DSQSS Documentation

リリース *2.0.0-Beta*

DSQSS developers

2019 年 04 月 18 日

Contents:

第 1 章

DSQSS とは？

1.1 概要

DSQSS は, 連続虚数時間向き付きループアルゴリズムに基づく量子モンテカルロ法によって離散空間上で定義された量子多体問題を解くためのプログラム群です. 単位格子の情報, 2 体相互作用ハミルトニアンの行列要素などを入力ファイルとして広い範囲のモデルに対応しています. 例えば, 次元, 格子の 1 辺の長さ, 相互作用の異方性, スピンの大きさ, 磁場の大きさ, 温度などのパラメータを任意にとって, XXZ ハイゼンベルクモデルの計算を行うことができます. また, ボーズ系のシミュレーションも可能です. 大規模並列化に対応したプログラム PMWA も含まれています.

DSQSS は, 以下のサブシステム群から構成されています.

- シリアル版 DSQSS/DLA
  1. シンプルモード入力ファイルジェネレータ: `dla_pre`
  2. ハミルトニアンファイルジェネレータ: `dla_hamgen`
  3. 格子ファイルジェネレータ: `dla_latgen`
  4. 波数ファイルジェネレータ: `dla_latgen`
  5. パラメータファイルジェネレータ: `dla_pgen`
  6. アルゴリズムジェネレータ: `dla_alg`
  7. 量子モンテカルロ法エンジン (向き付きループアルゴリズム)： `dla`
- 非自明並列版 DSQSS/PMWA
  1. 入力ファイルジェネレータ: `pmwa_pre`
  2. 格子定義ジェネレータ： `lattgene_P`

3. 量子モンテカルロ法エンジン (向き付きループアルゴリズム)： `pmwa_H` （XXZ 模型）, `pmwa_B` (ハードコアボソン系)

1.2 リリース情報

- 2018.10.19 dsqss Ver 1.2.0 CMake の導入, 入力ファイル生成支援スクリプト dsqss_pre.py の追加など
- 2015.11.20 dsqss-Ver.1.1.17+pmwa-Ver.1.1.2 GCC に対応, PMWA, configure 修正, その他/tool のスクリプトを追加
- 2014.03.28 Ver.1.1.16 シングルモード（非 MPI 環境）に対応
- 2013.07.22 Ver.1.1.15 レプリカ交換法の不具合の修正, マニュアルの一部修正
- 2013.01.10 Ver.1.1.14 メモリ増大の不具合の修正, マニュアルの一部修正
- 2012.10.03 Ver.1.1.13 サンプルの追加, その他微修正
- 2012.7.12 Ver.1.1.12 物性研システム B の MPI 環境に対応したマクロを生成
- 2012.6.7 Ver.1.1.11 doxygen, sphinx の自動生成 make の作成, その他微修正
- 2012.3.25 Ver.1.1.10 exact_H.cc の修正, 実行モジュール名 hamgen_H に変更
- 2012.3.23 Ver.1.1.9 runConfigure, dla.cc の不具合修正
- 2012.3.14 Ver.1.1.8 コード dla_alg.cc のコメント文修正
- 2012.2.29 Ver.1.1.6 バグ修正, マニュアルの整備追加
- 2011.9.30 Ver.1.1 レプリカ交換計算機能の追加
- 2011.9.25 Ver.1.0.20 温度の規格化, GNU 拡張の廃止等修正
- 2011.3.31 Ver.1.0

1.3 開発者

- 加藤康之（東京大学工学系研究科）
- 川島直輝（東京大学物性研究所）
- 坂倉耕太（NEC）
- 鈴木隆史（兵庫県立大学工学研究科）
- 原田健自（京都大学情報学研究科）

- 正木晶子（日立研究所）
- 本山裕一（東京大学物性研究所）
- 吉見一慶（東京大学物性研究所）

2018/10/19 現在

1.4 協力者

- 大久保毅 (東京大学理学系研究科)
- 加藤岳生 (東京大学物性研究所)

2018/10/19 現在

1.5 ライセンス

- GNU General Public License (GPL) に基づきます.
- 利用のための必須条件ではありませんが, 利用実態を把握したいので, 科学計算などに使用した場合, 関連公表論文の書誌情報などをアプリケーション管理者までお知らせ下さることを希望します. また, 論文などによる成果公開に際して謝辞に記載していただければ幸いです.

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.

1.6 謝辞

本ソフトウェアの研究開発は, 一部, 次世代スパコンプロジェクト「次世代ナノ統合シミュレーションソフトウェア研究開発」および 2018 年度東京大学物性研究所ソフトウェア開発・高度化プロジェクトの援助によって行われました. ここに感謝の意を記します.

1.7 連絡・問い合わせ先

GitHub ページの issue もしくは DSQSS 開発者用メーリングリスト dsqss-dev@issp.u-tokyo.ac.jp にご連絡ください.

第 2 章

インストール

2.1 必要なライブラリ

DSQSS の使用には以下のプログラム・ライブラリが必要です.

- (Optional) MPI (PMWA を使用する場合には必須)
- python 2.7 or 3.x
  - numpy
  - scipy
  - toml
  - pip (make install する場合には必須)

2.2 ダウンロード

- zip ファイルをダウンロードする場合

  DSQSS の最新版は https://github.com/issp-center-dev/dsqss/releases からダウンロードできます.

- git を利用する場合

  Git を利用されている方は, 端末から以下のコマンドを打つことで直接ダウンロードできます.

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

2.3 フォルダ構成

DSQSS のダウンロード後に zip ファイルを解凍すると, ファイルが展開されます (git を利用された方は, clone を行ったファイル直下のフォルダ構成になります). 以下, 重要なファイル・フォルダについてその構成を記載します.

```
├── CMakeLists.txt
├── LICENSE
├── README.md
├── config
│ ^^c2^^a0^^c2^^a0 ├── gcc.cmake
│ ^^c2^^a0^^c2^^a0 └── intel.cmake
├── doc
│ ^^c2^^a0^^c2^^a0 ├── en
│ ^^c2^^a0^^c2^^a0 └── jp
├── sample
│ ^^c2^^a0^^c2^^a0 ├── CMakeLists.txt
│ ^^c2^^a0^^c2^^a0 ├── dla
│ ^^c2^^a0^^c2^^a0 └── pmwa
├── src
│ ^^c2^^a0^^c2^^a0 ├── common
│ ^^c2^^a0^^c2^^a0 ├── dla
│ ^^c2^^a0^^c2^^a0 ├── pmwa
│ ^^c2^^a0^^c2^^a0 └── third-party
├── test
│ ^^c2^^a0^^c2^^a0 ├── CMakeLists.txt
│ ^^c2^^a0^^c2^^a0 ├── dla
│ ^^c2^^a0^^c2^^a0 ├── pmwa
│ ^^c2^^a0^^c2^^a0 └── tool
└── tool
    ├── CMakeLists.txt
    ├── cmake
    ├── dsqss
    └── setup.py
```

2.4 インストール

インストールは以下の手順で行うことが出来ます. 以下, ダウンロードしたファイルの直下にいることを想定しています.

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

`/path/to/install/to` をインストールしたい先のパスに設定してください. 指定しなかった場合のデフォルト値は `/usr/local` です. なお, cmake がうまくいかない場合にも, コンパイラを直接指定するとうまくいくこと

があります. 詳細については https://github.com/issp-center-dev/HPhi/wiki/FAQ をご覧ください.

これにより各実行ファイルが dsqss.build/src ディレクトリ以下に, 入力ファイルの生成ツールが dsqss.build/tool ディレクトリ以下に作成されます. 次に作成された実行ファイルが正常に動作するかテストするため, 以下のコマンドを打ちます.

```
$ make test
```

テストが 100% 通過したことが確認できた後, 以下のコマンドを入力しインストールします.

```
$ make install
```

実行バイナリが先に指定したインストールパスにある bin ディレクトリに, サンプルが share/dsqss/VERSION/samples にインストールされます. また, 補助ツールを含めた DSQSS の実行に必要な環境変数を設定するためのファイルが share/dsqss/dsqssvar-VERSION.sh に生成されます. このファイルを source コマンドで読み込んでください.

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

usage

第 3 章

DLA のチュートリアル

3.1 DSQSS/DLA とは

DSQSS/DLA は世界線モンテカルロ法の向き付きループアルゴリズムを実装したプログラムです. 負符号問題の現れない限り, 任意の模型・任意の格子で磁化や帯磁率などの計算を行えます. ハイゼンベルグスピン模型やボーズ・ハバード模型を表す入力ファイルを作るツールや超立方格子・三角格子を表す入力ファイルを作るツールが付属しています.

この章では, 環境変数 `$DSQSS_ROOT` で示されるディレクトリに DSQSS がインストールされていると仮定します.

3.2 DSQSS/DLA による反強磁性ハイゼンベルグダイマーのエネルギー計算

このチュートリアルでは, $S = 1/2$ 反強磁性ハイゼンベルグダイマー $\mathcal{H} = J\vec{S}_1 \cdot \vec{S}_2$ の基底状態エネルギー計算をすることで, DSQSS/DLA の使い方を学びます.

DSQSS/DLA による計算は,

1. 入力ファイルの準備
2. 計算の実行
3. 計算結果の解釈

の 3 段階に分かれます.

3.2.1 入力ファイルの準備

DSQSS/DLA を実行するには,

1. パラメータファイル

2. 格子定義ファイル

3. アルゴリズム定義ファイル

の 3 つの入力ファイルが必要です. そのため, まずはこれらの入力ファイルを作成します. そのためのユーティリティツールが dla_pre.py です. これは単一の入力ファイルから, DSQSS/DLA の入力ファイルを生成する Python スクリプトです. まず, dla_pre.py の入力ファイルとして, 次の内容を持つテキストファイル std.toml を準備します (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
```

このファイルを dla_pre.py に与えます.

```
$ $DSQSS_ROOT/bin/dla_pre.py std.toml
```

この結果, パラメータファイル param.in, 格子定義ファイル lattice.xml, アルゴリズム定義ファイル algorithm.xml が生成されます。

3.2.2 計算の実行

入力ファイルを作成したら, DSQSS/DLA による計算を実行します.

```
$ $DSQSS_ROOT/bin/dla param.in
```

なお, 計算を実行するときに MPI を用いることで, 乱数並列計算が可能です.

```
$ mpiexec -np 4 $DSQSS_ROOT/bin/dla param.in
```

並列数 (今回は 4) だけ独立に計算を行い, モンテカルロサンプル数を増やすことで計算精度を向上できます.[*1]

3.2.3 計算結果の解釈

計算結果は出力ファイル sample.log に書き出されます. サイトあたりのエネルギーは `ene` という名前で記されており, たとえば `grep` コマンドで

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

と取得できます. 2 つある数字はそれぞれ期待値と統計誤差です. 反強磁性ハイゼンベルグダイマーの基底状態でのサイトあたりエネルギーは $-3|J|/8 = -0.375|J|$ なので, 問題なく計算されていることがわかります.

3.3 DSQSS/DLA によるスピン鎖の帯磁率計算

このチュートリアルでは, 局在スピンの大きさが $S = 1/2, 1$ である 2 種類の反強磁性ハイゼンベルグ鎖 ($J = -1, L = 30$) において, 温度 T を 0.05 から 2.0 まで動かしたときの帯磁率変化を計算します.

次に示すのは, 入力ファイルの生成と本計算を自動で行い, 帯磁率の温度依存性をファイルに書き出す Python スクリプトです (sample/dla/02_spinchain/exec.py).

