DSQSS Tutorial

2015.12.01

Y. KATO, A. MASAKI-KATO, Y. MOTOYAMA AND N. KAWASHIMA

associate developers: Kenji Harada, Takafumi Suzuki, Kota Sakakura

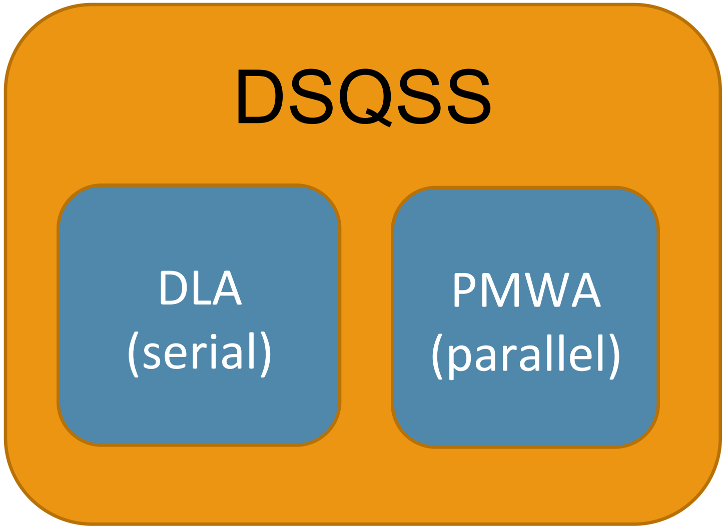
Contents

Part 1. Introduction

What's DSQSS?

DSQSS = **D**iscrete-**S**pace **Q**uantum **S**ystems **S**olver

A program package currently including quantum Monte Carlo codes based on the worm algorithm (incl. directed loop algorithm) with some handy tools (for lattice definition,
model definition,
data collecting, plotting, etc)



What you can/can't do with DSQSS?

You can do almost all quantum lattice model, provided that it does not cause negative sign problem, which means...

DSQSS is good for

(1) non-frustrated XXZ model (e.g., Heisenberg model) with general spins, on any lattice in any dimensions, with or without magnetic field.

(2) Bose-Hubbard model on any lattice in any dimensions, at an arbitrary chemical potential.

DSQSS is not good for

(1) frustrated model

(2) fermion systems

How you can use DSQSS

- Installing of your own DSQSS (UNIX environment required)
  --- download from github

- Use preinstalled version
  --- you can keep using psi for a while (probably easiest)
  --- get an account on ISSP supercomputer (for serious users)

- Using the "MateriApps Live!" package (UNIX env. included)
  --- visit "MateriApps" site

Source of More Information

"dsqss github" on google

https://github.com/qmc/dsqss/wiki

Monte Carlo Basics --- Importance Sampling

$$\langle Q\rangle \equiv \int dX\, W(X)Q(X) \Big/ \int dX\, W(X)$$

For example,

weight $W(x_1,x_2,\ldots,x_N)$ $=1$(inside the "square"), $=0$(otherwise)

observable $Q(x_1,x_2,\ldots,x_N)$ $=1$(inside the "circle"), $=0$(otherwise)

Our task: "Generate $X\equiv(x_1,x_2,\ldots,x_N)$ with the frequency $W(x_1,x_2,\ldots,x_N)$"

(1) Throw-and-discard rule : First generate a candidate X at random, and accept it with the probability proportional to W.

(2) Throw-and-stack rule: Make an attempt to modify X to X' at random. Accept it with the probability $W(X')/W(X)$. Even if it's rejected, still count the current state again.

Monte Carlo Basics --- Markov Chain

$$X_0 \rightarrow X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow \mathrm{L}$$

X_t :state at the t-th step

$T\left(X' \mid X\right)$:transition probability

$P_t\left(X_t\right)$: the probability of having X_t at the t-th step

$$P_{t+1}\left(X_{t+1}\right) = \sum_{X_t} T\left(X_{t+1} \mid X_t\right) P_t\left(X_t\right)$$

or simply,

$$\boxed{\boldsymbol{P}_{t+1} = T\boldsymbol{P}_t}$$

Monte Carlo Basics --- Detailed Balance

Designing a Markov chain

We demand, $$\boxed{T_{ij}W_j = T_{ji}W_i}$$

W_i : the target distribution

$$\Rightarrow \qquad \boldsymbol{TW} = \boldsymbol{W}$$

Monte Carlo Basics --- Ergodicity

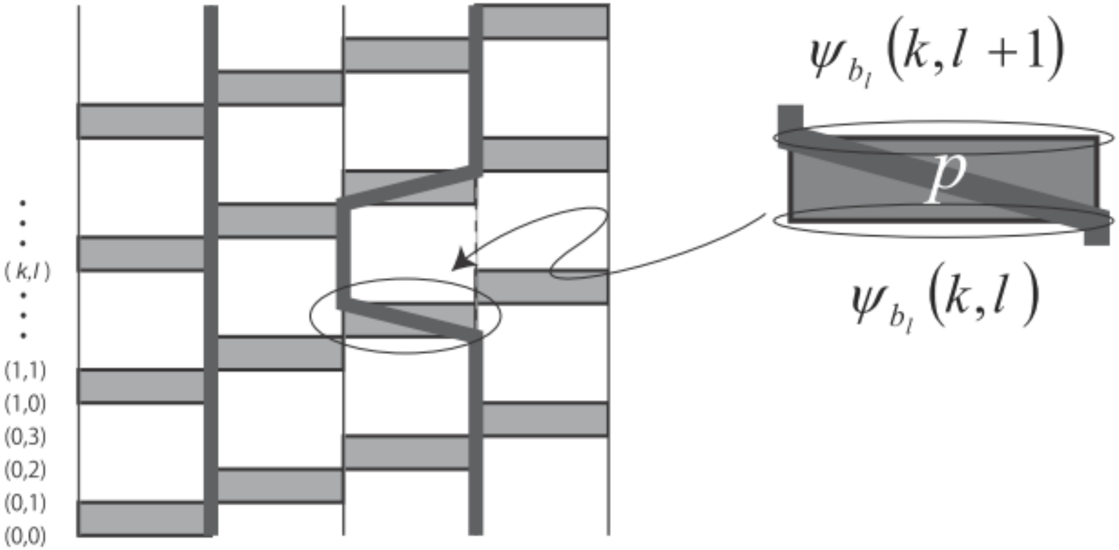
"After a sufficiently long time, any state can appear."

$$\exists t_0 \forall t > t_0 \forall \left(i, j \right) \left(\left(T^t \right)_{ij} > 0 \right)$$

- T must be irreducible.
- Cyclic solution should be excluded.

QMC Basics --- Path-Integral

$$Z \approx \sum_{S=\{\psi(k,l)\}} \prod_{k=0}^{M-1} \prod_{l=0}^{N_l-1} \left\langle \psi(k,l+1) \middle| e^{-\Delta\tau H_l} \middle| \psi(k,l) \right\rangle$$

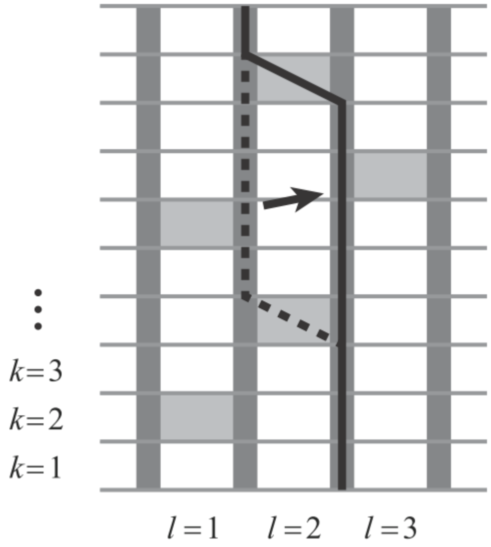


QMC Basics --- Conventional Update

$$Z_M = \sum_{\{\gamma_{k,l}\}'} \frac{(M-n)!}{M!} \beta^n \operatorname{Tr}\left(\prod_{k=1}^{M} \prod_{l=1}^{N_l} (-H_l)^{\gamma_{k,l}} \right)$$

(i) Vertex (= filled cell) Update:

$$p_l^{(\mathrm{fill})} \times \langle \psi(k+1) \,|\, \psi(k) \rangle$$
$$= p^{(\mathrm{empty})} \times \frac{\beta}{M-n} \left\langle \psi(k+1) \,\middle|\, -H_l \,\middle|\, \psi(k) \right\rangle$$

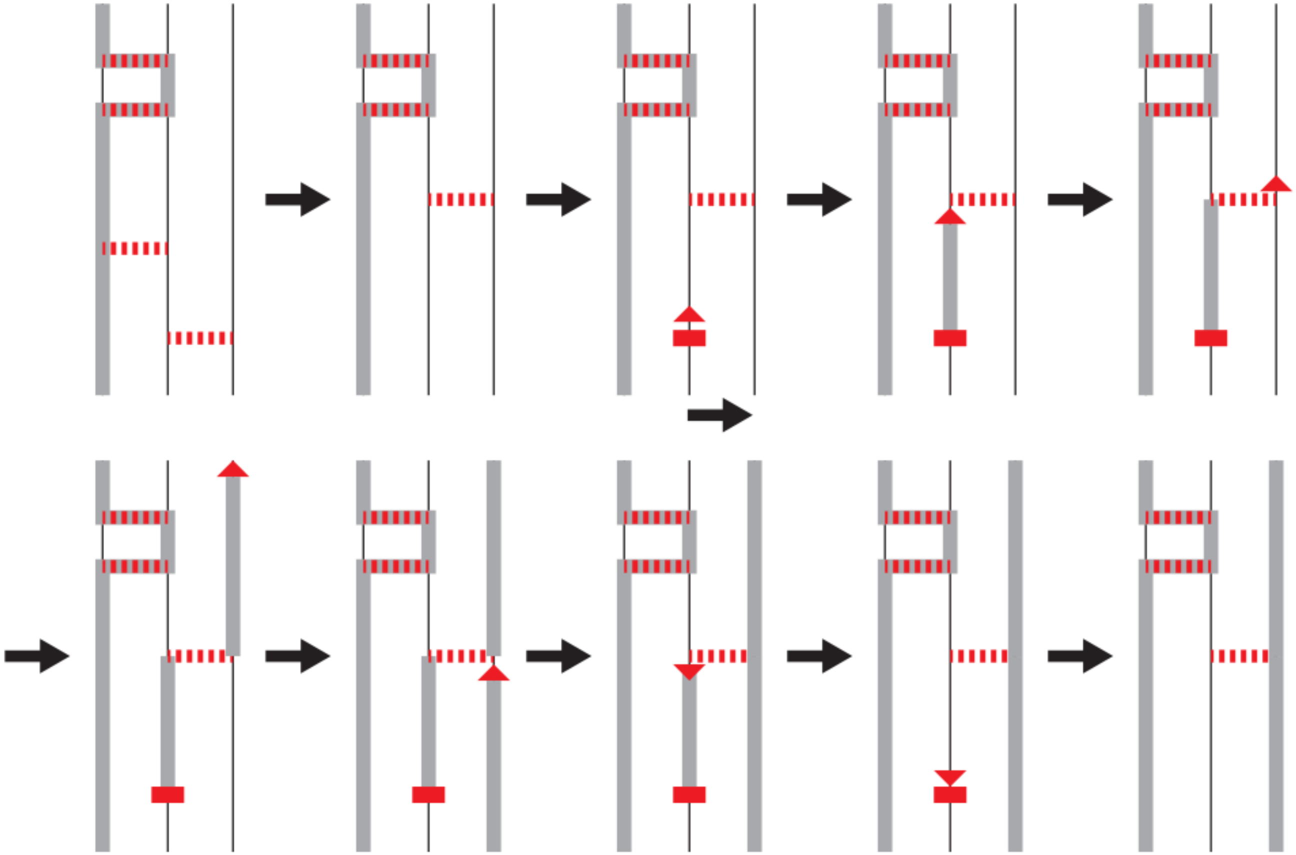


(ii) World-Line Update:

