graph) by defining these files properly. They are, however, too complicated to be made by hand. To help users, DSQSS/DLA offers utility tools for generating these files for widely used lattices and models such as a hypercubic lattice and the Heisenberg model.

4.4.1 Simple mode tool `dla_pre`

`dla_pre` is a utility tool for generating *Input files for DSQSS/DLA* from *Simple mode file std.toml*.

```
$ dla_pre [-p paramfile] <inputfile>
```

The meanings of the parameters are following.

`paramfile` The name of the parameter file to be generated (default: `qmc.inp`.)

`inputfile` The name of the input file. For details of the input file, see *Simple mode file std.toml* .

The names of the XML files such as the lattice XML file are automatically determined from parameters.

4.4.2 Lattice file generator `dla_latgen`

`dla_latgen` is a utility tool for generating *Lattice datafile* or *Lattice TOML file lattice.toml* from *Simple mode file std.toml* .

```
$ dla_latgen [-o datafile] [-t TOML] [-g GNUPLOT] input
```

The meanings of the parameters are following.

`datafile` The name of the generated lattice data file (default: `lattice.dat` .) If empty, `dla_latgen` never generate any lattice data file.

`TOML` The name of the generated lattice TOML file (default: empty.) If empty, `dla_latgen` never generate any lattice TOML file.

`GNUPLOT` The name of the generated lattice Gnuplot file (default: empty.) If empty, `dla_latgen` never generate any lattice Gnuplot file. Users can see the generated lattice by `load` the lattice Gnuplot file in `gnuplot`.

`inputfile` The name of the input file. For details of the input file, see *lattice* .

Examples

```
# 1d chain with 8 sites
[lattice]
lattice = "hypercubic"
dim = 1
L = 8
```

```
# 2d square, 4x4 sites
[lattice]
lattice = "hypercubic"
dim = 2
L = 4
```

```
# two leg ladder, 8x2 sites
[lattice]
lattice = "hypercubic"
dim = 2
L = [8,2]
bc = [true, false]
```

4.4.3 Hamiltonian file generator `dla_hamgen`

`dla_hamgen` is a utility tool for generating *Hamiltonian TOML file* from *Simple mode file std.toml*

```
$ dla_hamgen [-o filename] <inputfile>
```

The meanings of the parameters are following.

`filename` The name of the generated Hamiltonian file (default: `hamiltonian.toml` .)

`inputfile` The name of the input file. For details of the input file, see *hamiltonian* .

Example

```
# S=1/2 AF Heisenberg model
[hamiltonian]
model = "spin"
M = 1
Jz = -1.0
Jxy = -1.0
```

```
# S=1 J1 AF J2 FM XY model under the field
[hamiltonian]
model = "spin"
M = 2
Jxy = [-1.0, 1.0]
h = 1.0
```

```
# hardcore boson
[hamiltonian]
model = "boson"
M = 1
t = 1.0
V = 1.0
```

```
# softcore boson (upto N=2)
[hamiltonian]
model = "boson"
M = 2
t = 1.0
U = 1.0
V = 1.0
mu = 1.0
```

4.4.4 Parameter file generator `dla_pgen`

`dla_pgen` is utility tool for generating *Parameter file* from *Simple mode file std.toml* .

```
$ dla_pgen [-o filename] <inputfile>
```

The meanings of the parameters are following.

filename The name of the generated parameter file (default: `param.in` .)

inputfile The name of the input file. For details of the input file, see *parameter* .

4.4.5 Wavevector file generator `dla_wvgen`

`dla_wvgen` is a utility tool for generating *Wavevector datafile* from *Simple mode file std.toml* .

```
$ dla_wvgen [-o filename] [-s size] <inputfile>
```

The meanings of the parameters are following.

filename The name of the generated wavevector file (default: `kpoints.dat` .)

size Space separated integers denoting the lattice size (e.g., `-s "4 4"` .) If omitted, it will be detected from the `[lattice]` table of the input TOML file.

inputfile The name of the input file. For details of the input file, see *hamiltonian* .