```
import subprocess

from dsqss.parameter 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)
```

(次のページに続く)

[*1] macOS 上の OpenMPI を使う場合, プログラム終了時にエラーメッセージが出ることがあります (`No such file or directory (errno 2)`). DSQSS/DLA の実行自体に影響はありませんが, これを抑制したい場合は, `--mca shmem posix` オプションを `mpiexec` に付与してください.

(前のページからの続き)

```
        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()
```

必要なパスを設定するために, dsqssvars-VERSION.sh を読み込んでから実行してください (VERSION は DSQSS のバージョン番号, 例えば 2.0.0 に読み替えてください).

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

$S = 1/2$ の結果が xmzu_1.dat に, $S = 1$ の結果が xmzu_2.dat にそれぞれ書き出されます (図 3.1). スピンの大きさによって, スピンギャップの有無が異なり, その結果として絶対零度付近での帯磁率の振る舞いが異なってくることがわかります.

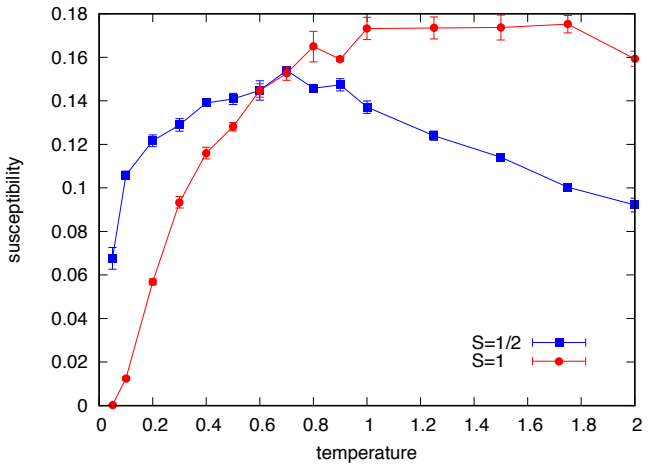

図 3.1　反強磁性ハイゼンベルグスピン鎖の帯磁率の温度依存性. 青が $S = 1/2$ で 赤が $S = 1$ の結果.

3.4 DSQSS/DLA による正方格子上ハードコアボソン系の粒子数計算

このチュートリアルでは, 大きさ 8×8 の正方格子上の, $V/t = 3, \beta t = 10$ というパラメータを持つハードコアボーズハバード模型について, 化学ポテンシャル μ を -4.0 から 14.0 まで動かして計算を行い, 粒子数密度の変化を計算します.

次に示すのは, 入力ファイルの生成と本計算を自動で行い, 粒子数密度の化学ポテンシャル依存性をファイルに書き出すスクリプトです (sample/dla/03_bosesquare/exec.py).

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

必要なパスを設定するために, `dsqssvars-VERSION.sh` を読み込んでから実行してください (`VERSION` は DSQSS のバージョン番号, 例えば `2.0.0` に読み替えてください).

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

結果は `amzu.dat` に書き出されます (図 3.2). $\mu = 6$ 付近では密度プラトーが観測されます. ここでは近接サイト間の反発によりチェッカーボード固体相になっています.

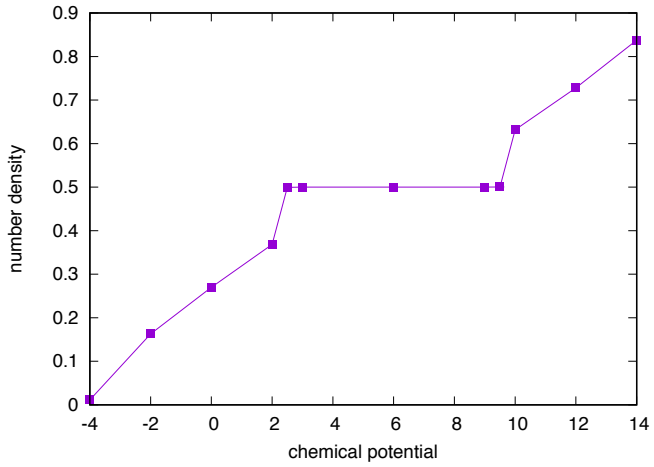

図 3.2　正方格子上のボースハバード模型における密度の化学ポテンシャル依存性.

第 4 章

DSQSS/DLA のユーザーマニュアル

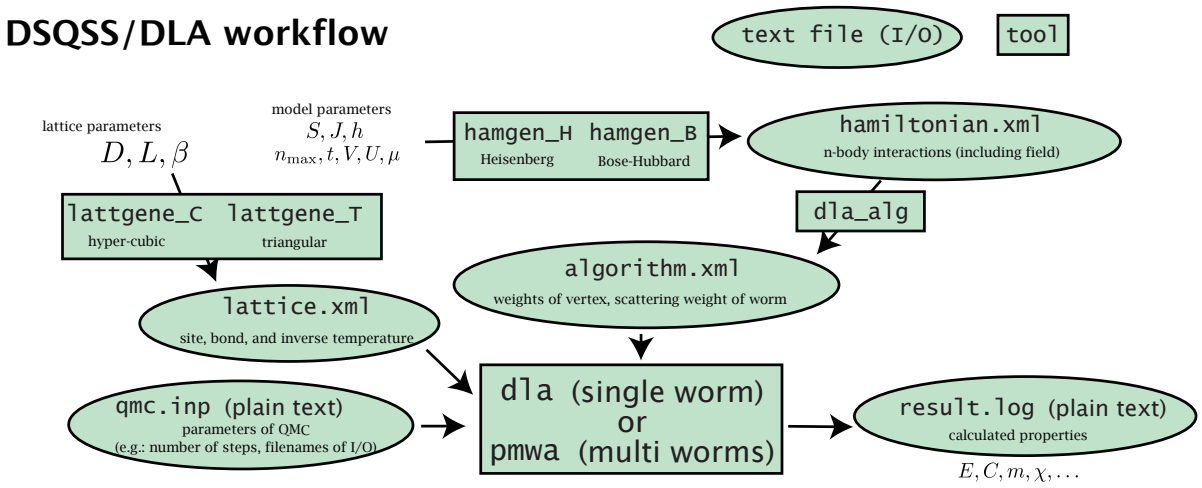

図 4.1　DSQSS/DLA のワークフロー

4.1 DLA のシンプルモードファイル

4.1.1 シンプルモードファイル

シンプルモードファイルは TOML 形式のテキストファイルです。`dla_pre` や `dla_hamgen` などのツールの入力ファイルとして使います。

`parameter`

逆温度やモンテカルロステップ数などの計算条件を指定するテーブルです。`dla_pre` で使用されます。

`parameter` テーブルに属するキーのリストと意味を次に示します。

パラメータ名	型	デフォルト値	説明
beta	double	–	逆温度.
npre	int	1000	モンテカルロスイープ中のワーム対生成回数を求める事前計算のためのモンテカルロ試行回数.
ntherm	int	1000	熱平衡化のためのモンテカルロスイープ数.
ndecor	int	1000	セット間の自己相関を取り除くためのモンテカルロスイープ数.
nmcs	int	1000	物理量計算のためのモンテカルロスイープ数.
nset	int	10	モンテカルロ計算の繰り返し数.
simulationtime	int	0.0	計算時間（単位は秒）.
seed	int	198212240	疑似乱数の種.
nvermax	int	10000	最大バーテックス数.
nsegmax	int	10000	最大セグメント数.
algfile	int	algorithm.xml	アルゴリズム定義ファイル名.
latfile	string	lattice.xml	格子定義ファイル名.
ntau	int	10	虚時間構造因子などの計算で使われる虚時間方向の離散化数.
wvfile	string	–	波数ベクトル XML ファイル名. 空文字列の場合, 構造因子は計算されない.
dispfile	string	–	相対座標 XML ファイル名. 空文字列の場合, 実空間表示温度グリーン関数は計算されない.
outfile	string	sample.log	メイン出力ファイル名.
sfoutfile	string	sf.dat	構造因子出力ファイル名.
cfoutfile	string	cf.dat	実空間表示温度グリーン関数出力ファイル名.
ckoutfile	string	ck.dat	波数空間表示温度グリーン関数出力ファイル名.

- simulationtime について
  - simulationtime > 0.0 のとき
    - 指定秒数が経過するか, 計算が完了したとき, 途中経過をチェックポイントファイルに書き出した後, プログラムを終了します.
    - 計算開始時にチェックポイントファイルがある場合, そのファイルを読み込んだ後に計算を再開します.
    - チェックポイントファイルの名前は outfile で指定されるメイン出力ファイル名の末尾に .cjob をつけたものです.
  - simulationtime <= 0.0 のとき
    - チェックポイントファイルは無視され, 書き出しも読み込みも行われません.

lattice

格子の情報を指定するテーブルです. dla_pre, dla_latgen で使用されます.

lattice テーブルに属するキーのリストと意味を次に示します.

パラメータ名	型	デフォルト値	説明
lattice	string	–	格子の種類.
dim	int	–	空間次元.
L	list(int) or int	–	格子の大きさ. 整数の配列か整数で指定します. 空間次元より要素が少ない場合, 足りない要素は最後の要素で自動的に埋められます.
bc	list(bool) or bool	true	各次元に対する、格子の境界条件. ブール値の配列かブール値で指定します. true が周期的境界条件を, false が開境界条件を示します.

現在利用可能な格子は次のとおりです.

hypercubic　d 次元超立方格子.

triangular　2 次元三角格子.

honeycomb　2 次元蜂の巣格子.

kagome　2 次元カゴメ格子.

hamiltonian

ハミルトニアンの情報を指定するテーブルです。dla_pre, dla_hamgen で使用されます。

hamiltonian テーブルに属するキーのリストと意味を次に示します。

パラメータ名	型	デフォルト値	説明
model	string	–	模型の種類. XXZ 模型 'spin' とボーズハバード模型 'boson' が利用可能です.
M	int	1	サイトあたりの取りうる状態数-1. XXZ 模型では局所スピンの大きさ $2S$ を, ボーズハバード模型では粒子数カットオフを指定します.

XXZ 模型

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

に特有のパラメータは次の通り.

パラメータ名	型	デフォルト値	説明
Jz	list(float) or float	0.0	交換相互作用. 相互作用の種類が複数ある場合は, 配列で指定する. 正が強磁性的相互作用を, 負が反強磁性的相互作用を意味する.
Jxy	list(float) or float	0.0	交換相互作用. 相互作用の種類が複数ある場合は, 配列で指定する. 正が強磁性的相互作用を, 負が反強磁性的相互作用を意味する.
D	list(float) or float	0.0	オンサイトのスピン異方性パラメータ. サイトの種類が複数ある場合は, 配列で指定する.
h	list(float) or float	0.0	磁場. サイトの種類が複数ある場合は, 配列で指定する.

ボーズハバード 模型

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

に特有のパラメータは次の通り.

パラメータ名	型	デフォルト値	説明
t	list(float) or float	0.0	ホッピングパラメータ. 相互作用の種類が複数ある場合は, 配列で指定する.
V	list(float) or float	0.0	オフサイトの粒子間相互作用. 相互作用の種類が複数ある場合は, 配列で指定する. 正が斥力ポテンシャル, 負が引力ポテンシャルを意味する.
U	list(float) or float	0.0	オンサイトの粒子間相互作用. サイトの種類が複数ある場合は, 配列で指定する. 正が斥力ポテンシャル, 負が引力ポテンシャルを意味する.
mu	list(float) or float	0.0	化学ポテンシャル. サイトの種類が複数ある場合は, 配列で指定する.

kpoints

波数の情報を指定するテーブルです。`dla_pre` および `dla_wvgen` で使用されます.

パラメータ名	型	デフォルト値	説明
ksteps	list(int) or int	0	波数の増分. 0 の場合、格子サイズの半分が設定される.

algorithm

ワームの散乱確率の計算アルゴリズムなどを指定するテーブルです. dla_pre で使用されます.

パラメータ名	型	デフォルト値	説明
kernel	string	'suwa todo'	バーテックスにおけるワームの散乱過程について、その遷移確率を計算するために用いる手法。

kernel として指定できる手法は次の通り。

suwa todo　詳細釣り合いを破る諏訪・藤堂アルゴリズム (H. Suwa and S. Todo, PRL 105, 120603 (2010).)

reversible suwa todo　詳細釣り合いを満たす諏訪・藤堂アルゴリズム (H. Suwa and S. Todo, arXiv:1106.3562.)

heat bath　熱浴法.

metropolice　メトロポリスアルゴリズム.

4.2 DLA のスタンダードモード入力ファイル

4.2.1 入力ファイル一覧

lattice.dat	格子ファイル.
lattice.toml	格子ファイル.
hamiltonian.toml	ハミルトニアンファイル.
kpoints.dat	波数ファイル.

4.2.2 格子データファイル

格子データファイルは空間の情報, たとえばサイトの数やサイト同士のつながりかたなどを定義するためのテキストファイルです. dla_alg の入力ファイルとして扱われます。

から行末まではコメントとして読み飛ばされます。また、空白行も飛ばされます。name, lattice, directions, sites, interactions の 5 つのセクションから構成されます。

name　格子の名前です。 1 行のテキストからなります。 コメントや識別用であり、実際の計算では利用されません。

lattice　格子の情報を指定するセクションです。

- **1 行目**
  - 格子の次元を表す整数。
- **2 行目**
  - 格子のサイズを表す、空白区切りで次元の数だけ並べられた整数の組。
- **3 行目**
  - 格子の境界条件を表す、空白区切りで次元の数だけ並べられた整数の組。1 で周期的境界条件を、 0 で開放端境界条件を示します。
- **4 行目以降**
  - 空間の基底ベクトル $\vec{e^i}$ を表す、空白区切りで次元の数 +1 だけ並べられた数の組。最初の整数はベクトルの番号 i を、残りの数はベクトルのデカルト座標 e^i_d を示します。

directions　ボンド（二体相互作用）の向きを指定するセクションです。

- **1 行目**
  - ボンドの向きの総数。
- **2 行目以降**
  - 各向きを表すベクトルを表す、空白区切りで次元の数 +1 だけ並べられた数の組。最初の整数は向きの番号を、残りの数はベクトルの要素を示します。lattice で指定されたベクトルを基底とします。

sites　サイトの情報を指定するセクションです。

- **1 行目**
  - 格子に含まれるサイトの総数を表す整数。
- **2 行目以降**
  - 各サイトの情報を表す、空白区切りで次元の数 +2 だけ並べられた数の組。最初の 2 つはそれぞれ「サイト番号」「サイトタイプ」を意味します. 3 つ目以降は lattice で指定されたベクトルを基底とするサイトの座標です.

interactions　相互作用の空間情報を指定するセクションです。

- **1 行目**
  - 格子に含まれる多体相互作用の総数を表す整数。
- **2 行目以降**
  - $N =$ 関与サイト数 $+ 5$ 個の整数の組. 意味は次表の通り.

列	説明
1	相互作用番号
2	相互作用タイプ
3	関与サイト数
4 ... (N-2)	サイト番号
N-1	ボンドが周期境界をまたぐ場合には 1, そうでなければ 0
N	ボンドの方向番号

二次元正方格子の例を示します。