$$p_{\mathrm{accept}} = \min(1, R) \ \text{ with } R \equiv \prod_{(k,l)} \frac{\langle \psi'(k+1) \,|\, -H_l \,|\, \psi'(k) \rangle}{\langle \psi(k+1) \,|\, -H_l \,|\, \psi(k) \rangle}$$

QMC Basics --- SSE Update

World-Line Update with Worms



SSE ("On-the-Fly" Version)

We can generate vertices only when they are necessary.

First Scattering Time $$\tau_{\text{first}} = \frac{1}{a} \log \frac{1}{r}$$

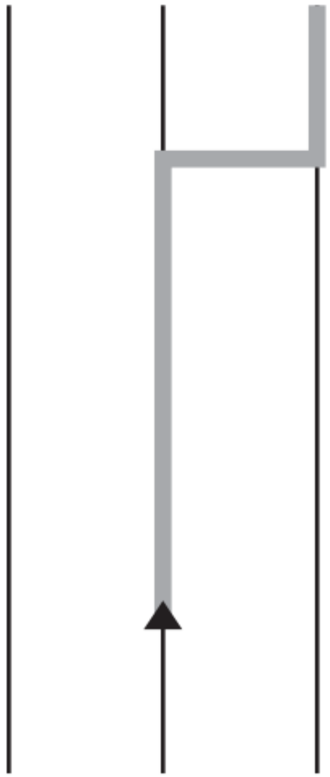
Scattering Rate $$a \equiv \sum_l a_l p_l^{\text{n.f.}}$$

$$a_l \equiv \langle \psi(\tau) | - H_l | \psi(\tau) \rangle$$

Interaction-Term
Selection Probability $$p_l \equiv \frac{a_l p_l^{\text{n.f.}}}{a}$$

Scattering Probability

$$p_l'(\nu \mid \mu) = \frac{p_l(\nu \mid \mu)}{p_l^{\text{n.f.}}}$$



SSE ("On-the-Fly" Version)

We can generate vertices only when they are necessary.

First Scattering Time $$\tau_{\text{first}} = \frac{1}{a} \log \frac{1}{r}$$

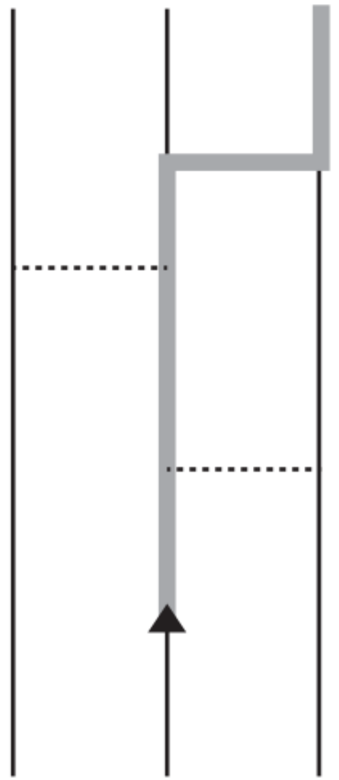
Scattering Rate $$a \equiv \sum_l a_l p_l^{\text{n.f.}}$$

$$a_l \equiv \langle \psi(\tau) | - H_l | \psi(\tau) \rangle$$

Interaction-Term
Selection Probability $$p_l \equiv \frac{a_l p_l^{\text{n.f.}}}{a}$$

Scattering Probability

$$p_l'(\nu \mid \mu) = \frac{p_l(\nu \mid \mu)}{p_l^{\text{n.f.}}}$$



SSE ("On-the-Fly" Version)

We can generate vertices only when they are necessary.

First Scattering Time $$\tau_{\text{first}} = \frac{1}{a}\log\frac{1}{r}$$

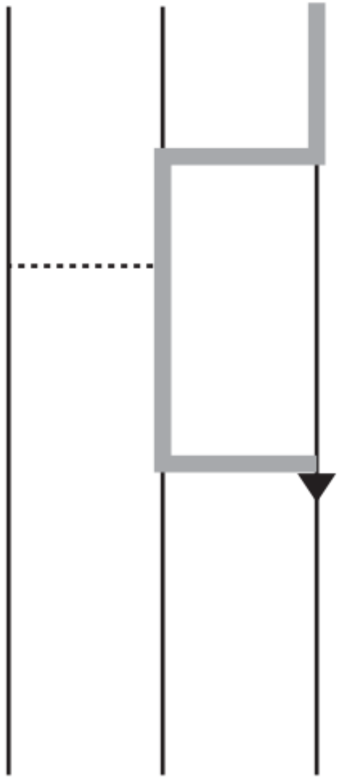
Scattering Rate $$a \equiv \sum_l a_l p_l^{\text{n.f.}}$$

$$a_l \equiv \langle\psi(\tau)| - H_l |\psi(\tau)\rangle$$

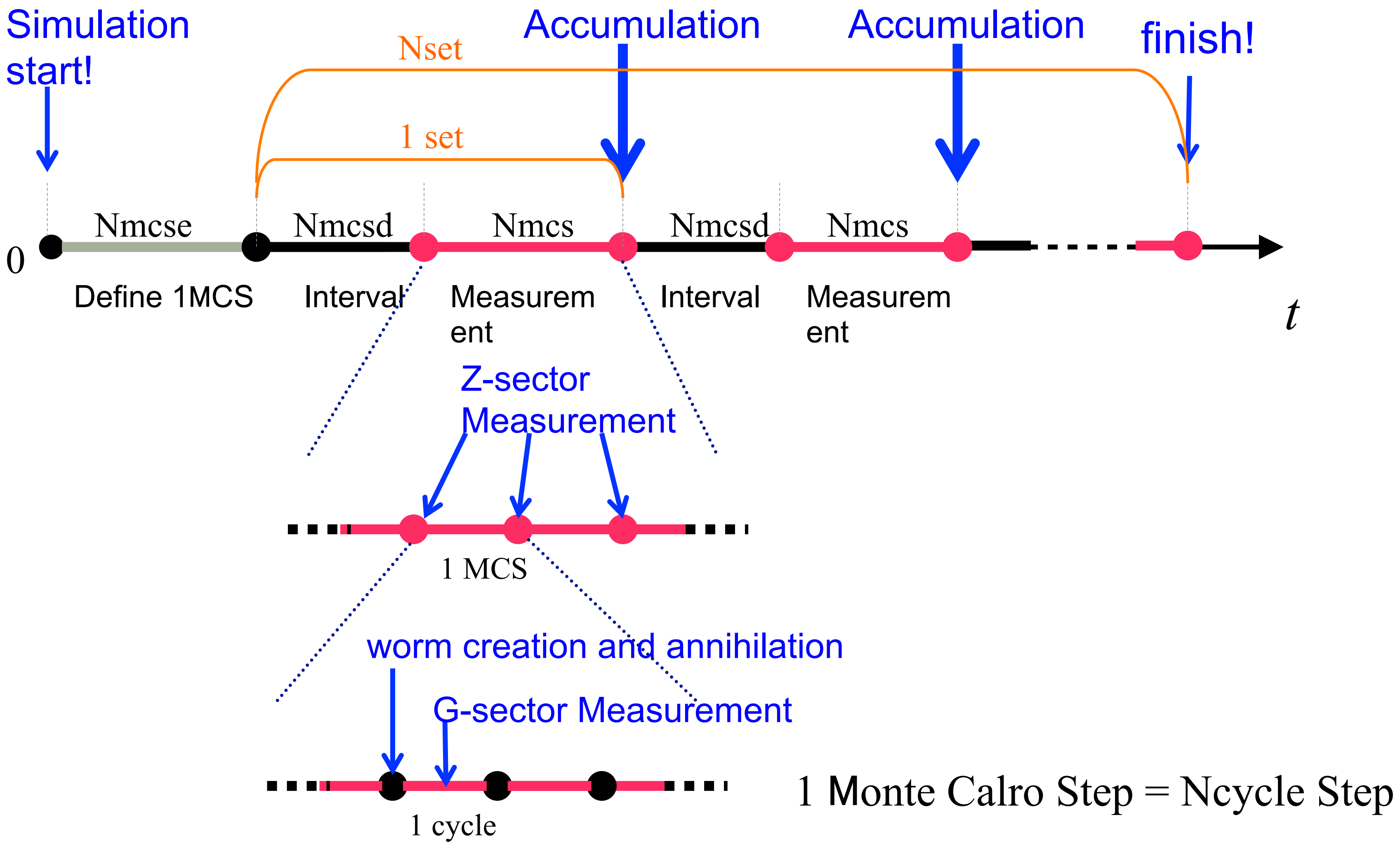
Interaction-Term Selection Probability $$p_l \equiv \frac{a_l p_l^{\text{n.f.}}}{a}$$

Scattering Probability

$$p_l'(\nu\,|\,\mu) = \frac{p_l(\nu\,|\,\mu)}{p_l^{\text{n.f.}}}$$



How is the whole simulation organized ...



One Monte Carlo Step

Step1 :Erase all vertices without a kink on it.