4.4.6 Algorithm file generator `dla_alg`

`dla_alg` is a utility tool for generating *Lattice XML file lattice.xml*, *Algorithm XML file algorithm.xml*, *Wavevector XML file wavevector.xml*, and *Relative coordinate XML file displacement.xml* from *Lattice datafile*, *Lattice TOML file lattice.toml*, *Hamiltonian TOML file*, and *Wavevector datafile* .

```
$ dla_alg [-l LAT] [-h HAM] [-L LATXML] [-A ALGXML]
          [--without_lattice] [--without_algorithm] [-k KPOINT]
          [--wv WV] [--disp DISP] [--distance-only]
          [--kernel KERNEL]
```

The meanings of the parameters are following.

LAT The name of the lattice dat/TOML file (default: `lattice.dat`.) The type whether dat or TOML is automatically detected.

HAM The name of the Hamiltonian TOML file (default: `hamiltonian.toml` .)

LATXML The name of the generated lattice XML file (default: `lattice.xml` .)

ALGXML The name of the generated algorithm XML file (default: `algorithm.xml` .)

without_lattice If set, `dla_alg` never generate a lattice XML file. Even in this case, the lattice dat/TOML file is still required.

without_algorithm If set, `dla_alg` never generate an algorithm XML file.

KPOINT The name of the wavevector file. If omit, `dla_alg` never generate a wavevector XML file.

WV The name of the generated wavevector XML file (default: `wavevector.xml` .)

DISP The name of the generated relative coordinate XML file. If omit, `dla_alg` never generate a relative coordinate XML file.

--distance-only If set, `dla_alg` groups pairs of sites by absolute distance instead of relative coordinate.

KERNEL The name of the algorithm for calculating the scattering probability of a worm head at a vertex (default: `"suwa todo"` .) For details, see *algorithm* .

4.5 Directed loop algorithm solver `dla`

`dla` is a quantum Monte Carlo solver with the directed loop algorithm. This takes an input file as the command line argument.:

```
$ dla param.in
```

DSQSS/DLA implements random number parallelization by using MPI. Since each process performs `NSET` Monte Carlo calculation independently, the total number of MC sets increases N_{process} and thus it is expected that the statistical error also reduces to $1/\sqrt{N_{\text{process}}}$ times.

```
$ mpiexec -np 4 dla param.in
```

4.6 Output of DSQSS/DLA

4.6.1 Format

DSQSS/DLA generates the result of a simulation as a plain-text file. The first character stands for the meaning of the line.

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

`R <name> = <mean> <error>` Results of observables. `<mean>` denotes the expected value and `<error>` denotes the statistical error of `<mean>`.

`I <text> = <value>` Other information.

`C <text>` Comments.

The following one is a result of an antiferromagnetic Heisenberg chain.

```
C This is DSQSS ver.1.2.0

P D       =            1
P L       =            8
P BETA    =      10.0000000000000000
P NSET    =           10
P NMCSE   =         1000
P NMCSD   =         1000
P NMCS    =         1000
P SEED    =    198212240
P NSEGMAX =        10000
P NVERMAX =        10000
P NCYC    =            7
P ALGFILE = algorithm.xml
P LATFILE = lattice.xml
P CFINPFILE  = cf.xml
P SFINPFILE  = sf.xml
P CKINPFILE  = sf.xml
P OUTFILE    = res.dat.000
P CFOUTFILE  = cfout.dat.000
P SFOUTFILE  = sfout.dat.000
P CKOUTFILE  = ckout.dat.000
P SIMULATIONTIME   =     0.000000
R anv = 3.03805000e+00 1.25395375e-02
```

(continues on next page)

(continued from previous page)

```
R ene = -4.55991910e-01 1.20267537e-03
R spe = -1.76672204e-02 4.09064489e-02
R len = 1.20014021e+01 4.78403202e-02
R xmx = 3.00035053e-01 1.19600800e-03
R amzu = -2.00000000e-04 1.08972474e-04
R bmzu = -2.00000000e-04 1.08972474e-04
R smzu = 1.32382500e-03 1.40792745e-04
R xmzu = 1.32382500e-02 1.40792745e-03
R amzs = -9.25000000e-04 4.02247160e-03
R bmzs = -2.03918502e-04 2.22828174e-03
R smzs = 8.72503175e-01 8.93939492e-03
R xmzs = 3.00500011e+00 2.99056535e-02
R time = 9.01378000e-08 1.61529255e-09
I [the maximum number of segments]          = 123
I [the maximum number of vertices]          = 66
I [the maximum number of reg. vertex info.] = 3
```

4.6.1.1 Notations

N_s The number of sites.