```
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
5 0 1 1
6 0 2 1
7 0 3 1
8 0 0 2
9 0 1 2
```

(次のページに続く)

(前のページからの続き)

```
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 格子 TOML ファイル

格子 TOML ファイルはユニットセルと基本並進ベクトルを用いて空間の情報を定義するための, TOML 形式のファイルです. dla_alg の入力ファイルとして扱われます。

parameter と unitcell のふたつのテーブルから構成されます。

parameter　格子の情報を記述するテーブルです。

parameter.name　格子の名前です。実際の計算には用いられません。

parameter.L　格子の大きさを表す整数配列です。

parameter.bc　格子の境界条件を表すブーリアンの配列です。true が周期的境界条件を、 false が開放端境界条件を示します。

parameter.basis　格子の基本並進ベクトルを表す 2 次元配列（配列の配列）です。

unitcell　ユニットセルの情報を記述するテーブルです。

unitcell.sites　ユニットセル内のサイトを示すテーブルの配列です。

unitcell.sites.siteid　サイトのユニットセル内での識別番号です。

unitcell.sites.type　サイトタイプを示す整数です。

unitcell.sites.coord　ユニットセル内におけるサイトの局所座標を示す配列です。

unitcell.bonds　ユニットセル内のボンドを示すテーブルの配列です。

unitcell.bonds.bondid　ボンドのユニットセル内での識別番号です。

unitcell.bonds.type　ボンドタイプを示す整数です。

unitcell.bonds.source　ボンドの始点サイトの情報を表すテーブルです。

unitcell.bonds.source.siteid　サイトのユニットセル内での識別番号です。

unitcell.bonds.target　ボンドの終点サイトの情報を表すテーブルです。

unitcell.bonds.target.siteid　サイトのユニットセル内での識別番号です。

unitcell.bonds.target.offset　始点サイトの属するユニットセルから見た、終点サイトの属するユニットセルの相対座標です。

二次元正方格子の例を示します.

```
[parameter]
name = "square lattice"
dim = 2
L = [4,4]
```

(次のページに続く)

(前のページからの続き)

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

4.2.4 ハミルトニアン TOML ファイル

ハミルトニアン TOML ファイルは局所ハミルトニアン, 例えばボンドハミルトニアン, を指定する, TOML 形式で記述されるテキストファイルです. `dla_alg` の入力として, アルゴリズム定義ファイルを作成するために用いる補助入力ファイルとなっています. ハイゼンベルグ模型などのよく用いられる模型については, 補助ツール `dla_hamgen` が用意されています.

`name`　ハミルトニアンの名前です。シミュレーション中で使われることはありません。

`sites`　サイトハミルトニアンの情報を記述するテーブルの配列です。

- `sites.id`　サイトタイプを示す整数です。
- `sites.N`　局所自由度が取りうる状態の数を示す整数です。例えば $S = 1/2$ スピンでは 2 です。
- `sites.values`　局所自由度の基底演算子の対角要素. 例えば $S = 1/2$ スピンでは `[-0.5, 0.5]`.
- `sites.elements`　サイトハミルトニアンの行列要素を示すテーブルの配列です。
  - `sites.elements.istate`　ハミルトニアンが作用する前の状態番号です。
  - `sites.elements.fstate`　ハミルトニアンが作用した後の状態番号です。
  - `sites.elements.value`　ハミルトニアンの行列要素の値です。
- `sites.sources`　ワームを導入するためのソースハミルトニアンの行列要素を示すテーブルの配列です。

sites.sources.istate ハミルトニアンが作用する前の状態番号です。

sites.sources.fstate ハミルトニアンが作用した後の状態番号です。

sites.sources.value ハミルトニアンの行列要素の値です。

interactions 多体相互作用の情報を記述するテーブルの配列です。

interactions.id 相互作用タイプを示す整数です。

interactions.nbody 相互作用に関与するサイトの数を示す整数です。

interactions.N 相互作用に関与するサイトそれぞれで局所自由度が取りうる状態の数です。

interactions.elements 相互作用ハミルトニアンの行列要素を記述するテーブルの配列です。

interactions.elements.istate 相互作用ハミルトニアンが作用する前のサイトの状態を指定する整数の配列です。

interactions.elements.fstate 相互作用ハミルトニアンが作用した後のサイトの状態を指定する整数の配列です。

interactions.elements.value 相互作用ハミルトニアンの行列要素の値です。

$S = 1/2$ 反強磁性ハイゼンベルグ模型の例を示します。

```
name = "S=1/2 XXZ model"
[[sites]]
id = 0
N = 2
[[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
```

(次のページに続く)

(前のページからの続き)

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

4.2.5 波数ファイル

波数ファイルは, 波数ベクトル

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

の $\vec{n}^{(i)}$ を指定するテキストファイルです。

`dim`　格子の次元を示す整数です。

`kpoints`　波数ベクトルを指定するセクションです。

- **1 行目**
  - 波数ベクトルの総数。
- **2 行目以降**
  - 波数ベクトルを表す、空白区切りで次元の数 +1 だけ並べられた数の組。最初の整数は波数ベクトルの番号を、残りの数はベクトルの要素 n_d を示します。

ベクトルの基底は逆格子ベクトルです。`lattice.dat` などで座標が `a_d` と指定されるような格子点 r と、`kpoints.dat` で `n_d` で指定されるような波数 k との内積は、:math:‘‘

正確には、格子の座標が $\vec{r} = \sum r_d \vec{e}_d$ で表現されて、波数が $\vec{k} = \sum k_d \vec{g}_d$ で表現されているとき、これらの内積は $\vec{r} \cdot \vec{k} = \sum_d 2\pi r_d k_d / L_d$ となります。ここで L_d は d 番目の次元における格子のサイズです。