Step2 :Put vertices with the probability.

Step3 : Repeat Ncycle until the total length of a worm-head moving reached to the volume of the configuration space.

Step4 : Perform *Z*-sector measurements without worms.

PMWA is implemented this fixed−vertex update method.
On the other hand DLA in dsqss is applied to on-the-fly update method, which does not have step1, 2 instead of putting and erasing kinks without vertices by a moving worm head during step 3.

One Cycle

Step1 :Choose a site and an imaginary time point.

Step2 :Put a worm pair. if no, go to Step4.

Step3 :The worm-head moving. When the head meets the tail, annihilate the worm pair and go to Step4.

Step4 : Accumulate the *G*-sector measurements then go to Step1.

Examples of *G*-sector measurements:

S^xS^x correlator, S^x susceptibility etc…

Part 2. Exercise

Install procedure

Initial setting

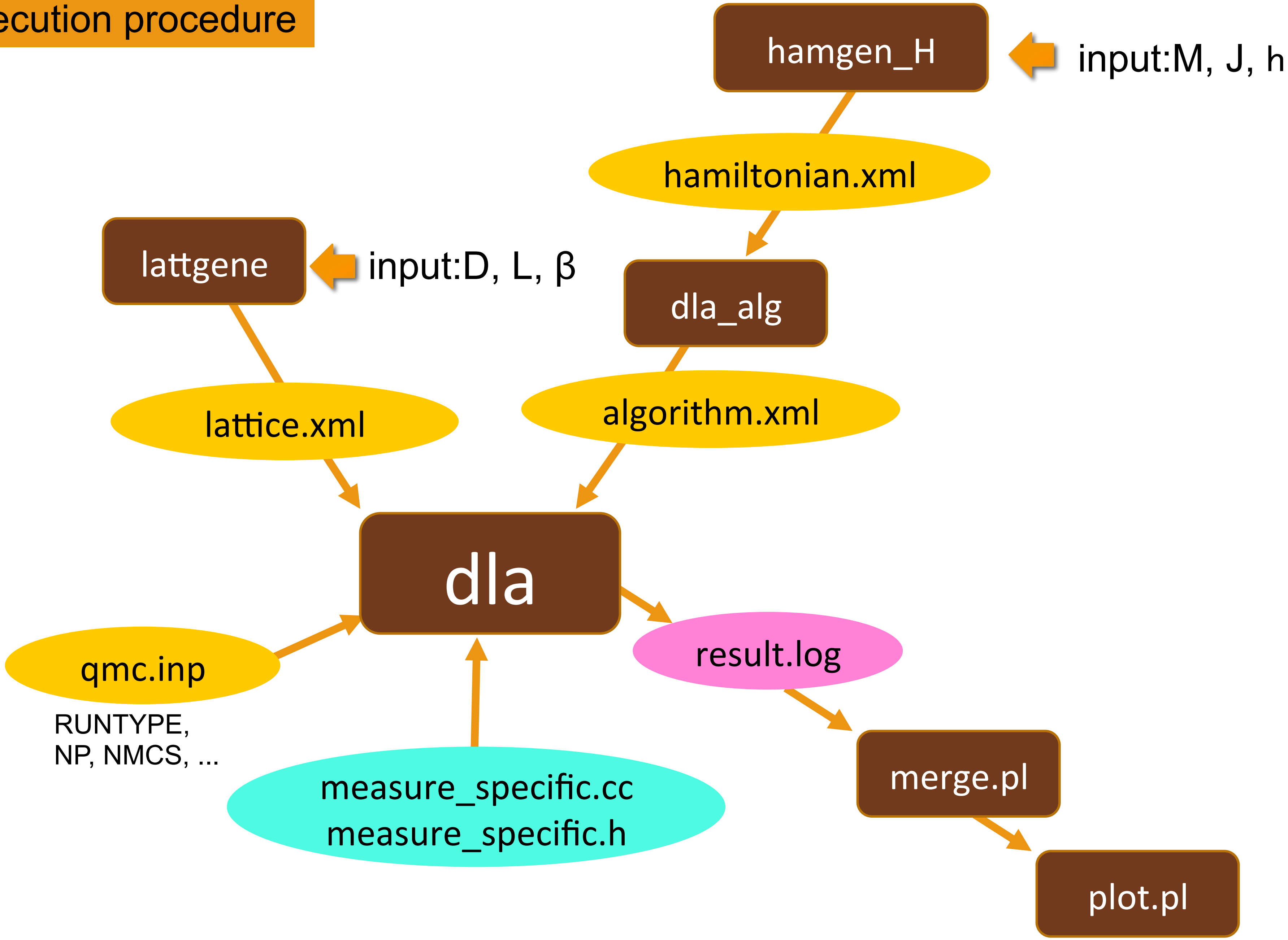
```
$ wget https://github.com/qmc/dsqss/tarball/v1.1.17+pv1.1.3
$ tar xvf v1.1.17+pv1.1.3
$ cd qmc-dsqss-0109476
$ vi runConfigure.sh
$ ./runConfigure.sh
$ make
```

Set path and the environment variable

```
$ source ./bin/wormvars.sh
```

Environment variable : $WORM_HOME

execution procedure



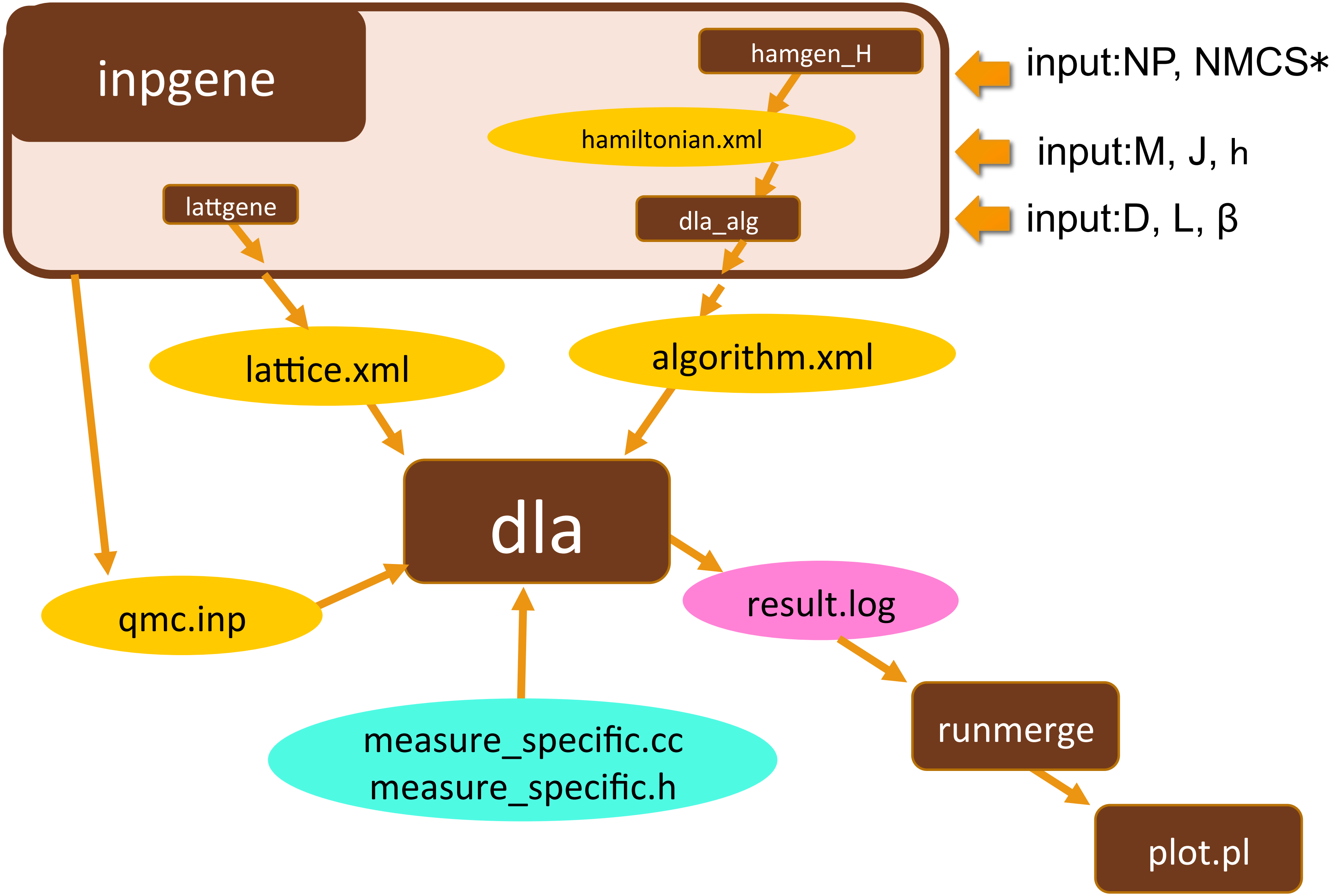
RUNTYPE : 4 different type of run

RUNTYPE:
0 → no parallelization or trivial parallelization for parameters
1 → trivial parallelization for replicas for parameters
2 → trivial parallelization for replicas for inverse of temperature
3 → nontrivial parallelization with domain decompositions

The number of parallelization:

- NP: total number of parallelization
    - NPTP: trivial parallelization for parameters
    - NREP: replicas
    - NPNT: nontrivial parallelization
        - NL: the number of parallelization for spatial axes
        - NB: the number of parallelization for the temporal axes

execution procedure by “inpgene” : an input-files generator



definitions of physical quantities

$S^z_i(\tau)$ corresponds to the number of world line on i-th site at τ and means $S^z_i=S^z_i(0)$.

amzu: $$\frac{1}{N_{site}}\left\langle \sum_{i=1}^{N_{site}} S_i^z \right\rangle$$

bmzu: $$\frac{1}{N_{site}\beta}\left\langle \sum_{i=1}^{N_{site}} \int_0^\beta S_i^z(\tau)\,d\tau \right\rangle$$

smzu: $$\frac{1}{N_{site}}\left\langle \left(\sum_{i=1}^{N_{site}} e^{ir_i k} S_i^z \right)^2 \right\rangle_{k=0}$$

xmzu: $$\frac{1}{N_{site}\beta}\left\langle \left(\sum_{i=1}^{N_{site}} e^{ir_i k} \int_0^\beta S_i^z(\tau)\,d\tau \right)^2 \right\rangle_{k=0}$$

smzs, xmzs: k=0 → k=π

xmx (len): $$\frac{1}{2N_{site}\beta}\sum_{i,j=1}^{N_{site}}\int_0^\beta\int_0^\beta \left\langle S_i^+(\tau)S_j^-(\tau')\right\rangle d\tau\, d\tau' = \frac{1}{2N_{site}\beta\eta^2}\mathrm{len} = \frac{1}{2}\left(M-\frac{1}{2}\right)\mathrm{len}$$

len: the traveling length of a worm per cycle.