$Q(\vec{k})$ The Fourier transformation of an arbitrary operator on a site i, Q_i.

$$Q(\vec{k}) \equiv \frac{1}{\sqrt{N_s}} \sum_i^{N_s} Q_i e^{-i\vec{r}_i \cdot \vec{k}}$$

$Q(\tau)$ An arbitrary operator at imaginary time τ.

$$Q(\tau) \equiv \exp[\tau\mathcal{H}]\, Q(\tau = 0) \exp[-\tau\mathcal{H}]$$

$\tilde{Q}$ The average of an arbitrary operator Q over the imaginary time, $\frac{1}{\beta}\int_0^\beta \mathrm{d}\tau Q(\tau)$

M^z The component of a local degree of freedom along with the quantized axis. For example, z component of the local spin operator S^z for spin systems and the number operator n for the Bose-Hubbard models.

$M^\pm$ The ladder operator. $M^\pm \equiv S^\pm$ for spin systems, and the creation/annihilation operators $M^+ \equiv b^\dagger$, $M^- \equiv b$ for the Bose-Hubbard models.

M^x The off-diagonal order parameter. $M^x \equiv (S^+ + S^-)/2$ for spin systems and $M^x \equiv (b + b^\dagger)$ for the Bose-Hubbard models.

T The temperature.

β The inverse temperature.

h The conjugate field to the operator M^z . The longitudinal magnetic field for spin systems and the chemical potential for the Bose-Hubbard models.

$\langle Q \rangle$ The expectation value of an arbitrary operator Q over the grand canonical ensemble.

4.6.2 Main results

Main results are written in a file with the name specified by `outfile` keyword in the input parameter file.

`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)

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

spe The specific heat

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

len The mean length of worm

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).

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

xmzu The longitudinal susceptibility (uniform).

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

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").

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

4.6.3 Structure factor output

The structure factor is written into a file with the name specified by `sfoutfile` keyword in the input file. The structure factor is defined as the following:

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

Wave vector $\vec{k}$ and imaginary time τ 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
```

where <k> is an index of the wave vector specified by `kindex` (the last element of each `RK` tag) in the wavevector XML file and <t> is an index of the discretized imaginary time.

4.6.4 Real space temperature Green's function output

The real space temperature Green's function is written into a file with the name specified by `cfoutfile` keyword in the input file. The real space temperature Green's function is defined as the following:

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

Displacement $\vec{r}_{ij}$ and imaginary time τ are specified by the name `C<k>t<t>` as the same way of structure factor, where <k> is an index of the displacement specified by `kind` (the first element of each `R` tag) in the relative coordinate XML file, and <t> is an index of the discretized imaginary time.

4.6.5 Momentum space temperature Green's function output

The momentum space temperature Green's function is written into a file with the name specified by `ckoutfile` keyword in the input file. The momentum space temperature Green's function is defined as the following:

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

Wave vector $\vec{r}_{ij}$ and imaginary time τ are specified by the name `C<k>t<t>` as the same way of structure factor, where <k> is an index of the displacement specified by `kind` (the last element of each `RK` tag) in the wavevector XML file, and <t> is an index of the discretized imaginary time.

CHAPTER 5

Tutorial of DSQSS/PMWA

5.1 About DSQSS/PMWA

`DSQSS / PMWA` is a numerical package of the world-line quantum Monte Carlo method with non-local updates algorithm for large-scale parallel computing. PMWA can treat $S = 1/2$ quantum many-body system such as XXZ spin model, Heisenberg model, transverse Ising model model and hard-core boson system with large lattice sizes at finite temperatures. Basically, any physical quantities can be calculated within the statistical error. By default, the energy under a longitudinal worm source field, longitudinal magnetization, transverse magnetization, winding number, and imaginary time $S^z S^z$ spin correlation function are calculated.

5.2 Usage

PMWA can be used by the following procedures:

1. Make an input file for a lattice (use `lattgene_P`).
2. Make an input file.
3. Execute PMWA (boson system:`pmwa_B` , spin system:`pmwa_H`)
4. Generate output files.

There is also a simple tool `pmwa_pre` for executing 1 and 2 at once. In this section, the usage and parameters are explained for each procedure.

5.2.1 Make an input file for a lattice

In PMWA, as a simple tool `lattgene_P` is prepared for generating a lattice definition file `lattice.xml` for a standard model. In this section, the usage of `lattgene_P` is explained.

`lattgene_P` generates a lattice definition file for a cubic lattice. The parameters to be specified are as follows.

Parameter	type	Remarks
D	int	The dimensions
L	int	The size of the lattice (specified for each dimension consecutively)
B	double	The inverse temperature
NLdiv	int	The number of divisions of L (divide NLdiv for each dimension)
NBdiv	int	The number of divisions of L
NFIELD	int	The type of magnetic field (basically set to 0)

Examples of use are described below.

1. One-dimensional 8-sites chain, inverse temperature 10.0, definition of lattice file when division number is 1:

   ```
   $ lattgene_P 1 8 10.0 1 1 0
   ```

2. Two dimensional 4*4 sites lattice, reverse temperature 10.0, definition of lattice file when division number is 1:

   ```
   $ lattgene_P 2 4 4 10.0 1 1 0
   ```

5.2.2 Make an input file for DSQSS/PMWA

To execute PMWA, a text format input file is required. There are two kinds of parameters in the input file: parameters specifying calculation conditions and parameters specifying a Hamiltonian.

An example of the input file is shown below:

```
RUNTYPE = 0
NSET   = 10
NMCS   = 10000
NPRE   = 10000
NTHERM = 10000
NDECOR = 10000
SEED  = 31415
NC    = 0
NVERMAX = 10000000
NWORMAX = 1000
latfile   = lattice.xml
outfile   = sample.log
CB      = 2
G       = 0.3
U       = 0
V       = 3
t       = 1
MU      = 2
NMAX    = 1
```

The meaning of each parameter is as follows.

- Parameter of calculation conditions

Parameter	type	Remarks
RUNTYPE	int	Calculation mode (0: normal calculation, 1: restart calculation)
CB	int	Initial arrangement (0: Vacuum, 1: Checker-Board, 2: Random)
NSET	int	The repetition number of Monte Carlo calculation
NMCS	int	The number of Monte Carlo sweeps used for physical quantity calculation
NPRE	int	The number of Monte Carlo steps used for precalculation to determine the number of trials to generate worms per sweep
NMCSE	int	The number of Monte Carlo sweeps used for initial relaxation
NMCSD	int	The number of Monte Carlo sweeps between measurements
SEED	int	If it is 0 or more, seed is actually used, if negative, seed is given with random number.
NVERMAX	int	The maximum number of vertexes (default is 10^8). There is no upper limit when -1.
NWORMAX	int	The maximum number of worms (default is 10^3). There is no upper limit when -1.
SFINPFILE	str	If this file name is given, calculate structure factors specified in this file.
SFOUTFILE	str	If SFINPFILE is given, output structure factors to the specified file (default is sf.out).
Step_x	int	Give the spatial width when computing the correlation function (default: 1).
Step_k	int	Give the width of wavenumber when computing correlation function of wavenumber representation (default: 1).
CFOUTFILE	str	When it is input, it outputs the correlation function to the specified file (default is cf.out).

Here, NVERMAX and NWORMAX are automatically resized and determined (implemented in ver. 1.2.0).

- Parameter of interactions

PMWA can treat the models for hard-core boson and $S = 1/2$ XXZ system. The Hamiltonian for hard-core boson system is given by

$$\mathcal{H} = -t_b \sum_{\langle i,j \rangle} b_i^\dagger b_j + U_{BB} \sum_i n_i(n_i - 1) + V_{B1} \sum_{\langle i,j \rangle} n_i n_j + \mu \sum_i n_i - \Gamma \sum_i (b_i^\dagger + b_i),$$

where $\langle i, j \rangle$ indicates a nearest-neighbor pair. For $S = 1/2$ XXZ model, the hamiltonian is given by

$$\mathcal{H}^{XXZ} = -J_{xy} \sum_{\langle i,j \rangle} (S_i^x S_j^x + S_i^y S_j^y) - J_z \sum_{\langle i,j \rangle} S_i^z S_j^z - H \sum_i S_i^z - \Gamma \sum_i S_i^x.$$

The parameters specified in the input file and the parameters in the above expression correspond as follows.

Parameter	Boson	Spin
t	t_b	J_{xy}
U	U_{BB}	-
V	V_{B1}	J_z
MU	μ	H
G	Γ	$\Gamma/2$

Each parameter is specified as double type.

5.2.3 Run DSQSS/PMWA

After generating the input file, PMWA runs by typing the following command (the name of the input file is assumed as `param.in`). Execution file is different depending on the system, i.e. the spin system or the hard core boson system.

1. Spin system

   ```
   $pmwa_H param.in
   ```

2. Hard core boson system

   ```
   $pmwa_B param.in
   ```

After calculation, one result file and two temporary files for restart mode (evout_sample.log, RND_evout_sample.log) are outputted.

5.2.4 Output files

Here, the parameters specific to PMWA for one result file and two temporary files for restart mode are described.

- Result file

Type	Name	Description
P	L	The size of the lattice.
P	DOML	The size of the domain divided by parallelization
P	DOMBETA	The size of the inverse temperature divided by parallelization
P	NDIVL	The number of lattice divisions
P	NTEST	Number of samples to be tested (for details, see the description of Monte Carlo calculation)
R	nver	The number of kinks and worms
R	nkin	The number of kinks
R	wndx	The expected value of square of winding number in x direction
R	wndy	The expected value of square of winding number in y direction
R	wndz	The expected value of square of winding number in z direction
R	wnd2	The total number of squares of winding number(wndx+wndy+wndz)
R	bmxu	The expected value of S_x (uniform τ integral)
R	bmpu	The expected value of S_+ (uniform τ integral)
R	bmmu	The expected value of S_- (uniform τ integral)
R	comp	The compressibility
R	lxmx	The local worm number fluctuation at each site
I	the maximum number of vertices	The maximum number of vertices
I	the maximum number of worms	The maximum number of worms

Here, the type is the letter given to the beginning of each line of output. P, R, I indicate Parameter, Result, Information respectively.

- Output files for restart mode

PMWA implements the restart function, and if there are two files below, a restart is automatically performed. Below, a brief description of the output contents of each file is described.

1. evout_sample.log

   Files outputted for the number of cycles at the end of computation, world lines information, and vertex information. For restart calculation, calculations are performed with the loaded arrangement as the initial condition.

```
26 : Number of cycles at the end of calculation
0 1 : Information on the world line at site :math:`0` in the domain.
i/N beta, (i+1)/N beta Information on the world line of the section : 0:␣
↪down, 1: up
0 0 :Information on the world line 1 at the site 1 in the domain.
1 1  :Information on the world line 2 at the site 2 in the domain.
...
8 0.056997107 2 1 4  Vertex label, tau, type of vertex, number of world␣
↪lines, bond number
9 0.056997107 2 0 5
44 0.28066013 2 1 3
```

Here, the types of vertexes are defined as follows.

The type of vertex	Description
-5	The start and end points in the imaginary time direction in each domain. It does not need to split domain.
-4(right), -2(left)	Vertex diagonally across domains. It does not need to split domain.
-3(right), -1(left)	Vertex off-diagonally across domains. It does not need to split domain.
0	On site vertex (except worms).
1	Two site vertex
2	Kink.
3	Two site vertex (next nearest) (it is left for compatibility).
4	The worm moving at that time (annihilation operator).
5	The worm stopping at that time (creation operator).
6	The worm stopping at that time (not related to neither annihilation or creation operators) or needless vertex.
7	Marker(for imaginary time correlation function measurement).

2. RNDevout_sample.log

   The output binary file for objects generating random numbers. At recalculation, calculations are performed with the random number information which is read as the initial condition.

5.3 Calculation of the energy on spin chain by DSQSS/PMWA

In this tutorial, you learn how to use DSQSS / PMWA by calculating the ground state energy of antiferromagnetic Heisenberg chain of $S = 1/2$.

The calculation by DSQSS / PMWA is done in the following three stages:

1. Make an input file,
2. Perform calculation,
3. Analyze calculation results.

5.3.1 Make an input file

To run DSQSS/PMWA, following two input files are needed:

1. lattice definition file,
2. parameter file.

The utility tool `pmwa_pre` for doing automatically these process is prepared. This tool is made by a Python script that generates input files for DSQSS/DLA and DSQSS/PMWA from a single input file. First of all, prepare a text file `std.in` with the following contents as an input file for `pmwa_pre`.