二次元の例を示します.

```
dim
2

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

4.3 DLA のエキスパートモード入力ファイル

4.3.1 入力ファイル一覧

qmc.inp	モンテカルロの繰り返し回数など, 計算制御のためのパラメータファイル.
lattice.xml	格子の定義ファイル.
algorithm.xml	アルゴリズム（ワームの散乱確率など）の定義ファイル.
wavevetor.xml	波数定義ファイル.
disp.xml	相対座標定義ファイル.

4.3.2 パラメータファイル

パラメータファイルは次に示すような書式のプレーンテキストファイルです.

- 1 行あたり 1 パラメータを, `<name> = <value>` という形式で表します.
- ファイル名以外は大文字小文字を区別しません.

- 空行, 空白は無視されます.
- "#" から行末まではコメントとして無視されます.

パラメータのリストと意味を以下に示します.

パラメータ名	型	デフォルト値	説明
beta	double	–	逆温度.
npre	int	1000	モンテカルロスイープ中のワーム対生成回数を求める事前計算のためのモンテカルロ試行回数.
ntherm	int	1000	熱平衡化のためのモンテカルロスイープ数.
ndecor	int	1000	セット間の自己相関を取り除くためのモンテカルロスイープ数.
nmcs	int	1000	物理量計算のためのモンテカルロスイープ数.
nset	int	10	モンテカルロ計算の繰り返し数.
simulationtime	int	0.0	計算時間（単位は秒）.
seed	int	198212240	疑似乱数の種.
nvermax	int	10000	最大バーテックス数.
nsegmax	int	10000	最大セグメント数.
algfile	int	algorithm.xml	アルゴリズム定義ファイル名.
latfile	string	lattice.xml	格子定義ファイル名.
ntau	int	10	虚時間構造因子などの計算で使われる虚時間方向の離散化数.
wvfile	string	–	波数ベクトル XML ファイル名. 空文字列の場合, 構造因子は計算されない.
dispfile	string	–	相対座標 XML ファイル名. 空文字列の場合, 実空間表示温度グリーン関数は計算されない.
outfile	string	sample.log	メイン出力ファイル名.
sfoutfile	string	sf.dat	構造因子出力ファイル名.
cfoutfile	string	cf.dat	実空間表示温度グリーン関数出力ファイル名.
ckoutfile	string	ck.dat	波数空間表示温度グリーン関数出力ファイル名.

- simulationtime について
  - simulationtime > 0.0 のとき
    - 指定秒数が経過するか, 計算が完了したとき, 途中経過をチェックポイントファイルに書き出した後, プログラムを終了します.
    - 計算開始時にチェックポイントファイルがある場合, そのファイルを読み込んだ後に計算を再開します.
    - チェックポイントファイルの名前は outfile で指定されるメイン出力ファイル名の末尾に .cjob をつ

けたものです.

- simulationtime <= 0.0 のとき
  - チェックポイントファイルは無視され, 書き出しも読み込みも行われません.

4.3.3 格子 XML ファイル

格子ファイルは空間の情報, たとえばサイトの数やサイト同士のつながりかたなどを定義するための, XML ライクな形式で記述されるテキストファイルです. これは一般に複雑になりますが, より単純な格子データファイルや格子 TOML ファイルから `dla_alg` を用いて生成することができます。

格子ファイルはただ一つの要素 Lattice を持ち, すべての情報は Lattice 要素の内容として含まれます.

Lattice　ファイル全体の要素. ほかのすべての要素は Lattice のサブ要素です.

Lattice/Comment　省略可能. コメント文を示し, 計算には使用されません.

Lattice/Dimension　格子の次元.

Lattice/LinearSize　ユニットセルを単位とした, 各次元の格子の長さ. 内容として, スペース区切りの正整数を Lattice/Dimension で指定した数だけ並べたものをとります.

```
<LinearSize> 3 4 </LinearSize>
# ユニットセルが第 1 次元方向に 3 個， 第 2 次元方向に 4 個並んでいる場合
```

Lattice/NumberOfSites　サイトの総数. ユニットセルの総数と 1 セル内のサイト数の積.

Lattice/NumberOfInteractions　相互作用項の総数. 二体相互作用のみの場合は, いわゆる「ボンド数」.

Lattice/NumberOfSiteTypes　サイトの種類数.

Lattice/NumberOfInteractionTypes　相互作用の種類数.

Lattice/NumberOfBondDirections　ボンドの方向 Direction の数.

Lattice/NumberOfEdgeInteractions　格子の周期的境界をまたぐボンドの総数.

Lattice/Basis　格子の空間座標を記述する基底ベクトル.

Lattice/S　サイト情報. Lattice/NumberOfSites で指定したサイト数だけ指定する必要があります. 内容として,「サイト番号」と「サイトタイプ」の 2 つの整数をスペース区切りで持ちます. サイトタイプの詳細は別途アルゴリズム定義ファイルの中で定義します.

```
<S> 3 0 </S>
# サイト番号が 3 のサイトはサイトタイプが 0 である .
```

Lattice/I　相互作用情報. Lattice/NumberOfInteractions で指定した相互作用数だけ指定する必要があります. 内容として,「相互作用番号」,「相互作用タイプ」,「相互作用サイト数」,「相互作用サイト番号」を指定するために, 相互作用サイト数 +3 個の整数をスペース区切りで持ちます. 相互作用タイプの詳細 — たとえば相互作用の大きさ — は別途アルゴリズム定義ファイルの中で定義します. サイト番号の順序は, アルゴリズム定義ファイルの Algorithm/Vertex/InitialConfiguration 要素で用いられるサイトの並び順と整合させる必要があります.

```
<I> 5 1 2 8 12 </I>
# 相互作用番号が 5 である相互作用は相互作用タイプが 1 で， 2 つのサイトが関与し，
# それらのサイト番号は 8 と 12 である .
```

Lattice/Direction　ボンドの方向. Lattice/NumberOfBondDirections の値だけ指定する必要があります. 内容として,「方向のインデックス」と「方向ベクトルの座標」をスペース区切りで指定します.

4.3.4 アルゴリズム定義ファイル

アルゴリズム定義ファイルは相互作用ごとのワームの散乱確率などを定義する, XML ライクな形式で記述されるテキストファイルです. これは一般に複雑になりえるので, より簡単なハミルトニアン定義ファイルから自動生成するためのツール dla_alg が用意されています.

アルゴリズム定義ファイルはただ一つの要素 Algorithm を持ち, すべての情報は Algorithm 要素の内容として含まれます.

Algorithm　ファイル全体の要素名. サブ要素として, General, Site, Interaction, Vertex があります. ワームの生成・消滅・散乱の仕方を定義します.

Algorithm/Comment　省略可能. コメント文を示し, 計算には使用されません.

Algorithm/General　サブ要素として, NSType, NIType, NVType, NXMax, WDiag があります. サイトの種類数や相互作用の種類数など, アルゴリズム定義の基本パラメータを設定します.

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

Algorithm/General/NSType　異なるサイト型の個数を指定する整数値.

Algorithm/General/NIType　異なる相互作用型の個数を指定する整数値.

Algorithm/General/NVType　異なるバーテックス型の個数を指定する整数値.

Algorithm/General/NXMax　各サイトが取りうる状態の数の最大値. 例えば大きさ S のスピン系ならば $2S+1$.

Algorithm/General/WDiag　ワームの行程長から相関関数を求めるときの比例係数.

Algorithm/Site　1 つのサイト型を定義します. 具体的には, そのサイト型をもつサイトに対する操作を定義します. サイトにワームを生成消滅する過程もここで定義します. サブ要素として, SType, NumberOfStates, LocalStates, VertexTypeOfSource, InitialConfiguration があります.

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

Algorithm/Site/SType　定義されるサイト型の識別番号.

Algorithm/Site/NumberOfStates　サイトが取りうる状態の数.

Algorithm/Site/LocalStates　状態のインデックスに対応する局所変数の値を空白区切りで指定したもの. 例えば局所基底がスピンの z 成分ではられる場合, z 成分の大きさ.

Algorithm/Site/VertexTypeOfSource　挿入される可能性のあるバーテックスのタイプ.

Algorithm/Site/InitialConfiguration　初期条件の定義. 初期条件ごとのワーム対の生成消滅過程を定義もこの要素のなかで行われます. サブ要素として, State, NumberOfChannels, Channel があります.

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

(次のページに続く)

(前のページからの続き)

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

Algorithm/Site/InitialConfiguration/State　ワーム対が生成される前（もしくは消滅後）のサイトの状態.

Algorithm/Site/InitialConfiguration/NumberOfChannels　可能性のある終状態（チャネル）の数.

Algorithm/Site/InitialConfiguration/Channel　各チャネルの定義. 整数値, 整数値, 浮動小数点値の 3 つの並びで指定.

- 第 1 の値はワーム生成後のヘッドの向き（0 は虚時間方向負の向き, 1 は正の向き.）.
- 第 2 の値はワーム生成後のヘッドとテールの間の状態.
- 第 3 の値はそのような終状態をとる確率.

終状態としてワーム対を生成しない場合は, その Channel の 第 1 と第 2 の整数値はともに -1 とする.

Algorithm/Interaction　1 つの相互作用型を定義します. サブ要素として IType, VType, NBody, EBase, VertexDensity, Sign があります.

```
<Algorithm>
  ...
  <Interaction>
    <IType> 0 </IType>
    <VType> 1 </VType>
    <NBody> 2 </NBody>
    <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　相互作用の型の識別番号.

Algorithm/Interaction/VType　挿入する可能性のあるバーテックスの型の識別番号. バーテックス型の内容は Vertex/Algorithm で定義します.

Algorithm/Interaction/NBody　相互作用に関与するサイトの数（ゼーマン項のような 1 体相互作用であれば 1 で, 交換相互作用のような 2 体相互作用であれば 2. 3 以上を指定することも可能）.

Algorithm/Interaction/EBase　エネルギーオフセットの値. シミュレーション自体には影響しませんが, 最終的なエネルギーの値を出すときに使用されます.

Algorithm/Interaction/VertexDensity　関与するサイトの状態ごとに挿入するバーテックスの密度を指定します. Algorithm/Interaction/NBody 個の整数値と, 1 個の浮動小数点値の並びで指定. 整数値は, 関与する各サイトの状態（順序は格子定義ファイルの I で指定するサイト番号の順序と対応します）. 浮動小数点値は密度.

Algorithm/Interaction/Sign　その相互作用における局所重みの符号, すなわち $\mathrm{Sgn}(\langle f|-\mathcal{H}|i\rangle)$ を指定します. 2× Algorithm/Interaction/NBody 個の整数値と, 1 個の浮動小数点値の並びで指定. 整数値は, 関与する各サイトのそれぞれについて, 相互作用演算子が適用される前と後の状態で, 浮動小数点値は重みの符号.

例えば, `<Sign> 0 1 1 0 -1.0 </Sign>` は $\langle 10|\left(-\mathcal{H}\right)|01\rangle < 0$ を意味します.

Algorithm/Vertex　1 つのバーテックスの型を定義します. バーテックスとしては, 通常の 2 体, 3 体, ……の相互作用を記述するもの（`VCategory=2`）と, ワームヘッドがテールと消滅する場合のテール（`VCategory=1`）があります. Algorithm/Interaction の要素になりえるのは, 前者です. （このほか, 時間方向の周期境界（`VCategory=0`）も 1 体のバーテックスとして扱っていますが, これをユーザが定義する必要はありません.）サブ要素として VType, VCategory, NBody, NumberOfInitialConfigurations, InitialConfiguration があります.