η : the weight of worm. We choose as $\eta = \left[N_{site}\beta\left(M-\frac{1}{2}\right)\right]^{-\frac{1}{2}}$

Hamiltonian of sample program 1

①exercise 1-3 in the hands-on manual (DLA)

S=1/2 Antiferromagnetic (AFM) Heisenberg on one-dimensional chain

$$\mathcal{H} = -J \sum_{<i,j>} \mathbf{S}_i \cdot \mathbf{S}_j - H \sum_i S_i^z$$

FM: J>0
AFM:J<0

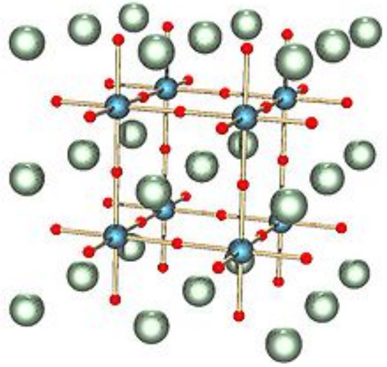
Input parameters:

M: S=1/2,1,3/2,…, for 1,2,3,…
J : coupling constant
H : magnetic field

D: dimension
L : linear−system size
β : Inverse of tempareture

examples of materials

- Sr_2CuO_3 (1D-AFM)
- $KCuF_3$ (1D-AFM)

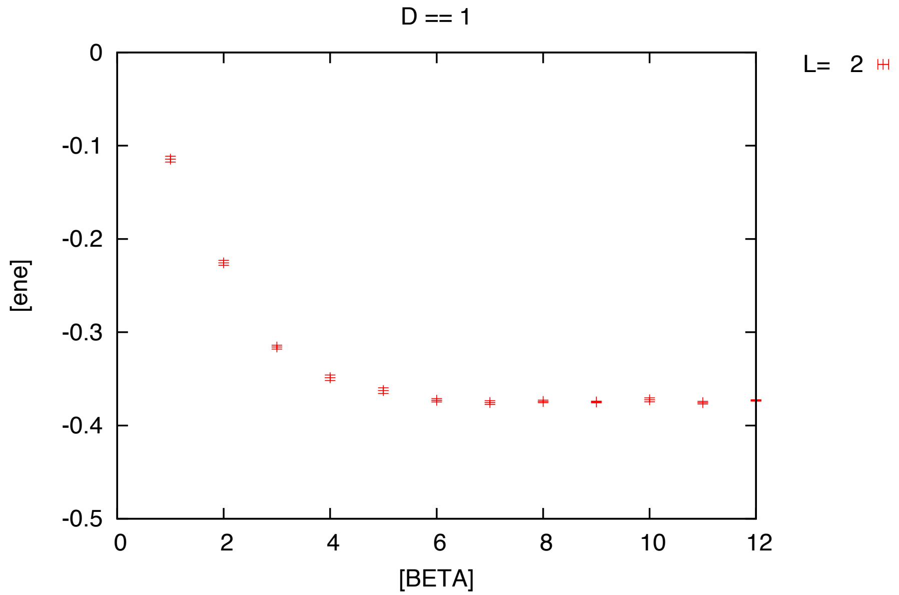


perovskite structure
[Ref. wikipedia]

exercise 2:

β dependence of the energy.

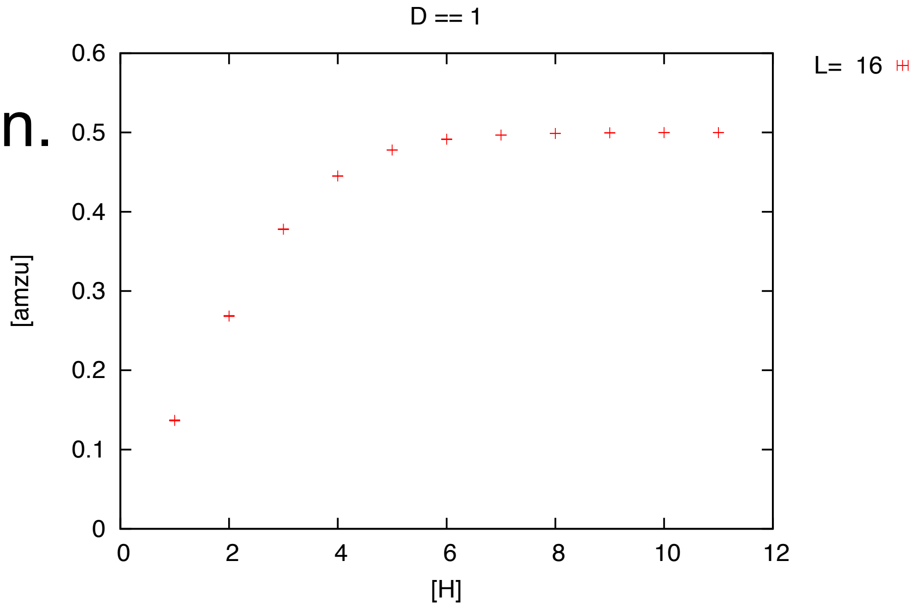
NPTP=12, NREP=NPNT=NPTS=1
M=1, J=-0.5, H=0.0,
D=1, L=2, β=1.0-12.0,
MCS=1000



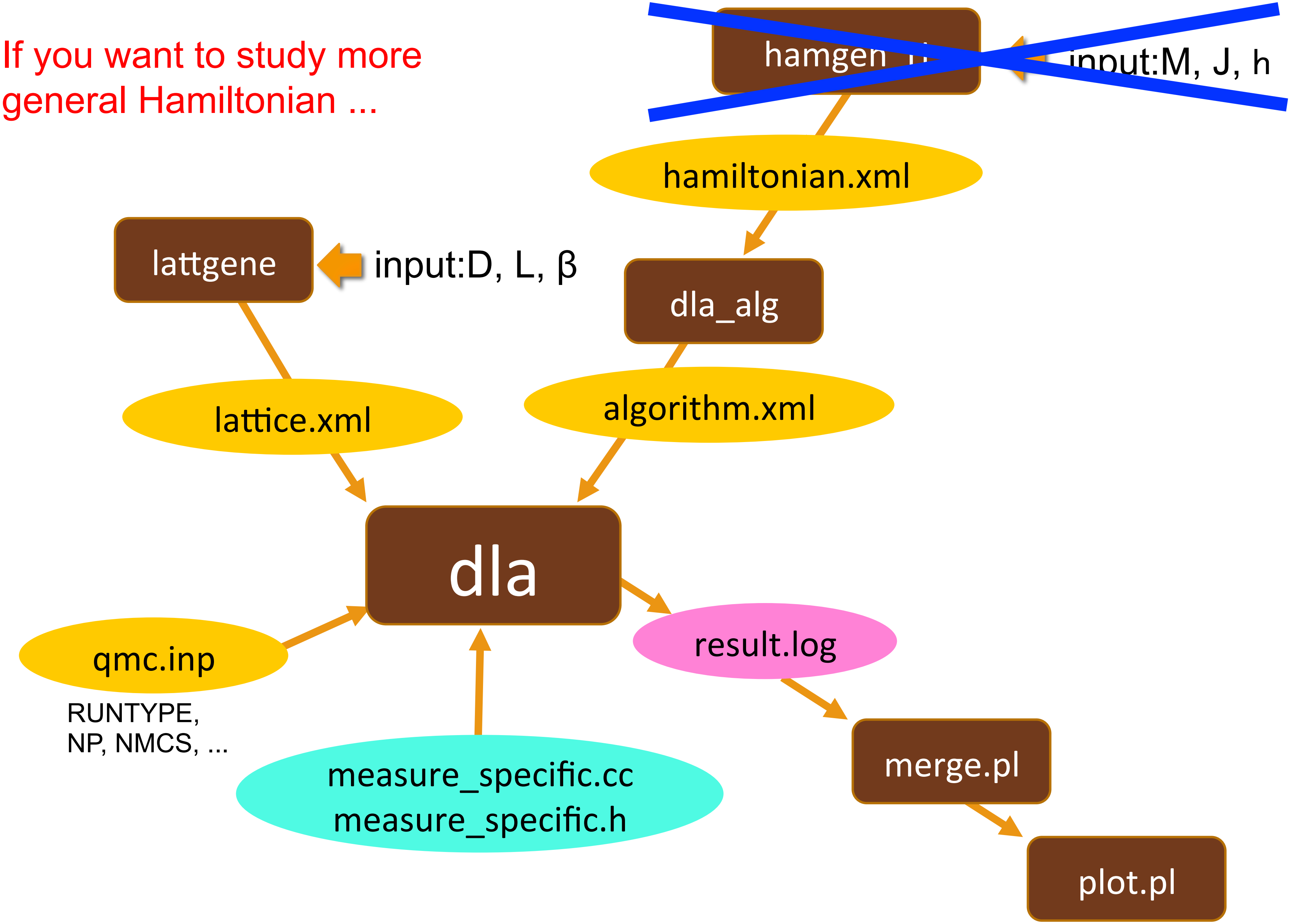
exercise 3:

H dependence of the magnetization.

NPTP=12, NREP=NPNT=NPTS=1
M=1, J=-1.0, H=0.0-11.0
D=1, L=16, β=1.0,
MCS=1000



If you want to study more general Hamiltonian ...



How to study general Hamiltonian*

```
$ vi dsqss/dsqss-1.1.17/src/measure_specific.h
$ vi dsqss/dsqss-1.1.17/src/measure_specific.cc
$ make
$ hamgen_H
$ lattgene
$ vi yourscript_for_xml.sh
$ ./yourscript_for_xml.sh
$ dla_alg
$ dla qmc.inp
```

1) If observables what you need are not in default measurement-files (measure_specific.*), prepare those. (See Part. 3)
2) Do “make”.
3) Generate hamiltonian.xml and lattice.xml by **default** hamgen_H and lattgene.
4) Modify the hamiltonian.xml and lattice.xml to implement the arbitrary models using script files written by yourself. you will find an example of the script in our hands-on manual, exercise 5.
5) Generate algorithm.xml by **default** dla_alg.
6) Execute dla.