```
[System]
solver = PMWA
[Hamiltonian]
model_type = spin
Jxy = -1.0
Jz = -1.0
Gamma = 0.1
[Lattice]
lattice_type = square
D = 1
L = 16
Beta = 100
[Parameter]
CB = 1
SEED = 31415
NSET = 10
NMCS = 100
NPRE = 100
NTHERM = 100
NDECOR = 100
```

This file is located in `sample/pmwa/1DDimer` directory. Next, type the following command:

```
$ pmwa_pre std.in
```

Then, input files `param.in` and `lattice.xml` are generated.

5.3.2 Perform calculation

After making input files, the calculation by `DSQSS/PMWA` starts by typing the following command:

```
$ pmwa_H param.in
```

Random numbers parallel computation is possible by using MPI (it is also possible to perform parallel computation by dividing space/imaginary time. Please see DLA user manual for details).

```
$ mpiexec -np 4 pmwa_H param.in
```

Calculation can be done independently for parallel number (4 for the above example), and the accuracy can be improved by increasing the number of Monte Carlo samples.

5.3.3 Analyze calculation results

The calculation result is written to the output file `sample.log`. The energy is obtained by typing the following command:

```
$ grep ene sample.log
R ene  = -0.5705441 0.0003774399737579577
```

In the case of DSQSS / PMWA, it is necessary to insert a transverse magnetic field. Therefore, please note that extrapolation is necessary to obtain the result at zero magnetic field.

CHAPTER 6

User's manual for DSQSS/PMWA

6.1 Input files for DSQSS/PMWA

In the input file of DSQSS/DLA and DSQSS/PMWA, there are many common parameters. In this section, the parameters which are not included in DSQSS/DLA or where the usage is different from DSQSS/DLA are explained.

- Parameters for calculation conditions

Parameter	type	default	description
RUNTYPE	int		Calculation mode (0: from scratch, 1: restart)
CB	int	0	Initial pattern (0: Vacuum, 1: Checker-Board, 2: Random)
NWORMAX	int	1000	The maximum number of worms. When the value is equal to be -1, the upper limit becomes infinity.
Step_x	int	1	The spatial width in calculating the correlation functions.
Step_k	int	1	The width of the wave-number space in calculating correlation functions of wave-number representation.

- Parameters related to the model

Parameter	type	description
t	double	t in boson system. J_{xy} in spin system.
U	double	U in boson system. This parameter is not used in spin system.
V	double	V in boson system. J_z in spin system.
MU	double	μ in boson system. H in spin system.
G	double	Γ in boson system. $\Gamma/2$ in spin system.
NMAX	-	This value is not used (always fixed to be 1).

An example of the input file is shown as follows:

```
RUNTYPE = 0
NSET  = 10
NMCS  = 1000
NPRE = 1000
NTHERM = 1000
NDECOR = 1000
SEED  = 31415
NC    = 0
NVERMAX = 10000000
NWORMAX = 1000
algfile   = algorithm.xml
latfile   = lattice.xml
outfile   = sample.log
CB      = 2
G       = 0.3
U       = 0
V       = 3
t       = 1
MU      = 2
NMAX    = 1
```

6.2 Output files for DSQSS/PMWA

DSQSS/PMWA outputs one result file and two temporary files for restart calculation (evout_sample.log, RND_evout_sample.log).

- Result file

  Many parameters are common to DSQSS/DLA. In this section, the parameters which are not included in DSQSS/DLA or

Type	Name	Description
P	L	The size of the lattice.
P	DOML	The size of the domain divided by parallelization
P	DOMBETA	The size of the inverse temperature divided by parallelization
P	NDIVL	The number of lattice divisions
P	NTEST	Number of samples to be tested (for details, see the description of Monte Carlo calculation)
R	nver	The number of kinks and worms
R	nkin	The number of kinks
R	wndx	The expected value of square of winding number in x direction
R	wndy	The expected value of square of winding number in y direction
R	wndz	The expected value of square of winding number in z direction
R	wnd2	The total number of squares of winding number(wndx+wndy+wndz)
R	bmxu	The expected value of S_x (uniform τ integral)
R	bmpu	The expected value of S_+ (uniform τ integral)
R	bmmu	The expected value of S_- (uniform τ integral)
R	comp	The compressibility
R	lxmx	The local worm number fluctuation at each site
I	the maximum number of vertices	The maximum number of vertices
I	the maximum number of worms	The maximum number of worms

Here, the type means the letter given to the beginning of each line of output. P, R, I indicate Parameter, Result, Information, respectively.

- File for restart calculation

  PMWA implements the restart function, and if there are two files `evout_sample.log` and `RNDevout_sample.log`, a restart is forcibly performed. In the following, the output contents of each file are briefly described.

  1. evout_sample.log

  In this file, the number of cycles at the end of computation, information on world lines, and vertex information are outputted. In restart calculation, the loaded arrangement is set as the starting condition.

```
26 : Number of cycles at the end of calculation.
0 1 : Information on the world line at 0-th site in the domain.
Information on the world line of the section between i/N beta, (i+1)/N beta; 0:␣
↪down, 1: up
0 0 : Information on the world line at the 1st site in the domain.
1 1 : Information on the world line at the 2nd site in the domain.
...
8 0.056997107 2 1 4 :lavel for vertex, tau, kind of vertex, Number of world lines,
↪ Bond number
9 0.056997107 2 0 5
44 0.28066013 2 1 3
```

PMWA implements the restart function, and if there are two files below, a restart is automatically performed. Below, a brief description of the output contents of each file is described.

1. evout_sample.log

Files outputted for the number of cycles at the end of computation, world lines information, and vertex information. For restart calculation, calculations are performed with the loaded arrangement as the initial condition.

```
26 : Number of cycles at the end of calculation
0 1 : Information on the world line at site :math:`0` in the domain.
i/N beta, (i+1)/N beta Information on the world line of the section : 0:␣
↪down, 1: up
0 0 :Information on the world line 1 at the site 1 in the domain.
1 1  :Information on the world line 2 at the site 2 in the domain.
...
8 0.056997107 2 1 4  Vertex label, tau, type of vertex, number of world␣
↪lines, bond number
9 0.056997107 2 0 5
44 0.28066013 2 1 3
```

Here, the types of vertexes are defined as follows.

The type of vertex	Description
-5	The start and end points in the imaginary time direction in each domain. It does not need to split domain.
-4(right), -2(left)	Vertex diagonally across domains. It does not need to split domain.
-3(right), -1(left)	Vertex off-diagonally across domains. It does not need to split domain.
0	On site vertex (except worms).
1	Two site vertex
2	Kink.
3	Two site vertex (next nearest) (it is left for compatibility).
4	The worm moving at that time (annihilation operator).
5	The worm stopping at that time (creation operator).
6	The worm stopping at that time (not related to neither annihilation or creation operators) or needless vertex.
7	Marker(for imaginary time correlation function measurement).

2. RNDevout_sample.log

   The output binary file for objects generating random numbers. At recalculation, calculations are performed with the random number information which is read as the initial condition.