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

Algorithm/Vertex/VType　バーテックス型の識別番号. バーテックス型の定義ごとに異なる番号である必要があります.

Algorithm/Vertex/VCategory　1 がワームテール, 2 が相互作用.

Algorithm/Vertex/NBody　相互作用に関与するサイトの個数. テールの場合には 1.

Algorithm/Vertex/NumberOfInitialConfigurations　バーテックスの可能な初期状態数.

Algorithm/Vertex/InitialConfiguration　特定のバーテックス初期状態に対するワームの可能なアクションを定義します. 従って, この要素は, Algorithm/Vertex/NumberOfInitialConfigurations の値と同じ数だけ存在する必要があります. サブ要素として, State, IncomingDirection, NewState, NumberOfChannels, Channel があります.

```
<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>
```

この例で定義されているのは,「バーテックスの左下 (0), 左上 (1), 右下 (2), 右上 (3) の脚の状態がそれぞれ 1, 0, 0, 1 であって, そこに, 左下（脚 0 の方向）から, その脚の状態を 0 に変化させるような ワームヘッドが入射した場合」のアクションであり, その内容は,「確率 1 で, そのワームヘッドを 脚 3 の方向に散乱させて, その方向の足の状態を 0 に変更する」ことを表しています. （つまり, この散乱が起こった場合, 散乱後のバーテックスの状態は 0, 0, 0, 0 になる.）

Algorithm/Vertex/InitialConfiguration/State　ワームヘッドが入ってくる前のバーテックスの初期状態を指定します. 具体的にはバーテックスの各脚の状態を指定します. 足の本数は, Algorithm/Vertex/NBody で指定される数 (=m) の 2 倍なので, 2m 個数の整数値をスペースで区切ったものを入力します. その順序として, 脚は対応するサイトの順序に並べられ, 同じサイトに対応する 2 本の脚については, 虚数時間の小さい側が先に来ます. （サイトの並び順は任意でよいが, 格子定義ファイルの Lattice/I 要素で 指定されているサイトの並び順はここで用いられたサイトの順序と整合している必要があります.）各整数はバーテックスの足の状態を示す 0 から n-1 までの値. （ここで, n は対応するサイトの, Algorithm/Site/NumberOfStates で指定される値.）

Algorithm/Vertex/InitialConfiguration/IncomingDirection　入射するワームヘッドが入射前に乗っている脚の番号. 対応する足が Algorithm/Vertex/InitialConfiguration/State の記述において何番目に出てくるかを 0 から 2m-1 の整数値で指定.

Algorithm/Vertex/InitialConfiguration/NewState　ワームヘッドが通過したあとの Algorithm/Vertex/InitialConfiguration/IncomingDirection の足の状態. 0 から n-1 の整数値で指定.

Algorithm/Vertex/InitialConfiguration/NumberOfChannels　可能な散乱チャネルの個数.

Algorithm/Vertex/InitialConfiguration/Channel　散乱チャネルの定義. Algorithm/Vertex/InitialConfiguration/NumberOfChannels の個数だけこの要素を用意する必要があります. 2 つの整数値と 1 つの浮動小数点値をスペースで区切ったもので指定.

- 第 1 の整数値は, 散乱後のワームヘッドが乗っている足の番号を 0 から 2m-1 の値で指定したもの.
- 第 2 の整数値は, ワームヘッドが飛び去ったあとのその足の状態を 0 から n-1 の値で指定したもの.

- 第 3 の浮動小数点値は, そのチャネルを選ぶ確率.

特別な場合として, ワームヘッドがテールに衝突して消滅する場合があり, この場合は 第 1 引数と第 2 引数に -1 を指定します.

4.3.5 波数ベクトル XML ファイル

波数ベクトル XML ファイルは, スタッガード秩序変数

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

や動的構造因子

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

波数表示温度グリーン関数

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

を計算するための波数の情報が XML ライクな形式で記述されるテキストファイルです. 波数ベクトルデータファイルから `dla_alg` を用いて生成可能です。

波数ベクトル XML ファイルはただ一つの要素 WaveVector を持ち, すべての情報は WaveVector 要素の内容として含まれます.

WaveVector　ファイル全体の要素名. サブ要素として, Comment, NumberOfSites, NumberOfWaveVectors, RK があります.

WaveVector/Comment　省略可能. コメント文を示し, 計算には使用されません.

WaveVector/NumberOfSites　系のサイト数.

WaveVector/NumberOfWaveVectors　波数 $\vec{k}$ の数.

WaveVector/RK　内積 $\vec{r}\cdot\vec{k}$ の情報. NumberOfSites と NumberOfWaveVectors の積だけ指定する必要があります. 内容として,「$\cos(\theta)$ の値」,「$\sin(\theta)$ の値」,「サイト番号」,「波数番号」 の 4 つの数字をスペース区切りで持ちます. ここで θ はサイト番号で示されるサイトの座標 $\vec{r}$ と波数番号で示される波数 $\vec{k}$ との内積です.

4.3.6 相対座標定義ファイル

相対座標定義ファイルは, 実空間表示温度グリーン関数

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

を計算するための相対座標 $\vec{r}_{ij}$ の情報が XML ライクな形式で記述されるテキストファイルです. `dla_alg` を用いて生成可能です.

格子ファイルはただ一つの要素 Displacements を持ち, すべての情報は Displacements 要素の内容として含まれます.

Displacements　ファイル全体の要素名. サブ要素として, Comment, NumberOfKinds, NumberOfSites, R があります.

Displacements/Comment　省略可能. コメント文を示し, 計算には使用されません.

Displacements/NumberOfSites　系のサイト数.

Displacements/NumberOfKinds　取りうる相対座標の数.

Displacements/R　内容として,「相対座標のインデックス」,「サイト i のインデックス」,「サイト j のインデックス」 の 3 つの整数をスペース区切りで持ちます.

4.4 DLA の入力ファイル生成ツール

DLA は入力ファイルとして格子定義ファイル, アルゴリズム定義ファイル, 波数ベクトル定義ファイルを, それぞれ XML 形式ファイルとして受け取ります. これらをうまく定義することで, 計算機資源の許す範囲で任意の格子や模型を計算できますが, 手で定義するには複雑になっています. そのため, 超立方格子やハイゼンベルグ模型などのよく使われるような格子・模型については生成ツールが用意されています.

4.4.1 シンプルモードツール `dla_pre`

`dla_pre` は **シンプルモードファイル** から *DLA* **のエキスパートモード入力ファイル** を生成するツールです.

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

パラメータは以下の通り.

`paramfile`　出力されるパラメータファイル名. デフォルトは `qmc.inp` です.

`inputfile`　入力ファイル名. ファイル形式の詳細は **シンプルモードファイル** を参照してください.

格子 XML ファイルなど、出力される XML ファイルの名前はパラメータから自動生成されます。

4.4.2 格子生成ツール `dla_latgen`

`dla_latgen` は **シンプルモードファイル** から **格子データファイル** や **格子** *TOML* **ファイル** を生成するツールです.

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

パラメータは以下の通り.

datafile　出力される格子データファイルの名前. デフォルトは lattice.dat です. 空文字列の場合, 格子データファイルは出力されません.

TOML　出力される格子 TOML ファイルの名前. デフォルトは空文字列で, ファイルは出力されません.

GNUPLOT　出力される格子 Gnuplot ファイルの名前. デフォルトは空文字列で, ファイルは出力されません. 出力されたファイルを ‘‘gnuplot‘‘ で load することで、格子を可視化することができます.

inputfile　入力ファイル名. ファイル形式の詳細は *lattice* を参照してください.

実行すると datafile や TOML で指定した名前の格子定義ファイルが生成されます.

入力ファイル例

```
# 1 次元鎖, 8 sites
[lattice]
lattice = "hypercubic"
dim = 1
L = 8

# 2 次元正方格子, 4x4 sites
[lattice]
lattice = "hypercubic"
dim = 2
L = 4

# leg はしご格子, 8x2 sites
[lattice]
lattice = "hypercubic"
dim = 2
L = [8,2]
bc = [true, false]
```

4.4.3 ハミルトニアン生成ツール dla_hamgen

dla_hamgen は **シンプルモードファイル** から **ハミルトニアン** *TOML* **ファイル** を生成するツールです.

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

パラメータは以下の通り.

filename　出力ファイル名. デフォルトは hamiltonian.toml です.

inputfile　入力ファイル名. ファイル形式は *hamiltonian* を参照してください.

実行すると filename で指定した名前を持つファイルが生成されます.

入力ファイル例

```
# 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 パラメータファイル生成ツール dla_pgen

dla_pgen は **シンプルモードファイル** から **パラメータファイル** を生成するツールです.

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

パラメータは以下の通り.

filename　出力ファイル名. デフォルトは param.in です.

inputfile　入力ファイル名. ファイル形式は *parameter* を参照してください.

4.4.5 波数ファイル生成ツール dla_wvgen

dla_wvgen は **シンプルモードファイル** から **波数ファイル** を生成するツールです.

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

パラメータは以下の通り.

`filename`　出力ファイル名. デフォルトは `kpoints.dat` です.

`size`　格子サイズ. 数字を空白区切りで並べた文字列で指定します (e.g. `-s "4 4"` .) 指定しない場合は, 入力ファイルの `[lattice]` テーブルから読み取ります.

`inputfile`　入力ファイル名. ファイル形式は *kpoints* を参照してください.

実行すると filename で指定した名前の波数ファイルが生成されます.

4.4.6 アルゴリズム生成ツール `dla_alg`

`dla_alg` は**格子データファイル**, **格子** *TOML* **ファイル**, **ハミルトニアン** *TOML* **ファイル**, **波数ファイル** から**格子** *XML* **ファイル**, **アルゴリズム定義ファイル**, **波数ベクトル** *XML* **ファイル**, **相対座標定義ファイル** を生成するツールです.

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

パラメータは以下の通り.

`LAT`　読み込む格子 dat/TOML ファイル. 省略した場合は `lattice.dat` が指定されます.

`HAM`　読み込むハミルトニアン TOML ファイル. 省略した場合は `hamiltonian.toml` が指定されます.

`LATXML`　書き出される格子定義ファイル. 省略した場合は `lattice.xml` が指定されます.

`ALGXML`　書き出されるアルゴリズム定義ファイル. 省略した場合は `algorithm.xml` が指定されます.

`without_lattice`　設定した場合、格子定義ファイルは書き出されません。なお、格子ファイル `LAT` そのものは、アルゴリズムなどの導出に必要なために読み込まれます。

`without_algorithm`　設定した場合、アルゴリズム定義ファイルは書き出されません。

`KPOINT`　読み込む波数ファイル. 省略した場合は波数ベクトル XML ファイルは出力されません.

`WV`　書き出される波数ベクトル XML ファイル. 省略した場合は `wavevector.xml` が指定されます.

`DISP`　書き出される相対座標定義ファイル。省略した場合は相対座標 XML ファイルは出力されません.

`--distance-only`　指定した場合、変位定義において変位 $\vec{r}_{ij}$ ではなくその絶対値 r_{ij} でグループ化します。

KERNEL　バーテックスにおけるワームヘッドの散乱確率の導出に使うアルゴリズム。省略した場合、 suwa todo が用いられます。利用できるアルゴリズムは *algorithm* を参照してください.

4.5 向き付きループアルゴリズムソルバ dla

dla は向き付きループアルゴリズムを実装した量子モンテカルロプログラムです. コマンドライン引数として入力ファイルをとります.

MPI 実行した場合, 指定したプロセスの数 N_{proc} だけ乱数並列を行います. 各プロセスは独立に, 入力ファイル中の NSET で指定したセット数だけモンテカルロ計算をします. その結果, 合計のセット数が N_{proc} 倍され, 統計誤差は $1/\sqrt{N_{\mathrm{proc}}}$ 倍になることが期待されます.

実行例

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

4.6 DLA の出力ファイル

4.6.1 フォーマット

DLA は計算結果を行区切りのプレーンテキストファイルで出力します. 行頭の文字はその行の意味を表します.

P <name> = <value>　入力パラメータファイルと格子ファイルから読み取ったパラメータ.

R <name> = <mean> <error>　計算で求められた物理量. <mean> は平均値を, <error> は標準誤差を示します.

I <text> = <value>　その他計算中に得られた情報.

C <text>　コメント.

以下にサンプル（反強磁性ハイゼンベルグ鎖）を示します.