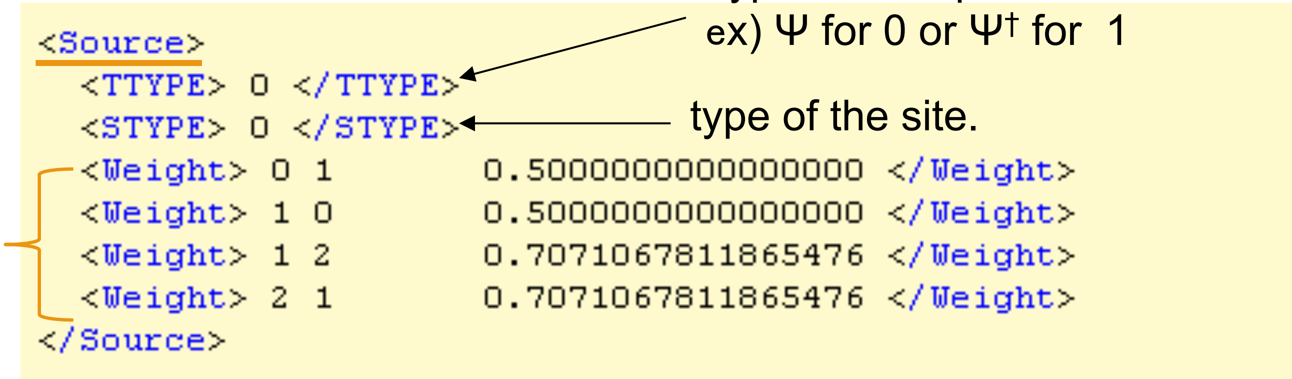
You can define arbitrary lattice in a similar way.

* At present, general Hamiltonian/lattice support is not available for pmwa.

About xml files ①＜hamiltonian.xml＞－1

<Source> … </Source> gives weights of a worm head or tail.

ex) softcore boson.



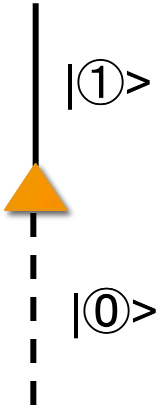
```
<Source>
  <TTYPE> 0 </TTYPE>
  <STYPE> 0 </STYPE>
  <Weight> 0 1       0.500000000000000 </Weight>
  <Weight> 1 0       0.500000000000000 </Weight>
  <Weight> 1 2       0.707106781186548 </Weight>
  <Weight> 2 1       0.707106781186548 </Weight>
</Source>
```

Type of the operator of the tail.
ex) Ψ for 0 or Ψ† for 1

type of the site.

<Weight> ⓪ ① ② </ Weight>

⓪ Local state below the head
① Local state above the head
② Weight of the configuration



$$W_{n\,n+1} = \langle \underset{①}{n} | \hat{Q} | \underset{⓪}{n+1} \rangle = \underset{②}{\frac{1}{2}\sqrt{n}},$$

$$\hat{Q} = \frac{1}{2}\left(\psi + \psi^{\dagger}\right)$$

Bosonic annihilation and creation operator.

About xml files ①＜hamiltonian.xml＞－2

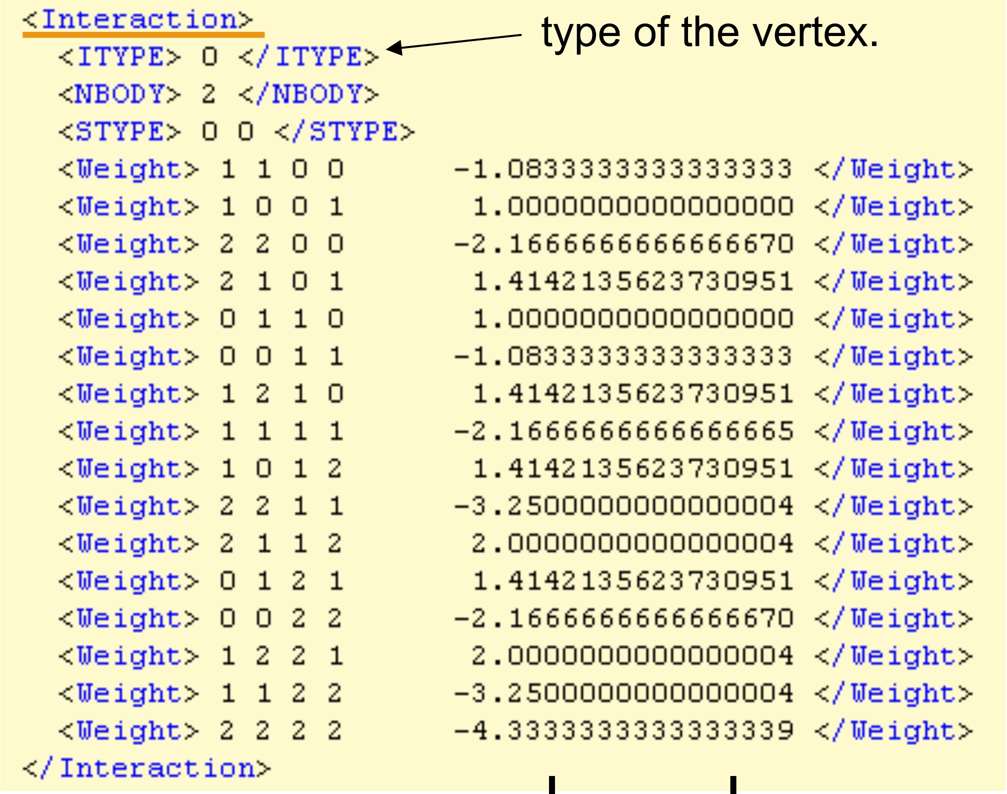
<Interaction> … </Interaction> gives weights of vertices.

ex) softcore boson.

$$H_{ij} = -t(\psi_i^{\dagger}\psi_j + \psi_i\psi_j^{\dagger})$$

$$+\mu(n_i + n_j) + \frac{U}{2}(n_i^2 + n_j^2)$$

$$W = \left\langle n_i^{initial} n_j^{initial} \right| -H_{ij} \left| n_i^{final} n_j^{final} \right\rangle$$



```
<Interaction>
  <ITYPE> 0 </ITYPE>
  <NBODY> 2 </NBODY>
  <STYPE> 0 0 </STYPE>
  <Weight> 1 1 0 0      -1.083333333333333 </Weight>
  <Weight> 1 0 0 1       1.000000000000000 </Weight>
  <Weight> 2 2 0 0      -2.166666666666670 </Weight>
  <Weight> 2 1 0 1       1.414213562373095 </Weight>
  <Weight> 0 1 1 0       1.000000000000000 </Weight>
  <Weight> 0 0 1 1      -1.083333333333333 </Weight>
  <Weight> 1 2 1 0       1.414213562373095 </Weight>
  <Weight> 1 1 1 1      -2.166666666666665 </Weight>
  <Weight> 1 0 1 2       1.414213562373095 </Weight>
  <Weight> 2 2 1 1      -3.250000000000004 </Weight>
  <Weight> 2 1 1 2       2.000000000000004 </Weight>
  <Weight> 0 1 2 1       1.414213562373095 </Weight>
  <Weight> 0 0 2 2      -2.166666666666670 </Weight>
  <Weight> 1 2 2 1       2.000000000000004 </Weight>
  <Weight> 1 1 2 2      -3.250000000000004 </Weight>
  <Weight> 2 2 2 2      -4.333333333333339 </Weight>
</Interaction>
```

type of the vertex.

<NBODY> 1 for onsite, 2 for 2 sites vertex.

<Weight> ⓪ ① ② ③ ④</ Weight>

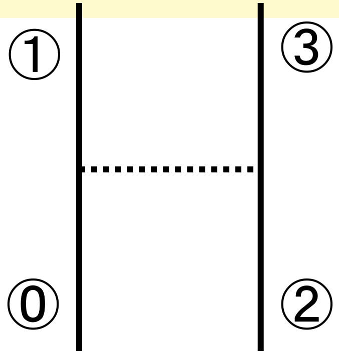
⓪ initial state (left hand side).

① final state (left hand side).

② initial state (right hand side).

③ final state (right hand side).

④ weight.



∗You must determine “EBASE” so that Diagonal elements are positive!

Hamiltonian of sample program 2

②exercise 4,5 in the hands-on manual (DLA)

S=1 Heisenberg model with an easy-axis isotropy on one-dimensional chain.

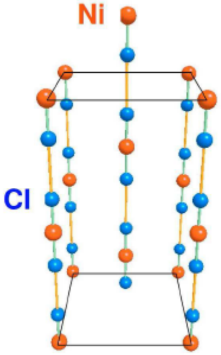
$$\mathcal{H} = -J\sum_{<i,j>}\mathbf{S}_i\cdot\mathbf{S}_j - H\sum_i S_i^z - G\sum_i \left(S_i^z\right)^2$$

Input parameter:

G: easy-axis isotropy

examples of materials

●$NiCl_2$-$4SC(NH_2)_2$



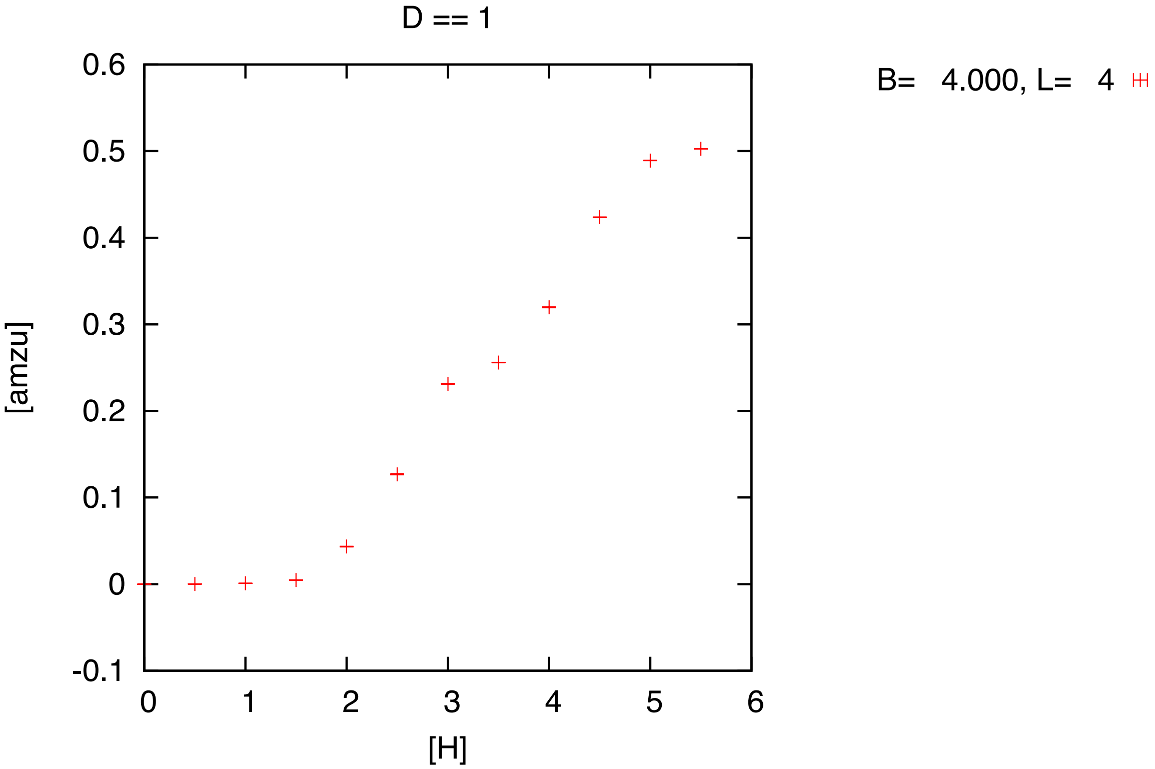
Dichloro-tetrakis thioureanickel
[Ref. A. Paduan-Filho, Brazilian J. phys. 42, 292 (2012)]

exercise 5:

sample code : “samples/manual_run”

H dependence of the magnetization.