```
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
```

(次のページに続く)

(前のページからの続き)

```
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
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
```

以下の物理量定義に現れる記号の意味を示します.

N_s　サイト数.

$Q(\vec{k})$　格子点 i 上で定義される任意の演算子 Q_i のフーリエ変換.

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

$Q(\tau)$　虚時間 τ における演算子.

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

$\tilde{Q}$　任意の演算子 Q について, 虚時間方向の平均 $\frac{1}{\beta}\int_0^\beta \mathrm{d}\tau Q(\tau)$

M^z　局所自由度の量子化軸方向成分. たとえばスピン系では局在スピン演算子の z 成分 S^z で, ボース粒子系では数演算子 n です.

$M^{\pm}$　M^z の昇降演算子. スピン系では $M^{\pm} \equiv S^{\pm}$, ボース粒子系では生成消滅演算子 $M^{+} \equiv b^{\dagger}$ および $M^{-} \equiv b$.

M^x　非対角秩序変数. スピン系では $M^x \equiv (S^{+} + S^{-})/2$, ボース粒子系では $M^x \equiv (b + b^{\dagger})$.

T　温度.

β　逆温度.

h　M^z に共役な外場. スピン系では縦磁場, ボース粒子系では化学ポテンシャル.

$\langle Q \rangle$　任意の演算子 Q のグランドカノニカル平均.

4.6.2 メイン出力

メイン出力ファイルは, 入力パラメータファイルの `outfile` キーワードで指定した名前で出力されます.

`sign`　重みの符号.

$\frac{\sum_i W_i}{\sum_i |W_i|}$, ここで i はモンテカルロサンプルの番号.

`anv`　平均バーテックス数.

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

`ene`　エネルギー密度.

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

`spe`　比熱.

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

`som`　比熱と温度の比.

$$\gamma \equiv \frac{C_V}{T} = \beta C_V$$

`len`　平均ワーム長さ.

`xmx`　横感受率.

`amzu`　「磁化」(uniform, $\tau = 0$).

$m^z \equiv \frac{1}{N_s} \sum_i^{N_s} M_i^z$ としたときの $\langle m^z \rangle$.

`bmzu`　「磁化」(uniform, τ 平均). $\langle \tilde{m}^z \rangle$.

`smzu`　構造因子 (uniform).

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

`xmzu` 縦感受率 (uniform).

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

`amzsK` 「磁化」("staggered", $\tau = 0$)

$m_K^z \equiv \dfrac{1}{N_s} \displaystyle\sum_i^{N_s} M_i^z \cos\left(\vec{k}\cdot\vec{r_i}\right)$ としたときの $\langle m_s^z \rangle$. K は波数ベクトル XML ファイルで定義された波数 k のインデックス.

`bmzuK` 「磁化」("staggered", τ 平均). $\langle \tilde{m}_K^z \rangle$.

`smzsK` 構造因子 ("staggered").

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

`xmzsK` 縦感受率 ("staggered").

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

4.6.3 構造因子出力ファイル

構造因子出力ファイルは, 入力パラメータファイルの `sfoutfile` キーワードで指定した名前で出力されます. このファイルには虚時間構造因子

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

が出力されます. 波数 $\vec{k}$ や虚時間 τ の値は, 物理量名を用いて

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

のように `C<k>t<t>` という形で区別されます. ここで `<k>` は波数ベクトル XML ファイルの `kindex` (`KR` タグの最終要素) で指定される波数のインデックスで, `<t>` は離散化した虚時間のインデックス.

4.6.4 実空間表示温度グリーン関数出力ファイル

実空間表示温度グリーン関数出力ファイルは, 入力パラメータファイルの `cfoutfile` キーワードで指定した名前で出力されます. このファイルには温度グリーン関数

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

が出力されます. 変位 $\vec{r}_{ij}$ や虚時間 τ の値は構造因子と同様に, `C<k>t<t>` という形で物理量名によって区別されます. ここで `<k>` は変位 XML ファイルの `kind` (`R` タグの第一要素) で指定される変位のインデックスで, `<t>` は離散化した虚時間のインデックス.

4.6.5 波数表示温度グリーン関数出力ファイル

波数表示温度グリーン関数出力ファイルは, 入力パラメータファイルの `ckoutfile` キーワードで指定した名前で出力されます. このファイルには温度グリーン関数

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

が出力されます. 波数 $\vec{k}$ や虚時間 τ の値は構造因子と同様に, `C<k>t<t>` という形で物理量名によって区別されます. ここで `<k>` は波数ベクトル XML ファイルの `kindex` (`RK` タグの最終要素) で指定される波数のインデックスで, `<t>` は離散化した虚時間のインデックス.

第 5 章

DSQSS/PMWA のチュートリアル

5.1 DSQSS/PMWA とは？

大規模並列計算向け非局所更新アルゴリズムを実装した世界線量子モンテカルロ法のパッケージ. 有限温度の大規模格子系を取り扱うことができ, S=1/2 量子多体系（XXZ スピン模型,Heisenberg 模型, 横磁場 Ising 摸型などのスピン摸型やハードコア・ボース粒子系など）を統計誤差の範囲内で厳密に計算することができます. 基本的には任意の物理量が測定可能ですが, デフォルトではワームソース場有限の下でのエネルギー, 縦磁化, 横磁化, ワインディングナンバーや, 虚時間 S^zS^z スピン相関関数などを測定できます.

5.2 使用方法

PMWA は以下の手順で使用します。

1. 格子定義ファイルの作成 (`lattgene_P` を利用)
2. 入力ファイルの作成
3. PMWA の実行 (ボソン系：`pmwa_B` , スピン系：`pmwa_H` の実行)
4. 出力ファイルの生成

また,1, 2 を同時に行うための簡易ツール `pmwa_pre.py` も用意されています。ここでは 1-4 について使用方法やパラメータについて順に説明します。

5.2.1 格子定義ファイルの作成

PMWA では標準的な模型に対して格子定義ファイル `lattice.xml` を生成するための簡易ツールとして `lattgene` を用意しています. ここでは `lattgene` の使用方法について説明します.

`lattgene` では cubic lattice に関する格子定義ファイルを作成することが出来ます. 指定するパラメータは下記の通りです.

Parameter	type	備考
D	int	次元数
L	int	格子のサイズ (各次元について連続で指定します)
B	double	逆温度
NLdiv	int	L の分割数 (各次元についてそれぞれ NLdiv 分割します)
NBdiv	int	B の分割数
NFIELD	int	磁場の種類の数 (基本的には 0 に設定)

使用例を以下に記載します.

1. 1 次元 8 サイト, 逆温度 10.0, 分割数は 1 の場合の格子ファイルを定義

   ```
   $ lattgene 1 8 10.0 1 1 0
   ```

2. 2 次元 4*4 サイト, 逆温度 10.0, 分割数は 1 の場合の格子ファイルを定義

   ```
   $ lattgene 2 4 4 10.0 1 1 0
   ```

5.2.2 DSQSS/PMWA の入力ファイルの作成

PMWA を実行するには, テキスト形式の入力ファイルが必要です. 入力ファイルでは計算条件を指定するパラメータとハミルトニアンを指定するパラメータの二種類があります.

以下, 入力ファイル例を記載します

```
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
```

各パラメータの意味は下記の通りです.

- 計算条件のパラメータ

Parameter	type	備考
RUNTYPE	int	計算モード (0: 通常計算,1: リスタート計算)
CB	int	初期配置 (0: Vacuum,1: Checker-Board,2: Random)
NSET	int	モンテカルロ計算の繰り返し数
NMCS	int	物理量計算に用いるモンテカルロスイープ数
NPRE	int	1 スイープあたりのワーム生成試行回数を決めるための事前計算につかうモンテカルロステップ数
NMCSE	int	初期緩和に用いるモンテカルロスイープ数
NMCSD	int	測定間のモンテカルロスイープ数
SEED	int	0 以上の場合は実際に使用するシード, 負の場合は乱数でシードを与える.
NVERMAX	int	バーテックスの最大数 (デフォルト数 10^8),-1 の場合は上限なし.
NWORMAX	int	ワームの最大数 (デフォルト数 10^3),-1 の場合は上限なし.
SFINPFILE	str	入力された場合, 本ファイルで指定された Structure Factors を計算する.
SFOUTFILE	str	入力された場合,Structure Factors を指定したファイルに出力する (デフォルトは sf.out).
Step_x	int	相関関数を計算する場合の空間幅を与える (デフォルト:1).
Step_k	int	波数表示の相関関数を計算する場合の波数空間幅を与える (デフォルト:1).
CFOUTFILE	str	入力された場合, 相関関数を指定したファイルに出力する (デフォルトは cf.out).

ここで,NVERMAX,NWORMAX については自動でリサイズして決定します (ver. 1.2.0 で実装).

- 相互作用関連のパラメータ

PMWA ではハードコアボソン系 (サイトに最大 1 つのボソン) と S=1/2 の XXZ 模型について計算可能です. ハードコアボソン系のハミルトニアンは

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

で与えられます. ここで $\langle i,j \rangle$ は最近接のペアを表します. S=1/2 の XXZ 模型は

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

で与えられます. 入力ファイルで指定するパラメータと上記式のパラメータは以下のように対応しています.

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

各パラメータは double 型で指定します.

5.2.3 計算実行

入力ファイル作成後, 以下のコマンドを入力することで実行できます (入力ファイルは `param.in` としています). スピン系とハードコアボソン系に応じて,

それぞれ実行ファイルが異なります.

1. スピン系の場合

   ```
   $pmwa_H param.in
   ```

2. ハードコアボソンの場合

   ```
   $pmwa_B param.in
   ```

計算終了後, 結果ファイル 1 つとリスタート用の一時ファイル 2 つ (evout_sample.log, RND_evout_sample.log) が出力されます.

5.2.4 出力ファイル

ここでは結果ファイル 1 つとリスタート用の一時ファイル 2 つについて, PMWA に特有のパラメータをそれぞれ記載します.

- 結果ファイル

Kind	Name	Description
P	L	三次元の格子情報
P	DOML	並列化により分割されたドメインの大きさ
P	DOMBETA	並列化により分割された逆温度のドメインサイズ
P	NDIVL	格子の分割数
P	NTEST	テストをするサンプルの数 (詳細はモンテカルロ計算の箇所で説明)
R	nver	キンクとワームの数
R	nkin	キンクの数
R	wndx	x 方向のワインディング数の二乗の期待値
R	wndy	y 方向のワインディング数の二乗の期待値
R	wndz	z 方向のワインディング数の二乗の期待値
R	wnd2	ワインディング数の二乗の総数 (wndx+wndy+wndz)
R	bmxu	S_x の期待値 (uniform τ 積分)
R	bmpu	S_+ の期待値 (uniform τ 積分)
R	bmmu	S_- の期待値 (uniform τ 積分)
R	comp	圧縮率
R	lxmx	各サイトの局所的なワーム数の揺らぎ
I	the maximum number of vertices	バーテックスの最大数
I	the maximum number of worms	ワームの最大数

ここで種別は出力の各行の先頭に付与される文字で, P, R, I はそれぞれ Parameter, Result, Information を示します.

- リスタート用のファイル

PMWA ではリスタート機能が実装されており, 下記の 2 ファイルが存在する場合には強制的にリスタートが行われます. 以下, 各ファイルの出力内容について簡単に説明します.

1. evout_sample.log

計算終了時のサイクル数, ワールドラインの情報, バーテックスの情報について出力したファイル. 再計算時には読み込んだ配置を始条件として計算が行われます.