NPTP=12, NREP=NPNT=NPTS=1
M=2, J=-1.0, H=0.0-5.5 ,
G=-4.0 (easy-plane)
D=1, L=4, β=4.0,
MCS=10000



③exercise 6 in the hands-on manual (PMWA)

S=1/2 XXZ model on a hypercubic lattice

$$\mathcal{H} = -J_{xy} \sum_{<i,j>} \left(S_i^x S_i^x + S_i^y S_i^y \right) - J_z \sum_{<i,j>} S_i^z S_i^z - H \sum_i S_i^z - \Gamma \sum_i S_i^x$$

Input parameters:

Jxx: coupling constant of xy term
Jz : coupling constant of z term
H : longitudinal magnetic field
Γ : transvers magnetic field (source field for introducing worms)

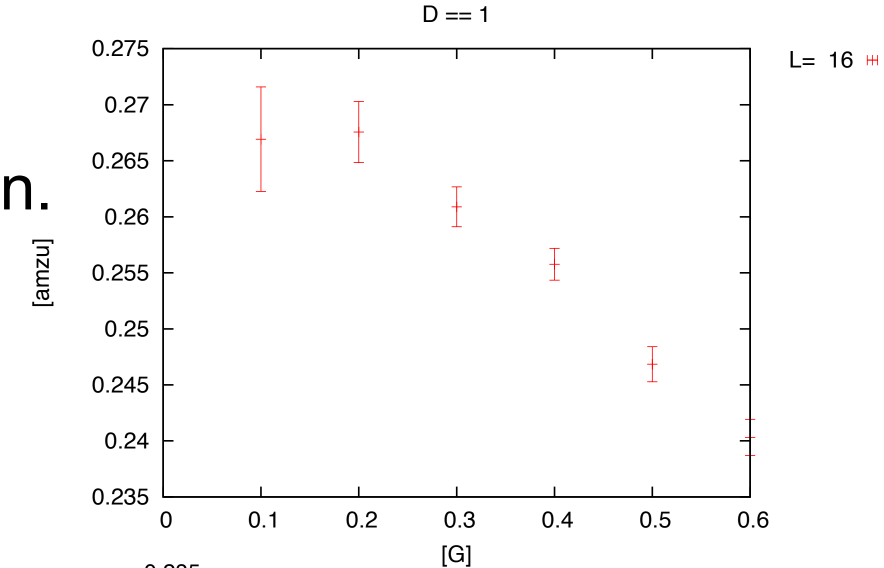
[Ref. A. Masaki-Kato, T. Suzuki, K. Harada, S. Todo, and N. Kawashima, PRL 112, 140603 (2014)]

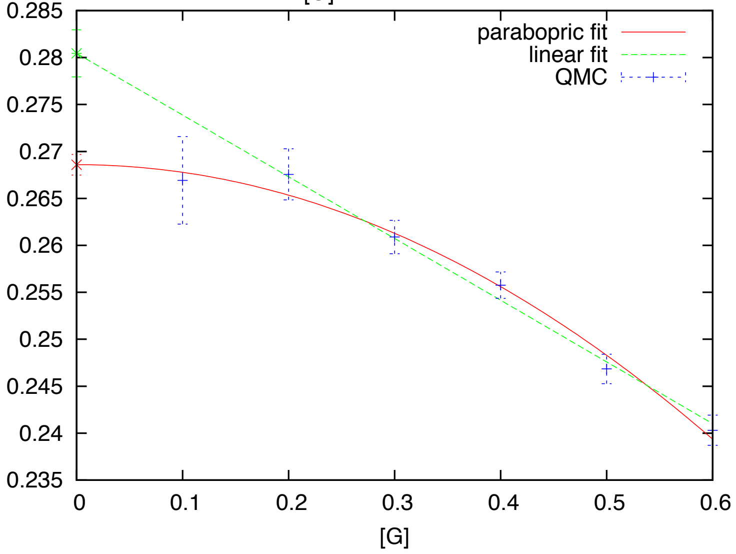
exercise 6:

extrapolation: “extrap.pl”

Γ dependence of the magnetization.

NREP=1, NPNT=2, NPTS=1, NPTP=6,
Jxy=-1.0, Jz=-1.0, H=2.0,
Γ=0.1-0.6
D=1, L=16, β=1.0,
NL=2, NB=1
MCS=10000
(refer exercise 3, H=2.0)





extrapolation values in G=0 limit are shown in Plot/*.param.

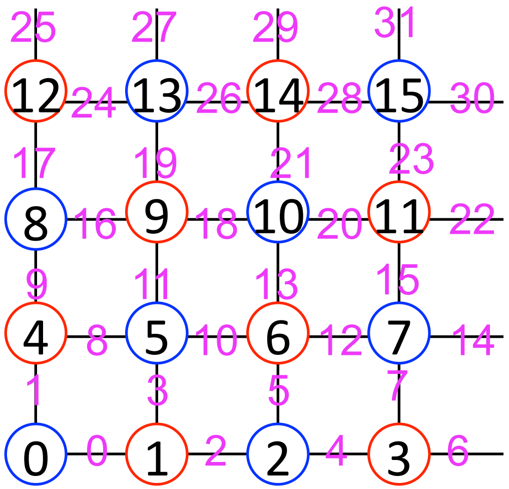
```
$ cat Plot/*.param
0.0 0.268592363974903 0.00109840530233336 0.28044765218082 5 0.0025071656843044
```

#[G=0.0] [extrapolation with a linear fitting] [error of the linear fitting] [extrapolation with a parabolic fitting] [error of the parabolic fitting]

NOTE: This algorithm is recommended to calculate with large L and β, and Γ<1/L.

About xml files ②<lattice.xml>－1

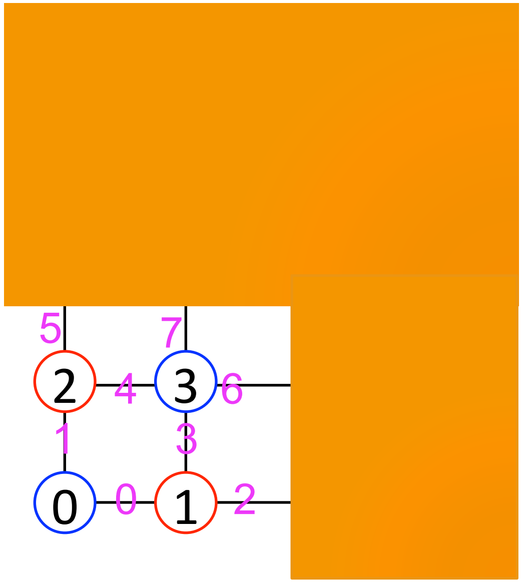
Serial version



```
<Dimension> 2 </Dimension>
<LinearSize> 4 4 </LinearSize>
<Beta> 8 </Beta>
<NumberOfCells> 16 </NumberOfCells>
<NumberOfSites> 16 </NumberOfSites>
<NumberOfInteractions> 32 </NumberOfInteractions>
<NumberOfSiteTypes> 1 </NumberOfSiteTypes>
<NumberOfInteractionTypes> 1 </NumberOfInteractionTypes>

<!-- <S> [id] [stype] [mtype] </S> -->

<S> 0 0 0 </S>
<S> 1 0 1 </S>
…
<S> 14 0 1 </S>
<S> 15 0 0 </S>

<!-- <I> [id] [itype] [nbody] [s0] [s1] … </I> -->

<I> 0 0 2 0 1 </I>
<I> 1 0 2 0 4 </I>
…
<I> 30 0 2 15 12 </I>
<I> 31 0 2 15 3 </I>
```

mtype: The phase factor for measuring to k-space quantities.
For example, when k=π, blue sites take exp(-ikr)=1 labeling with mtype=0, red sites are -1 with mtype=1.

stype, itype: <NumberOfSites> =<NumberOfCells> *<NumberOfSiteTypes> .
If you define an unit cell, sites in a cell are labeled with stype and itype.



edge id (>= 0) **:** label of inter-domain interactions.

-1 : intra-domain interaction.

```
<Dimension> 2 </Dimension>
<LinearSize> 4 4 </LinearSize>
<NumberOfLDecomposition> 2 </NumberOfLDecomposition>
<LinearDomainSize> 2 2 </LinearDomainSize>
<Beta> 8 </Beta>
<OldBeta> 8 </OldBeta>
<NumberOfBDecomposition> 2 </NumberOfBDecomposition>
<BetaOfDomain> 4 </BetaOfDomain>
<NumberOfCells> 4 </NumberOfCells>
<NumberOfSites> 4 </NumberOfSites>
<NumberOfInteractions> 8 </NumberOfInteractions>
<NumberOfSiteTypes> 1 </NumberOfSiteTypes>
<NumberOfInteractionTypes> 1 </NumberOfInteractionTypes>
<NumberOfExternalField> 0 </NumberOfExternalField>

<!-- <S> [id] [stype] [mtype] </S> -->

<S> 0 0 0 </S>
<S> 1 0 1 </S>
<S> 2 0 1 </S>
<S> 3 0 0 </S>

<!-- <I> [id] [itype] [nbody] [edge id] [s0] [s1] ... </I> -->

<I> 0 0 2 -1 0 1 </I>
<I> 1 0 2 -1 0 2 </I>
<I> 2 0 2 0 1 0 </I>
…
<I> 7 0 2 3 3 1 </I>
```

Part 3. Advanced

About objects in measure.hpp : To edit “measure_specific”

measure_specific.h:

enum { ..., NACC }; ← fill names of quantities measured at each MC step in “…”.
enum { ..., NPHY }; ← fill names of quantities measured at each set in “…”.