```
26 : 計算終了時のサイクル数
0 1 : ドメイン内のサイト 0 のワールドラインの情報
i/N beta, (i+1)/N beta 区間のワールドラインの情報 : 0: down, 1: up
0 0 : ドメイン内のサイト 1 のワールドラインの情報
1 1  :ドメイン内のサイト 2 のワールドラインの情報
```

(次のページに続く)

(前のページからの続き)

```
...
8 0.056997107 2 1 4 :バーテックスのラベル, tau, バーテックスの種類 , ワールドラインの本数, ボンド番号
9 0.056997107 2 0 5
44 0.28066013 2 1 3
```

ここでバーテックスの種類については以下のものが定義されています.

バーテックスの種類	説明
-5	各ドメインにおける虚時間方向の始点及び終点. ドメイン分割しなくても有効.
-4(右), -2(左)	ドメインをまたぐ対角的バーテックス. ドメイン分割しなくても有効.
-3(右), -1(左)	ドメインをまたぐ非対角的バーテックス (キンク). ドメイン分割しなくても有効.
0	オンサイトバーテックス (ワーム以外).
1	2 サイトバーテックス.
2	キンク.
3	2 サイトバーテックス (次近接)(互換性のために残してあります).
4	その時動いているワーム (消滅演算子).
5	その時動いているワーム (生成演算子).
6	その時止まっているワーム (生成消滅関係なし), もしくは不要なバーテックス.
7	マーカー (虚時間相関関数測定用).

2. RNDevout_sample.log

乱数生成を行っているオブジェクトをバイナリ形式で出力したファイル. 再計算時には読み込んだ乱数情報を始条件として計算が行われます.

5.3 DSQSS/PMWA によるスピン鎖のエネルギー計算

このチュートリアルでは, S=1/2 の反強磁性ハイゼンベルグ鎖の基底状態エネルギー計算をすることで、DSQSS/PMWA の使い方を学びます.

DSQSS/PMWA による計算は,

1. 入力ファイルの準備
2. 計算の実行
3. 計算結果の解釈

の 3 段階に分かれます.

5.3.1 入力ファイルの準備

DSQSS/PMWA を実行するには,

1. 格子定義ファイル
2. パラメータファイル

の 2 つの入力ファイルが必要です. そのため, まずはこれらの入力ファイルを作成します. そのためのユーティリティツールが pmwa_pre です. これは単一の入力ファイルから, DSQSS/PMWA の入力ファイルを生成する Python スクリプトです. まず, pmwa_pre の入力ファイルとして, 次の内容を持つテキストファイル std.in を準備します.

```
[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
```

自分の好きなエディタで書くか, sample/pmwa/1DDimer ディレクトリにあるものを利用してください. このファイルを pmwa_pre に与えます.

```
$ $DSQSS_ROOT/bin/dsqss_pre.py std.in
```

この結果, パラメータファイル param.in, 格子定義ファイル lattice.xml が作成されます.

5.3.2 計算の実行

入力ファイルを作成したら, DSQSS/PMWA による計算を実行します.

```
$ $DSQSS_ROOT/bin/pmwa_H param.in
```

なお, 計算を実行するときに MPI を用いることで, 乱数並列計算が可能です (入力ファイルの指定により空間分割, 虚時間方向の分割を行うこともできます. 詳細は DLA のユーザーマニュアルをご覧ください).

```
$ mpiexec -np 4 $DSQSS_ROOT/bin/pmwa_H param.in
```

並列数 (今回は 4) だけ独立に計算を行い, モンテカルロサンプル数を増やすことで計算精度を向上できます.

5.3.3 計算結果の解釈

計算結果は出力ファイル sample.log に書き出されます. エネルギーは

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

となります. なお, DSQSS/PMWA の場合は横磁場を必ず入れる必要があります. そのため, ゼロ磁場にするには外挿する必要があるので, ご注意ください.

第 6 章

DSQSS/PMWA のユーザーマニュアル

6.1 DSQSS/PMWA の入力ファイル

DSQSS/DLA と DSQSS/PMWA の入力ファイルでは, 共通するパラメータが多く存在します. ここでは DSQSS/DLA と使用方法が異なる, もしくは新規に追加されたパラメータについて記載します.

- 計算条件のパラメータ

Parameter	type	default	備考
RUNTYPE	int		計算モード (0: 通常計算,1: リスタート計算)
CB	int	0	初期配置 (0: Vacuum,1: Checker-Board,2: Random)
NSET	int	10	モンテカルロ計算の繰り返し数
NMCS	int	1000	測定する際のモンテカルロステップ (Number of Monte Carlo Steps)
NPRE	int	1000	ハイパーパラメータ決定のためのモンテカルロステップ数 (Number of Pre-calculation)
NTHERM	int	1000	空回しモンテカルロステップ数 (Number of Monte Carlo Steps for Thermalization)
NDECOR	int	1000	測定間の間隔数 (Decorrelation)
SEED	int	13	0 以上の場合は実際に使用するシード, 負の場合は乱数でシードを与える.
NVERMAX	int	100000000	バーテックスの最大数,-1 の場合は上限なし.
NWORMAX	int	1000	ワームの最大数,-1 の場合は上限なし.
SFINPFILE	str		入力された場合, 本ファイルで指定された Structure Factors を計算する.
SFOUTFILE	str	sf.out	入力された場合,Structure Factors を指定したファイルに出力する.
Step_x	int	1	相関関数を計算する場合の空間幅を与える.
Step_k	int	1	波数表示の相関関数を計算する場合の波数空間幅を与える.
CFOUTFILE	str	cf.out	入力された場合, 相関関数を指定したファイルに出力する.

- 模型関連のパラメータ

Parameter	type	備考
beta	double	逆温度.
t	double	ボソン系では t を, スピン系では J_{xy} を表す.
U	double	ボソン系で U を表す. スピン系では使用されない.
V	double	ボソン系で V を, スピン系では J_z を表す.
MU	double	ボソン系では μ を, スピン系では H を表す.
G	double	ボソン系では Γ を, スピン系では $\Gamma/2$ を表す.
NMAX	-	1 に固定

入力ファイル例を以下に示します.

```
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 DSQSS/PMWA の出力ファイル

PMWA では計算後, 結果ファイル 1 つとリスタート用の一時ファイル 2 つ (evout_sample.log, RND_evout_sample.log) を出力します.

- 結果ファイル

  多くのパラメータは DLA と同じです. ここでは PMWA に特有もしくは DLA と異なったパラメータについて記載します.

Kind	Name	Description
P	L	三次元の格子情報
P	DOML	並列化により分割されたドメインの大きさ
P	DOMBETA	並列化により分割された逆温度のドメインサイズ
P	NDIVL	格子の分割数
P	NTEST	テストをするサンプルの数 (詳細はモンテカルロ計算の箇所で説明)
R	nver	キンクとワームの数
R	nkin	キンクの数
R	wndx	x 方向のワインディング数の二乗の期待値
R	wndy	y 方向のワインディング数の二乗の期待値
R	wndz	z 方向のワインディング数の二乗の期待値
R	wnd2	ワインディング数の二乗の総数 (wndx+wndy+wndz)
R	bmxu	S_x の期待値 (uniform τ 積分)
R	bmpu	S_+ の期待値 (uniform τ 積分)
R	bmmu	S_- の期待値 (uniform τ 積分)
R	comp	圧縮率
R	lxmx	各サイトの局所的なワーム数の揺らぎ
I	the maximum number of vertices	バーテックスの最大数
I	the maximum number of worms	ワームの最大数

ここで種別は出力の各行の先頭に付与される文字で,P, R, I はそれぞれ Parameter, Result, Information を示します.

- リスタート用のファイル

PMWA ではリスタート機能が実装されており, 下記の 2 ファイルが存在する場合には強制的にリスタートが行われます. 以下, 各ファイルの出力内容について簡単に説明します.

1. evout_sample.log

計算終了時のサイクル数, ワールドラインの情報, バーテックスの情報について出力したファイル. 再計算時には読み込んだ配置を始条件として計算が行われます.

```
26 : 計算終了時のサイクル数
0 1 : ドメイン内のサイト 0 のワールドラインの情報
i/N beta, (i+1)/N beta 区間のワールドラインの情報；0: down, 1: up
0 0：ドメイン内のサイト 1 のワールドラインの情報
1 1：ドメイン内のサイト 2 のワールドラインの情報
```

(次のページに続く)

(前のページからの続き)

```
...
8 0.056997107 2 1 4 :バーテックスのラベル, tau, バーテックスの種類 , ワールドラインの本数, ボンド番号
9 0.056997107 2 0 5
44 0.28066013 2 1 3
```

ここでバーテックスの種類については以下のものが定義されています.