“enumeration” automatically assign integer values to its members.

→ DON’T touch NACC and NPHY

measure.hpp:

Accumulator* ACC; → accumurator for snapshot values
Accumulator* PHY; → accumurator for set averages

void Accumulator::accumulate(x); → sum up x to s1. here x is double. $s1 = \sum_{i=1}^{n} x_i$
void Accumulator::average(); → averaged over x.
double Accumulator::value; → after done average(), set the averaged value

About algorithm.xml

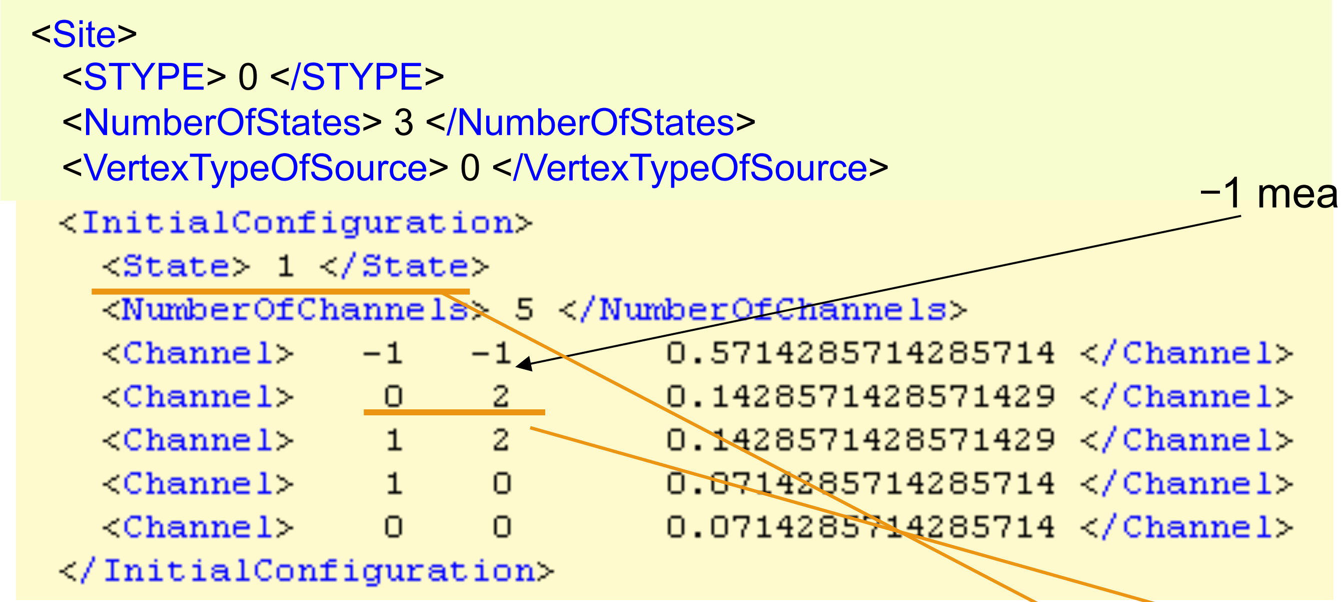
What is algorithm.xml ?

Vertex densities and tables of worm-scattering probabilities are shown in "algorithm.xml" by reading hamiltonian.xml.

By changing this file, you can apply other algorithms of Markov-chain Monte Carlo methods.

About xml files ③＜algorithm.xml＞－1

<Site> … </Site> gives probabilities of putting a worm pair on each “State”.



```
<Site>
  <STYPE> 0 </STYPE>
  <NumberOfStates> 3 </NumberOfStates>
  <VertexTypeOfSource> 0 </VertexTypeOfSource>
 <InitialConfiguration>
   <State> 1 </State>
   <NumberOfChannels> 5 </NumberOfChannels>
   <Channel>    -1    -1         0.5714285714285714 </Channel>
   <Channel>     0     2         0.1428571428571429 </Channel>
   <Channel>     1     2         0.1428571428571429 </Channel>
   <Channel>     1     0         0.0714285714285714 </Channel>
   <Channel>     0     0         0.0714285714285714 </Channel>
 </InitialConfiguration>
```

−1 means no worm.

<State> The initial state.

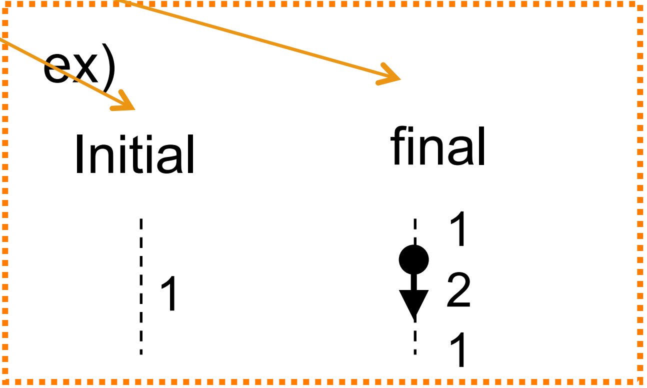
<NumberOfChannels>The number of channels without zero probabilities.

<Channel> ⓪ ① ② </ Channel>

⓪ direction of the head (1 for up, 0 for down)

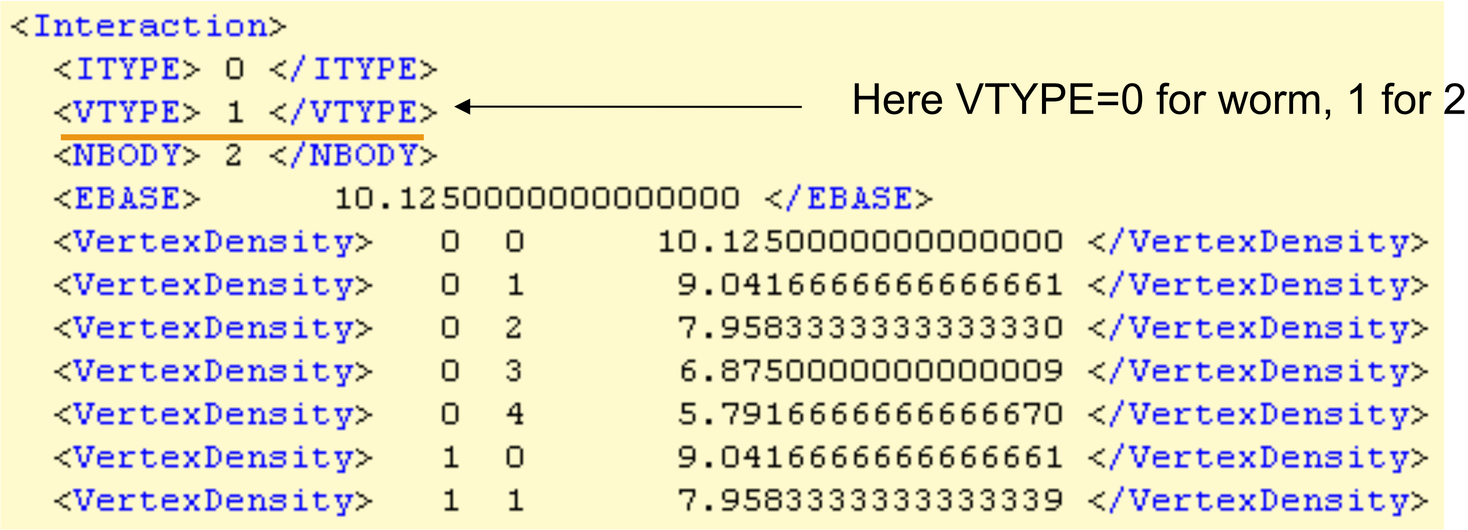
① final state (an intermediate state between the head and the tail.)

② Probability



About xml files ③＜algorithm.xml＞－2

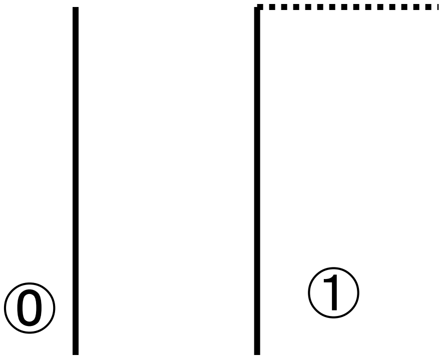
<Interaction> … </Interaction> gives probabilities, called a vertex density, of putting a vertex on each state.



```
<Interaction>
  <ITYPE> 0 </ITYPE>
  <VTYPE> 1 </VTYPE>
  <NBODY> 2 </NBODY>
  <EBASE>      10.1250000000000000 </EBASE>
  <VertexDensity>   0  0       10.1250000000000000 </VertexDensity>
  <VertexDensity>   0  1        9.0416666666666661 </VertexDensity>
  <VertexDensity>   0  2        7.9583333333333330 </VertexDensity>
  <VertexDensity>   0  3        6.8750000000000009 </VertexDensity>
  <VertexDensity>   0  4        5.7916666666666670 </VertexDensity>
  <VertexDensity>   1  0        9.0416666666666661 </VertexDensity>
  <VertexDensity>   1  1        7.9583333333333339 </VertexDensity>
```

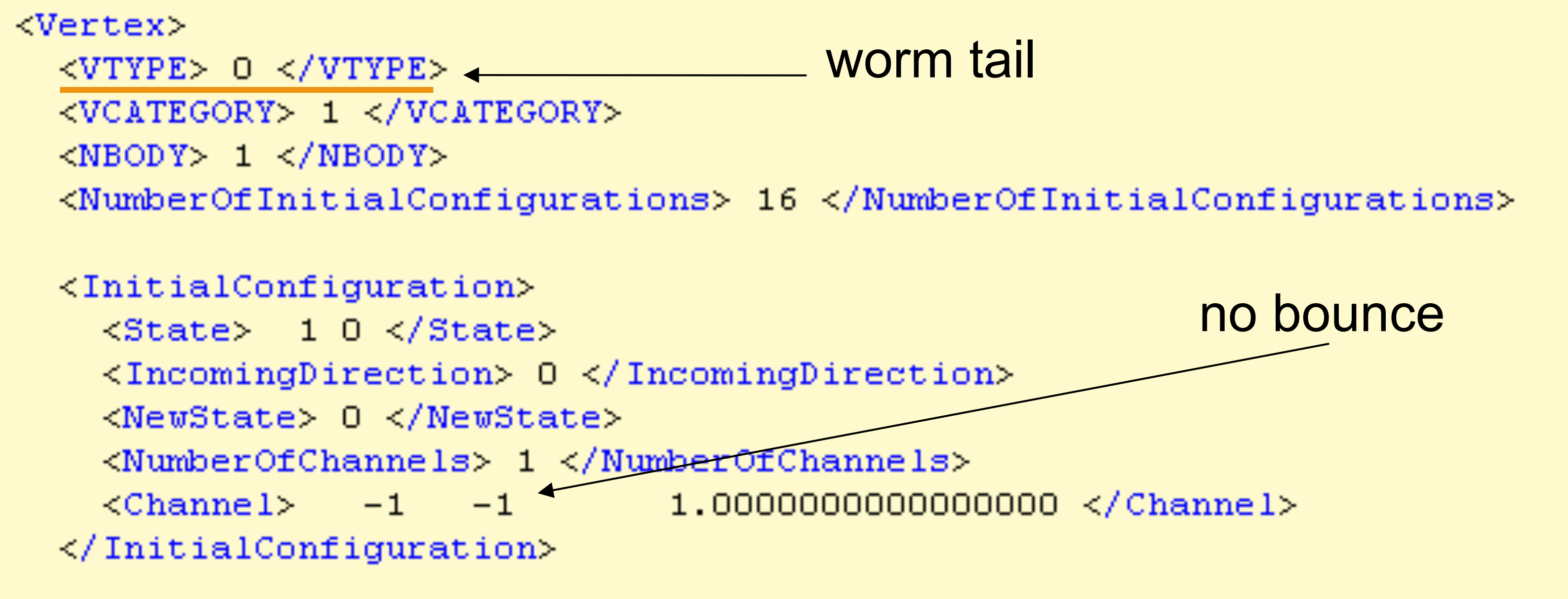
Here VTYPE=0 for worm, 1 for 2sites vertex

<ITYPE>, <NBODY> corresponds to that in hamiltonian.xml.
<VTYPE>
<EBASE> arbitrary energy-shift.
<VertexDensity>⓪ ① ②</ VertexDensity>
⓪ state of the left hand side of the segment.
① state of the right hand side of the segment.
② vertex density.



About xml files ③＜algorithm.xml＞－3

<Vertex> … </Vertex> gives scattering probabilities of a worm with each vertex labeled ‘’VTYPE’.



```
<Vertex>
  <VTYPE> 0 </VTYPE>
  <VCATEGORY> 1 </VCATEGORY>
  <NBODY> 1 </NBODY>
  <NumberOfInitialConfigurations> 16 </NumberOfInitialConfigurations>

  <InitialConfiguration>
    <State>  1 0 </State>
    <IncomingDirection> 0 </IncomingDirection>
    <NewState> 0 </NewState>
    <NumberOfChannels> 1 </NumberOfChannels>
    <Channel>   -1   -1       1.0000000000000000 </Channel>
  </InitialConfiguration>
```

worm tail

no bounce

<State> ⓪ ①</ State>

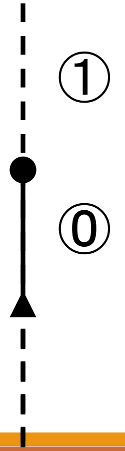
<incomeingDirection>:direction of the head (0 for up, 1 for down)

<Channel> ② ③ ④</ Channel>

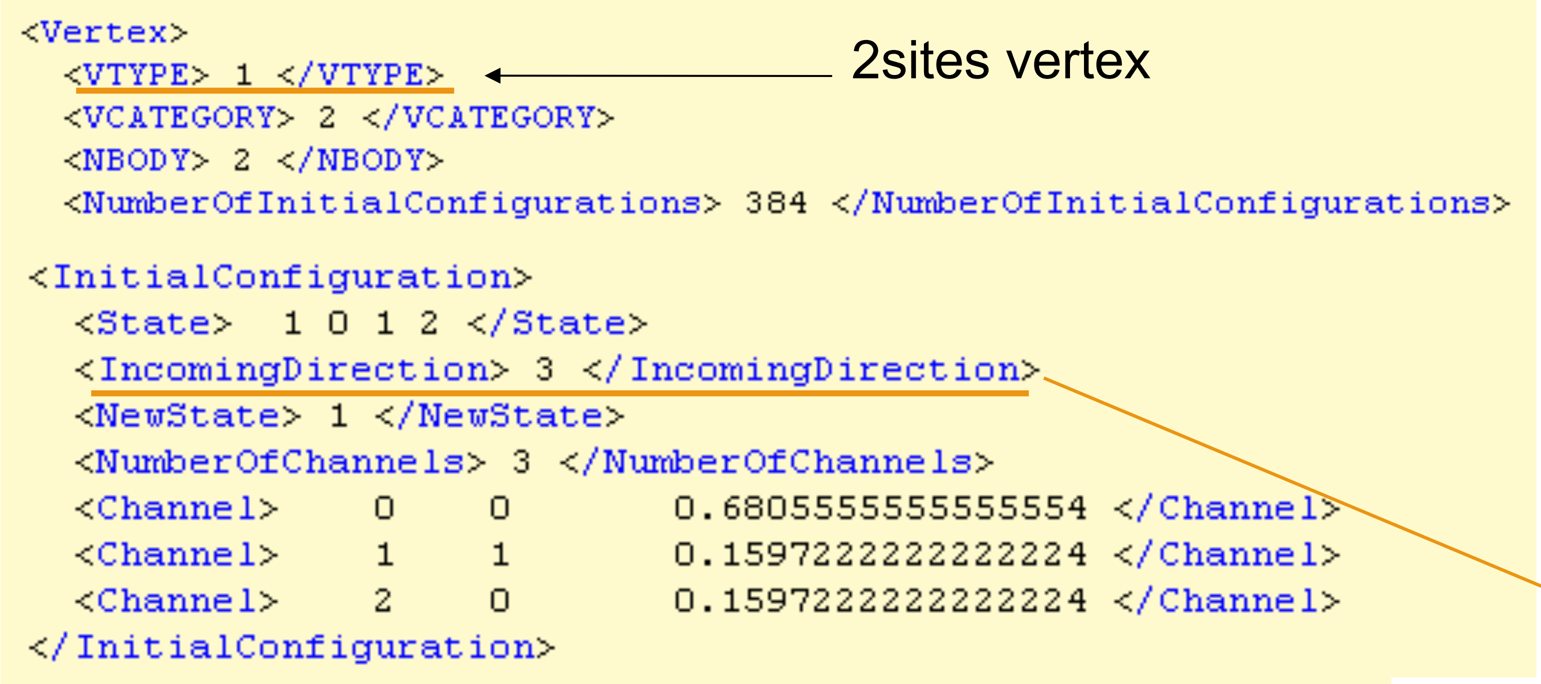
②: The state of the lower side of the worm.

③: The state of the upper side of the worm.

④: probability



About xml files ③＜algorithm.xml＞－4



```
<Vertex>
  <VTYPE> 1 </VTYPE>
  <VCATEGORY> 2 </VCATEGORY>
  <NBODY> 2 </NBODY>
  <NumberOfInitialConfigurations> 384 </NumberOfInitialConfigurations>

 <InitialConfiguration>
   <State>  1 0 1 2 </State>
   <IncomingDirection> 3 </IncomingDirection>
   <NewState> 1 </NewState>
   <NumberOfChannels> 3 </NumberOfChannels>
   <Channel>    0    0       0.6805555555555554 </Channel>
   <Channel>    1    1       0.1597222222222224 </Channel>
   <Channel>    2    0       0.1597222222222224 </Channel>
 </InitialConfiguration>
```

2sites vertex

<State>⓪ ① ② ③</ State>

<IncomingDirection>④</ IncomingDirection>

the incoming worm on this leg.

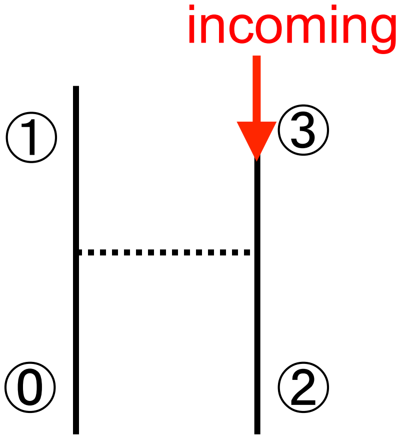
<NewState>:the state of ④ after update

<Channel> ⑤ ⑥ ⑦</ >

⑤ : a leg where worm outgoes after update

⑥ : the state of ⑤ after update

⑦ : probability



Appendix

main–source files of DSQSS:

dla → dla.cc (include: dla.hpp, algorithm.hpp, lattice.hpp, graphic.cc (with option –D), link.h, measure.hpp, objects.hpp, parameter.hpp),

random.cc (include: random.hpp)

sample of XML–generation files:

lattgene → lattgene.cc

:For hyper cubic lattice. 1~3D are supported.

hamgen_H → exact_H.cc (include : spin_H.h, canonical.h, matrix.h)

: it generates hamiltonian.xml which describes weights for Heisenberg model.

dla_alg → dla_alg.cc (include: dla_alg.h, io.h, xml.h, arry.h,name.h)

: it generates algorithm.xml which describes MC probabilities using hamiltonian.xml.

model-dependent files:

dla → measure_specific.cc, measure_specific.h
hamgen_H → exact_H.cc, spin_H.h
lattgene → lattgene.cc

About include files of dsqss

main:

dla.hpp → simulation core

parameter.hpp → read an input file

algorithm.hpp → read algorithm.xml

lattice.hpp → read lattice.xml

link.hpp → template for linked lists of vertices

graphic.cc (with option –D) → for graphical display of worldlines

measure.hpp → for measurement with including measure_specific.*

objects.hpp → DLA objects mean vertex, segment, worm etc…

sub:

spin_H.h, canonical.h, dla_alg.h, random.hpp

misc:

arry.h

io.h

matrix.h → for matrix operation.

xml.h → generate XML files.

name.h ß

Please see “Appendix” in wiki for further details.