バーテックスの種類	説明
-5	各ドメインにおける虚時間方向の始点及び終点（ドメイン分割しなくても有効）
-2(左) , -4(右)	ドメインをまたぐ対角的バーテックス（ドメイン分割しなくても有効）
-1(左) , -3(右)	ドメインをまたぐ非対角的バーテックス（キンク）（ドメイン分割しなくても有効）
0	オンサイトバーテックス（ワーム以外）
1	2 サイトバーテックス
2	キンク
3	2 サイトバーテックス（次近接）（互換性のために残してあります）
4	その時動いているワーム（消滅演算子（このワームより τ が小さい側の方がワールドラインの本数が多い)）
5	その時動いているワーム（生成演算子（このワームより τ が大きい側の方がワールドラインの本数が多い)）
6	その時止まっているワーム（生成消滅関係なし, 双方向リストには繋がっている）, もしくはいらなくなった（双方向リストに繋がっていない）バーテックス
7	マーカー （虚時間相関関数測定用）

2. RNDevout_sample.log

乱数生成を行っているオブジェクトをバイナリ形式で出力したファイル. 再計算時には読み込んだ乱数情報を始条件として計算が行われます.

第 7 章

アルゴリズム

7.1 経路積分サンプリング

DSQSS では, 分配関数を

$$Z \equiv \mathrm{Tr} e^{-\beta H} = \sum_S W(S)$$

と経路積分表示したのち, マルコフ過程によって状態 S を確率的時系列的に発生し, これをサンプリングします（経路積分モンテカルロ法）. ここで H は系を記述するハミルトニアンで, 「状態」とは $d+1$ 次元時空上で定義された古典変数の場です（モデルが定義されている空間次元を d として, これに虚数時間軸を加えたものを $d+1$ 次元時空と呼びます). また c-数 $W(S)$ は状態 S のボルツマン重みです. 「状態」をひとつ定めることは, スピンや粒子の空間的な配置が虚数時間の増加に伴って変化する経路をひとつ定めることと等価であるため, 状態 S は「経路」とも呼ばれます. ボーズ粒子系のように局所的な粒子数保存則が成立する場合に, 状態を粒子の存在位置を実線でつないだ軌跡（＝世界線）の集まりとして視覚化することが多いため, 経路積分モンテカルロ法は「世界線モンテカルロ法」とも呼ばれます. マルコフ過程の遷移確率は, 定常分布において状態の出現頻度が重み $W(S)$ に比例するように定義されます. マルコフ過程で順次出現する状態を $S_t (t = 1, 2, 3....)$ とした時, 演算子 Q の期待値

$$\langle Q \rangle \equiv \mathrm{Tr}(Qe^{-\beta H})/\mathrm{Tr}(e^{-\beta H})$$

は Q に対応する観測量 $Q(S)$ の統計的期待値

$$\langle Q \rangle_{\mathrm{MC}} \equiv \frac{1}{N_{\mathrm{MCS}}} \sum_{t=N_{\mathrm{therm}}+1}^{N_{\mathrm{MCS}}+N_{\mathrm{therm}}} Q(S_t)$$

によって近似されます. この近似は初期条件の影響による系統誤差と, サンプリングによる統計誤差を含みます. 系統誤差が無視できるためには, 空回し数 N_{therm} が初期緩和時間よりも大きいことが必要で, その限りでは指数関数的に速やかに収束します. 一方統計誤差はサンプル数 N_{MCS} を大きくしたときに, この $1/2$ 乗に逆比例して小さくなることが期待されます.

7.2 ワーム更新法

マルコフ過程の遷移確率の構成方法, すなわち状態更新方法にはいろいろな種類があり, それぞれ利点・欠点があります. スピン系, ボーズ系のシミュレーションで用いられる代表的な更新方法はループ更新とワーム更新です. ループ更新は全系をループと呼ばれるクラスターに分割してループごとに状態更新するもので, 高速な更新が可能だが, 一様磁場中での反強磁性体やボーズ系では効率が著しく低下してしまいます. 一方ワーム更新は, 全系に保存則を破る点（ワーム）を 2 つ導入して, これらの点を移動させることで状態を更新する方法です. ワームは非対角成分のみをもつ演算子に対応します. たとえば, 各点での粒子数が対角化されている表示を用いたボーズ系のシミュレーションでは, ワーム演算子として生成消滅演算子を用います. このとき, ワームの前後の虚数時刻では粒子数が 1 だけ異なっていて, このワームが移動すると, ワームが通過した部分の粒子数は通過前から 1 だけ変化し, ワームの移動によって状態が更新されていきます. ワーム更新法は, 上記のループ更新に比べると広い範囲のモデルで有効です.

オリジナルのワーム更新法では, ワームは虚時間方向および実空間方向にランダムウォークしていました. 一方, ワームの空間方向での移動が起きる候補であるバーテックスを, ループ更新と同様にしてあらかじめ作り, ワームはバーテックスにぶつかるまで虚時間方向で直進するようにしたものが向き付きループアルゴリズム (DLA) です. DSQSS/DLA は DLA を実装したプログラムです.

7.3 マルチワームアルゴリズム

DLA ではただひとつのワームヘッドを動かすことで状態を更新しているため, 時空間分割などの非自明並列が行えません. この問題を解決するために提案されたアルゴリズムが, ワーム対の個数制限を取り除いたマルチワームアルゴリズム (MWA) およびその時空間並列版アルゴリズムである PMWA です. DSQSS/PMWA は PMWA を実装したプログラムです.

7.4 on-the-fly バーテックス法

ワーム更新法では, ワームの生成, ワームの時間方向の移動, ワームの空間方向の移動（散乱）, ワームの消滅などからなるサイクルが更新の時間的単位となっていて, このサイクルを繰り返すことで逐次的に状態が更新されます. このうちワームの空間方向の移動を実現する仕方として, 移動の起こる場所（バーテックス）をあらかじめ全系に配置しておくやり方 (固定バーテックス法) と, ワームの進行方向に必要に応じて配置するやり方 (on-the-fly バーテックス法) とがあります. 固定バーテックス法では, 状態更新の基本的単位（1 モンテカルロステップ）は,

1. バーテックスの配置
2. ワームの生成から消滅までの（複数）サイクル

のふたつのフェーズからなりますが, on-the-fly バーテックス法ではこのうちの 2 のみを行います.

DSQSS/DLA では, on-the-fly バーテックス法を, DSQSS/PMWA では固定バーテックス法を採用します.

7.5 バーテックス配置

分配関数に現れる重み $W(S)$ は格子点 i, j 間の相互作用を H_{ij} として,

$$W(S) = \prod_{(ij),\tau} \langle S_i(\tau + \Delta\tau) S_j(\tau + \Delta\tau) | e^{-\Delta\tau H_{ij}} | S_i(\tau) S_j(\tau) \rangle$$

と書けます. ここで, 虚数時間 τ に関する積は $\Delta\tau$ を単位として離散化されているとしました（これは説明のためであり, 実際の計算では連続虚数時間で行われます）. H_{ij} で相互作用する実空間の 2 点 i, j に関して, 虚数時間区間 $[0, \beta)$ は状態 $S_i(\tau)$ か $S_j(\tau)$ に不連続変化がある時刻によって有限個の区間に分割されます. 各区間内では, 2 点の状態は $S_i(\tau), S_j(\tau)$ は一定であり, 密度

$$\langle S_i S_j | E_0 - H_{ij} | S_i S_j \rangle$$

でバーテックスが一様ランダムに確率的に配置されます. ここで, エネルギーシフト E_0 はある定数であり, 上記の密度が正である限り任意にとることができます. 状態一定の区間に配置されるこれらのバーテックスに加えて, 状態変化（キンク）のある虚数時刻に対しては, 確率 1 でその時刻にバーテックスが配置されます. バーテックスはファインマンダイアグラムを用いた分配関数の展開における相互作用グラフに対応します.

7.6 ワームの生成・消滅

ワームの生成にあたっては, まず状態遷移先の候補として, 時空から一様ランダムに時空点 i, τ とワーム対を表す非対角演算子対 Q, Q' を選びます. 次に確率 p_{create} で実際にワーム対を生成します. 消滅はこの逆過程であり, ワーム対が同じ時空点に来た時に一定の確率 $p_{\text{annihilate}}$ で消滅します. これらの確率は次の式で表される詳細つり合い条件

$$\frac{\Delta\tau}{\beta N_{\text{site}} N_Q} \times p_{\text{create}} = p_{\text{annihilate}} \times (\eta\Delta\tau)^2 \langle S_i(\tau) | Q_i | S_i'(\tau) \rangle \langle S_i'(\tau) | Q_i' | S_i(\tau) \rangle$$

が成り立つように定められます. ここで, N_Q は取りうる Q の数で, S' は $\langle S|Q|S' \rangle$ が非ゼロの値を取る（普通は S, Q の組に対して一意に定まる）状態です. また, η はワーム演算子に共役な場の量で, 例えばスピン系でワーム演算子を昇降演算子にとったときは横磁場（の半分）です. DLA では, η の値は任意にとれるので, DSQSS/DLA の補助ツール `dla_alg` では $\eta^{-2} = \beta N_{\text{site}} N_Q \Delta\tau \max_{S,Q} |\langle S|Q_i|S' \rangle|^2$ とすることで

$$p_{\text{create}} = |\langle S_i(\tau)|Q_i|S_i'(\tau)\rangle|^2 / \max_{S,Q} |\langle S|Q_i|S' \rangle|^2$$

かつ $p_{\text{annihilate}} = 1$ としています. `algorithm.xml` をユーザが独自に編集することで, ユーザの指定した生成消滅確率でシミュレーションを行うことも可能です.

7.7 ワームの散乱

バーテックスにおけるワームの散乱は, バーテックス自体の重みとワームの持つ重みとの間に詳細つり合いが成立するように決定されます. たとえば, 虚数時刻が増加する方向にサイト i 上を移動してきたワームが虚数時刻 τ に

あるバーテックスに当たり, サイト j を虚数時刻が減少する方向に出ていく過程を考えると, このような衝突が選ばれる確率 P と, 逆向きの衝突が選ばれる確率 P' との間には

$$P \times \langle S_i(\tau+0)S_j(\tau+0)|E_0 - H_{ij}|S_i(\tau)S_j(\tau)\rangle\langle S_i(\tau)|Q_i|S_i'\rangle$$
$$= P' \times \langle S_i(\tau+0)S_j(\tau+0)|E_0 - H_{ij}|S_i'(\tau)S_j'(\tau)\rangle\langle S_j'(\tau)|Q_j|S_j\rangle$$

の関係が満たされる必要があります. ここで Q_i, Q_j はワームを表す非対角演算子, $S_i'(\tau), S_j'(\tau)$ はワームが通過した後のそれぞれのサイト, 時刻の状態です. DSQSS/DLA では, `algorithm.xml` ファイルをユーザが直接編集することによって, 任意の散乱確率を指定できるほか, ハミルトニアンを指定したときに, このような条件を満たす散乱確率を自動的に計算する補助ツール `dla_alg` を持っています.

7.8 参考文献

- N. Kawashima and K. Harada, "Recent Developments of World-line Monte Carlo Methods", Journal of the Physical Society of Japan, Vol. 73, 1379-1414 (2004).
- J. Gubernatis, N. Kawashima, and P. Werner, "Quantum Monte Carlo Methods; Algorithms for Lattice Models", Cambridge University Press (2016).